3 Configuring Circuits for OVO

About This Chapter

This chapter provides detailed information which aims to help the OpenView Operations administrator better understand the configuration and use of event-correlation circuits in the context of OVO. It assumes familiarity with the terms used both in OVO and ECS. This chapter also explains some of the concepts behind message correlation in OVO and helps the reader decide where to carry out the correlation process. Finally, it gives some hints on how to go about investigating problems relating to OVO-specific correlation circuits. The chapter is divided into the following general sections:

- "Understanding Correlation in OVO"
- "Setting up Event Correlation in OVO"
- "Deciding Where to Correlate"
- "OVO Message Attributes"
- "Automatic Actions and Trouble Tickets"
- "Accessing External Data"
- "Logging Correlation Events in OVO"
- "Troubleshooting"

NOTE

To edit OVO correlation circuits, both the ECS Designer GUI and OVO itself must be installed and running on the same system. For a list of the platforms currently supported by OVO as well as a list of the platforms supported by OVO's agent and management server correlation runtime engines, see the *HP OpenView Operations Installation Guide for the Management Server* or the latest *HP OpenView Operations Software Release Notes*.

Understanding Correlation in OVO

In OVO, messages are generated by the conditions defined in OVO message-source templates. These messages are used as input by OVO's **correlation templates**. The correlation templates then process the OVO messages and, where appropriate, generate new messages, modify existing messages or discard the messages entirely.

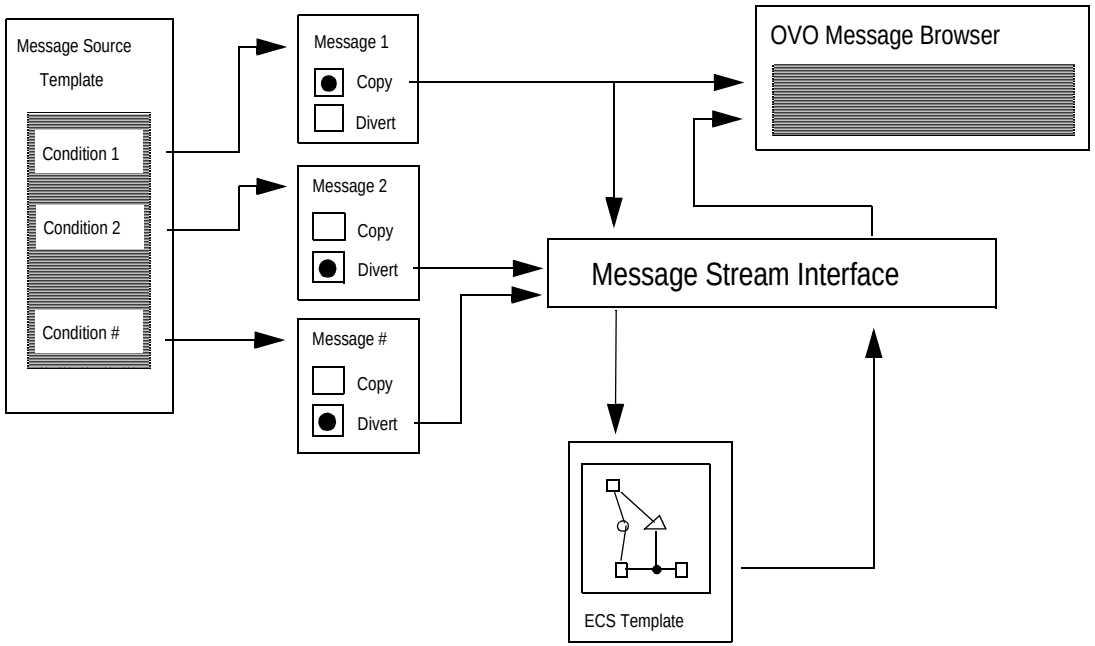
NOTE

In the context of OVO, a correlation circuit is viewed and treated as a template, which may be grouped, assigned, and distributed in the same way as any other OVO template. Consequently, for all practical purposes, an OVO event correlation (EC) template is an ECS correlation circuit.

OVO supplies a set of default correlation templates that you can assign, distribute, and use in the same way as other OVO templates. However, correlation templates may only be modified with the ECS Designer GUI, which is also used to create, verify, and simulate new correlation templates. Default correlation templates serve as examples, which the administrator can modify to meet the demands of a particular environment. For more information on the example correlation templates provided with OVO, see the *HP OpenView Operations Concepts Guide* and the *HP OpenView Operations Administrator's Reference Volume I*.

Figure 3-1 on page 52 shows how the correlation process works in the context of OVO. It also illustrates how the OVO message-source template allows you to either copy or divert messages to the **Message Stream Interface** (MSI), where they can be processed by the correlation template. OVO allows you to copy rather than divert messages to the correlation engine so that critical messages are not delayed or lost in the correlation process. The ability to copy rather than divert messages to the MSI is also very useful in the troubleshooting process.

Figure 3-1 Correlation Flow in OVO



Example Correlation Scenarios in OVO

Event correlation can be used for many purposes. Uses include consolidating multiple messages into a single message, annotating messages with additional information or automating procedures that are currently performed manually. Event correlation can also be used to reduce the number of messages arriving in the `Message Browser` window, allowing the operator to more easily identify the root cause of the problem to which messages relate. For example, you could set up a template on an OVO managed node that correlates messages generated as a result of system-resource limits having been reached:

- maximum number of processes exceeded
- out of disk space
- maximum number of named pipes reached
- maximum number of inodes exceeded
- out of configured shared memory
- out of configured semaphores

You could also monitor MIB values over time to determine if and when action needs to be taken to correct problems relating to the same shared network devices such as printers. For example:

- ”Paper jam” status longer than 5 minutes
- ”Load paper” status longer than 5 minutes
- ”Printer door open” status longer than 2 minutes

Users wishing to make use of the more advanced features in both OVO and the ECS Designer could take the correlation concept a step further by setting up a OVO correlation template to:

- retrieve additional, specific, context-related data
- automatically acknowledge earlier OVO messages
- change OVO message attributes

Setting up Event Correlation in OVO

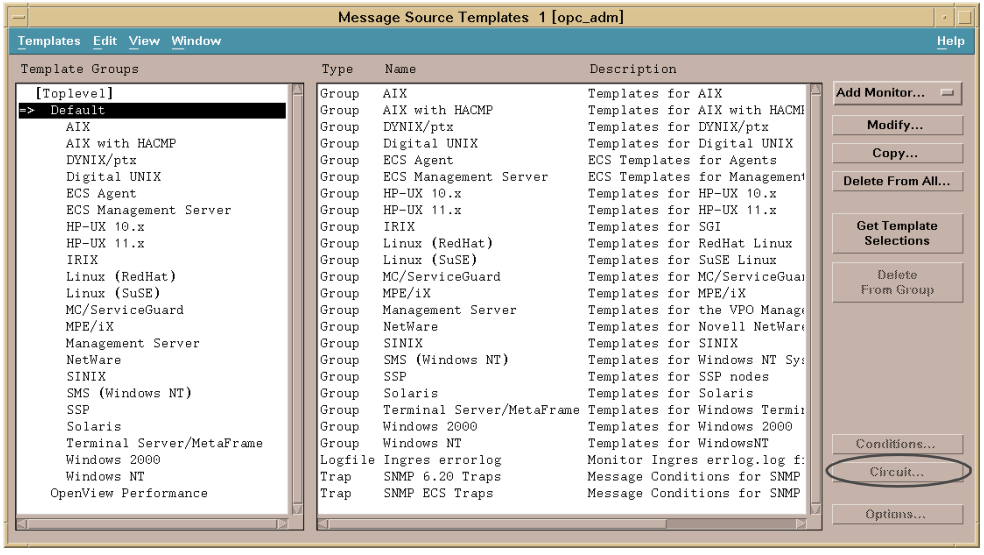
The first step in setting up correlation in OVO is to use the ECS Designer GUI to create the correlation logic. The ECS Designer is also used to define the message types that will be processed by a given correlation template. Once you have designed the correlation template, the appropriate message types must be diverted/copied to the Message Stream Interface (MSI) to make them available for correlation. Finally the correlation template must be distributed to the managed node and/or the management server.

Creating the Correlation Template

Correlation templates (circuits) are created using the ECS Designer product. The ECS Designer is used to build correlation circuits and to simulate their operation for testing and debugging purposes.

If an EC template is selected in the `Message Source Templates` window, and the ECS Designer product is installed, the ECS Designer can be started by clicking on the `[Circuit...]` button.

Figure 3-2 Starting the ECS Designer



The procedure below should be followed to create a new correlation template.

1. Design Correlation Logic.
2. Configure Message Types.
3. Save and Verify Circuit.
4. Test Correlation Logic in Simulate Mode.

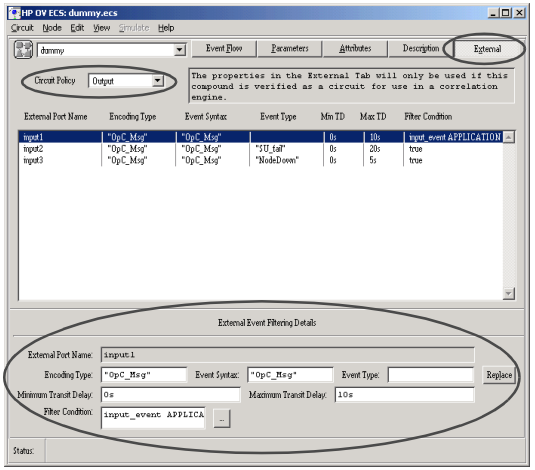
Design Correlation Logic

The ECS Designer is used to define the correlation logic. The process of designing a correlation circuit is beyond the scope of this document. For more information refer to the *HP OpenView ECS Designers Reference*.

Configure Message Types

The `External` tab is where you specify which message types can enter the correlation circuit. It is selected by clicking on the `[External]` button in the top right hand corner of the ECS Designer GUI.

Figure 3-3 ECS Designer External Tab



It is important to configure the External tab to ensure that only the required OVO message types are accepted by the correlation circuit. The External tab has two sections that must be configured:

- Circuit Policy.
- External Event Filtering Details.

Circuit Policy

This must always be set to Output or Unspecified for OVO correlation templates. In depth discussion of circuit policy is beyond the scope of this document. For more information refer to the *HP OpenView ECS Designers Reference*.

External Event Filtering Details

A correlation circuit can have many Input ports. Each Input port can be configured to accept events based on the event's encoding type, event syntax, event type, transit delay and/or an ECDL (Event Correlation Description Language) filter condition. The External tab displays an entry for every Input port that exists in the correlation circuit.

Each ECS Source node receives events from a different Input port. Therefore, adding additional Source nodes to a circuit will result in additional Input ports which must be configured in the External tab.

Encoding Type

In OVO the only events you will be correlating are OVO messages (also called OpC messages). Therefore the Encoding Type must always be set to "OpC_Msg".

Event Syntax

The Event Syntax must also be set to "OpC_Msg".

Event Type

The ECS event_type header attribute matches the OVO Message-type attribute. Therefore, if this field is filled in, only OVO messages with a Message-type attribute that matches the specified Event Type will be allowed to enter the Input port.

This field operates in conjunction with the Filter Condition field (described below). A message will only enter the Input port if it's Message-type attribute

matches the Event Type field *and* the Filter Condition evaluates to true. If the Event Type field is left blank, just the Filter Condition will be used to determine if a given message will enter the Input port.

NOTE

If the Filter Condition field is set to true and the Message-type attribute of a message does not match the Event Type field, the message will not be sent to the correlation process (opceca/m). This means the load on the MSI will be much lower.

See "Ensure Message-type Matches Event Type" on page 107 for more information on setting the Message-type attribute.

Filter Condition

The Filter Condition allows you to enter any valid ECDL expression that evaluates to true or false. This field works in conjunction with the Event Type field (described above). If a message has a Message-type attribute that matches the Event Type field (or the Event Type field is blank), and the Filter Condition expression evaluates to true, the message will enter the Input port.

The Filter Condition is particularly useful if you can not determine whether a given message should enter the circuit just by looking at the Message-type attribute. For example, you may wish to only accept messages from a given Application, or a particular Message-source.

NOTE

If the Filter Condition is set to true, and the Event Type field is left blank, *all* messages that have been diverted to MSI will enter the Input port. This could be the intended behavior if you have a particular correlation template that has been designed to operate on *all* messages that have been diverted to MSI.

For more information on writing ECDL expressions refer to the *HP OpenView ECS Designers Reference*. For more information on OVO message attributes, see “OVO Message Attributes” on page 77.

Save and Verify Circuit

A correlation circuit/template can be saved at any time. However, if you wish to activate the template on a managed node or the management server, you will need to verify the correlation circuit. Select the `Circuit:Verify Circuit` menu item in the ECS Designer GUI.

If you save and verify a correlation circuit, this will cause the ECS rule compiler to transform the circuit data into platform independent ASCII files that are distributed with the EC template, allowing it to be loaded into the ECS engine.

Un-verified templates can still be distributed, but they will not be loaded by the ECS engine. See “Distributing the Correlation Template” on page 59 for information on distributing the correlation template.

Test Correlation Logic in Simulate Mode

The ECS Designer has a simulate mode that can be used to verify the operation of the ECS circuit prior to deploying it in a live environment. The simulator allows you to load logfiles and simulate the operation of the correlation circuit. See “Logging Correlation Events in OVO” on page 100 for more information on obtaining logfiles in OVO.

For more information on creating and modifying correlation circuits or using the ECS Designer simulator, refer to the *HP OpenView ECS Designers Reference*.

Distributing the Correlation Template

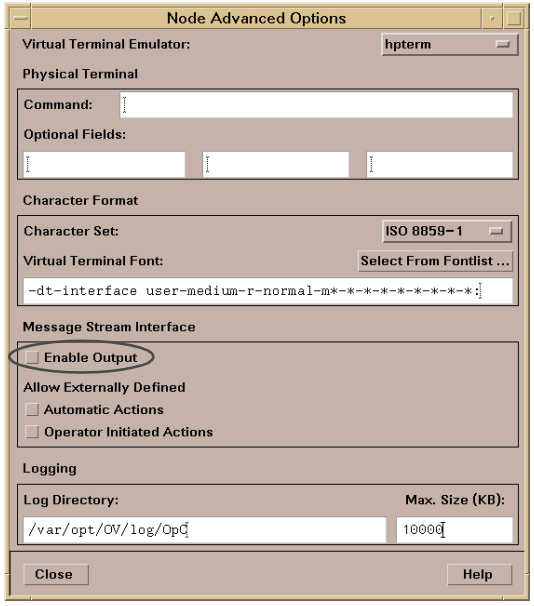
Once you have designed the correlation template, it needs to be distributed. The first step is to ensure that all messages requiring correlation are diverted to the Message Stream Interface (MSI). You then need to distribute the correlation template to the appropriate managed nodes and/or management server.

Enabling MSI

Before diverting messages to MSI, you have to ensure that MSI has been enabled on the managed node and/or management server.

Managed Node: Select the following menu sequence `Actions:Node -> Modify -> Advanced Options`. This will bring up the following window:

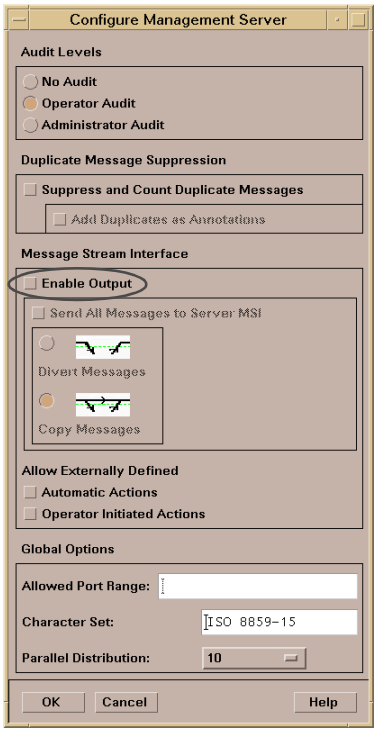
Figure 3-4 Enable Output to MSI - Managed Node



Select the `Enable Output` check box.

Management Server: Select the following menu sequence `Actions:Server -> Configure`. This will bring up the following window.

Figure 3-5 Enable Output to MSI - Management Server

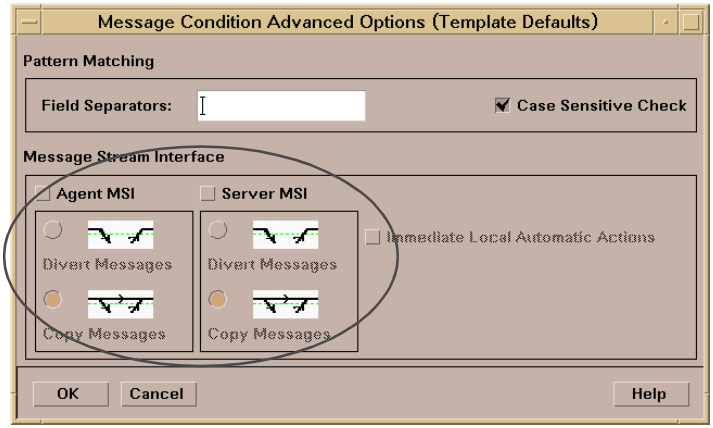


Select the `Enable Output` check box.

Diverting Messages to MSI

Once MSI has been enabled, you need to configure each individual message to either divert or copy to MSI. In the `Message Source Templates` window highlight the message template of interest and select `Modify -> Advanced Options`. This will bring up the following window.

Figure 3-6 **Divert to MSI**



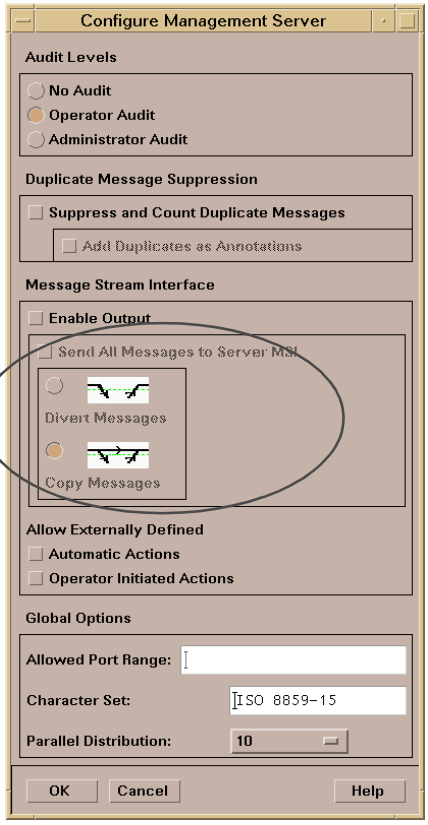
Select the `Agent MSI` or `Server MSI` check box (or both), and select whether you wish to Divert or Copy the message to MSI.

NOTE You would normally only choose the `Copy messages` option when testing to see if your correlation template is working as expected.

Divert *all* Messages to Server MSI

On the management server there is an additional option available. You can divert or copy *all* messages to the MSI. This option will cause all messages received on the management server to be sent to the MSI. Select the following menu sequence `Actions:Server -> Configure`. This will bring up the following window.

Figure 3-7 **Divert ALL to Server MSI**



Select the `Send ALL messages to Server MSI` check box, and select whether you wish to Divert or Copy messages.

Note that this option will cause additional overhead, as every single message received on the management server will be processed to determine if it should enter each correlation circuit.

Distributing EC Templates

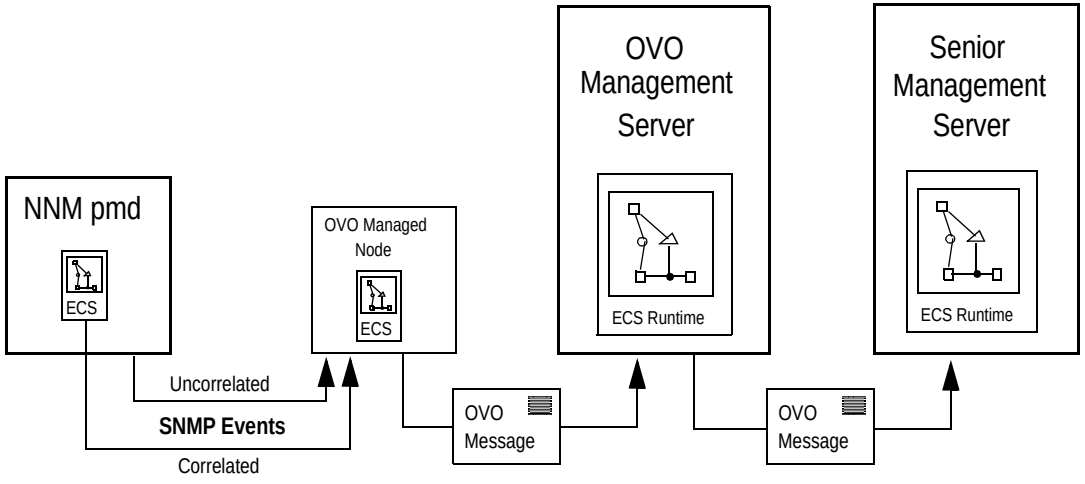
Once the appropriate messages have been diverted to MSI you should distribute the correlation template(s). Event correlation (EC) templates are distributed in exactly the same way as other OVO templates. They should be distributed to the managed node or management server as appropriate. For more information on distributing templates see the *HP OpenView Operations Administrator's Reference Volume I*.

NOTE When a new or modified correlation template is distributed, the ECS engine forwards all open messages to the message agent, `opcmsga`, stops the engine, re-loads all correlation circuits, and restarts. The message agent does not subsequently send back the forwarded messages.

Deciding Where to Correlate

It is important that you take time to consider the advantages and disadvantages of where the correlation takes place in an OVO environment. In a standard environment, this could be a simple choice between the managed node and the management server. There is even a case to be made for doing both. In larger environments using OVO's flexible-management configuration where several layers of management servers are configured hierarchically, the choice of where to correlate widens to include the relationship between the various levels of management server. In addition, setting up event correlation in the NNM domain first can significantly reduce the amount of SNMP-related messages intercepted by OVO's event interceptor. See Figure 3-8 for an idea of the correlation possibilities.

Figure 3-8 Configuring Correlation in OVO



In general terms; the earlier the correlation the better, since the load down-stream is reduced and less messages arrive in the `Message Browser` window. Correlating events in the NNM domain reduces the load on OVO's event interceptor. In the same way, correlating messages on the managed node reduces the number of messages sent to the OVO management server and, as a result, both the general amount of network traffic and the load on the management server is reduced. Correlating messages on the OVO management server allows you to compare and, if necessary, suppress similar or related messages coming from different managed nodes.

There is no need to restrict correlation to individual sources either. Great benefit may be gained by correlating messages from different message sources within OVO, namely: SNMP traps, `opcmsg`, logfiles, and threshold monitoring. For example, messages generated by SNMP events and relating to a node being down might be correlated with messages generated by the logfile encapsulator and relating to entries in a client/server application's logfile about a server that is unreachable.

On the management server, OVO's serial MSI allows you to choose which processes/applications (amongst those that are connected to the MSI) see which messages and in which order. As a general rule, you should configure ECS to be the first application to receive messages, as this allows you to perform correlation first, thus only passing relevant messages to other applications connected to the MSI. For more information on OVO's serial MSI, see the *HP OpenView Operations Concepts Guide* and the *HP OpenView Operations Administrator's Reference Volume I*.

Briefly, you need to consider the following when setting up correlation in the context of OVO:

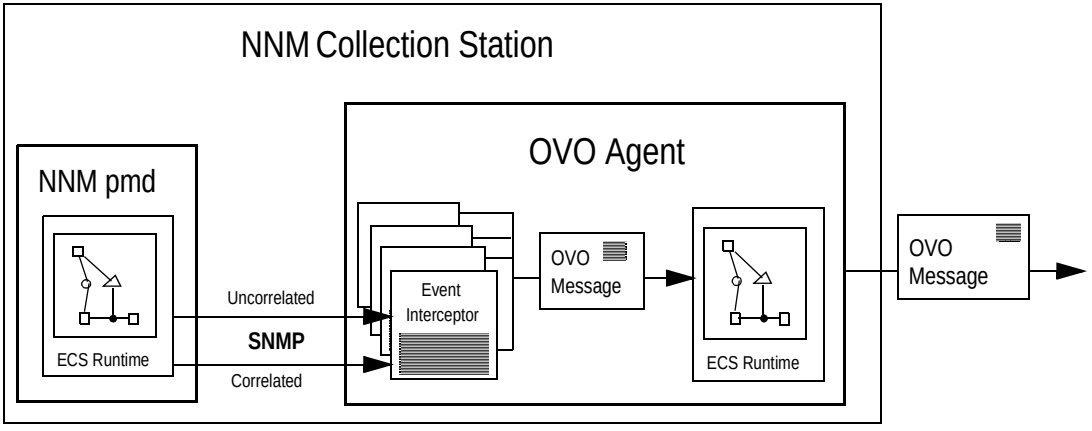
1. Where do you want the correlation to take place:
    a. Before arriving at the managed node, for example; in NNM's post master daemon (pmd)?
    b. On the OVO managed node?
    c. On the OVO management server?
    d. Higher up the OVO flexible-management hierarchy?
    e. On all or a combination of the above?
2. Which type of messages do you want to correlate?
3. Do you want the messages generated by the message-source templates to be copied or diverted to the MSI?
4. Has the correlation template been set up to receive the appropriate message types?

The OVO Event Interceptor

OVO's event interceptor is the link between the NNM domain and the world of OVO. The OVO event interceptor taps the stream of (by default, correlated) SNMP events produced by NNM's postmaster daemon (pmd) and, where appropriate, generates OVO messages. The resulting messages can then be passed through the OVO agent's correlation templates along with messages generated by other OVO sources such as logfiles. For example, if a system acting as an NFS file server generates an SNMP event as a result of an interface going down, the OVO message generated by this SNMP event could be correlated with the messages generated as a result of logfile entries on the NFS clients reporting stale NFS handles. You can greatly reduce the number of events being intercepted by the OVO event interceptor by using NNM's correlation circuits to correlate events before they reach OVO. Clearly this strategy helps to reduce the overall load on the OVO agent and the OVO management server.

For example, if you configure a circuit in NNM to correlate events generated as a result of a router going down and then ensure that the OVO event interceptor on the NNM collection station sees only correlated events, you can greatly reduce the number of OVO messages generated. These messages could then be further correlated by passing them through correlation templates on the OVO managed node.

Figure 3-9 Correlation in NNM



The OVO event interceptor process (`opctrapi`) can be configured to listen to the *correlated* flow or the *raw* flow of events from NNM. The `opcinfo` file controls this behavior.

- managed node (UNIX): `/opt/OV/bin/OpC/install/opcinfo`
- managed node (Windows): `\usr\OV\bin\OpC\install\opcinfo`

Simply add the line

```
SNMP_EVENT_FLOW CORR
```
or
```
SNMP_EVENT_FLOW RAW
```

If neither of the above two lines are present in the file, the event interceptor will default to receive the *correlated* flow of events from NNM.

NOTE If the event interceptor is listening to the *raw* stream of events, *all* events will be received regardless of any correlation configured in NNM.

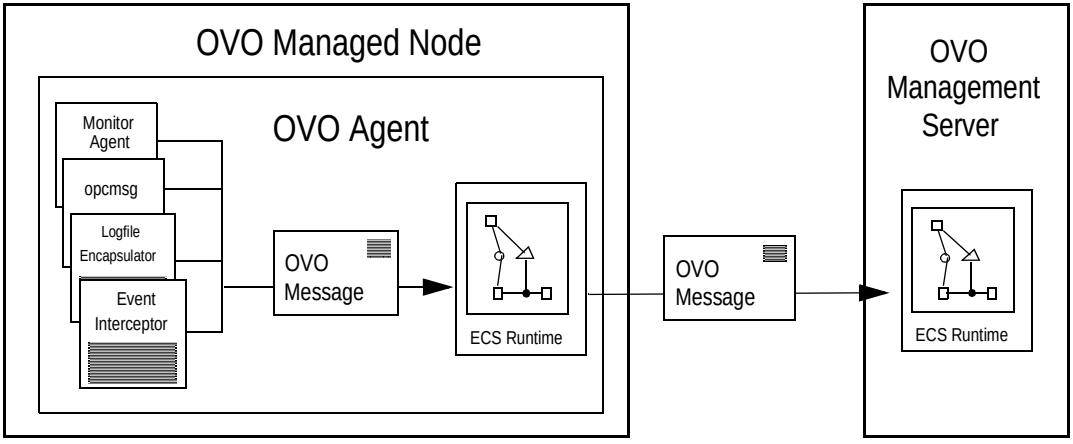
It is possible to view the results of NNM's correlation from OVO. In the OVO `Message Browser`, an annotation indicates that a message has correlated events associated with it. To view the correlated events, start the OVO application `Corr Events` from the `NNM Tools` application group. In addition, if an event is acknowledged in NNM as a result of correlation, the corresponding message will be automatically acknowledged in the OVO `Message Browser`.

For more information on setting up event correlation in Network Node Manager, see the other sections in this manual.

Correlating on OVO Managed Nodes

The main benefit of using the OVO agent to correlate messages on the managed nodes is the significant reduction of network traffic between agent and server. Since this is true for all managed nodes on which the correlation agent is running, another major consequence is the reduction in CPU load on the management server. Note that the OVO agent also runs on the OVO management server.

Figure 3-10 Correlation on the OVO Managed Node



The OVO agent correlation process, `opceca`, connects to the agent MSI to allow access to messages from the OVO agent message stream. Figure 3-11 on page 70 shows how the messages are then correlated and sent back to OVO's message agent. Messages that are created or modified by the agent correlation process appear in the `Message Browser` window with the message source `MSI: opceca`. If a message does not match any of the rules and conditions specified in the correlation circuits/templates, it retains its original message source.

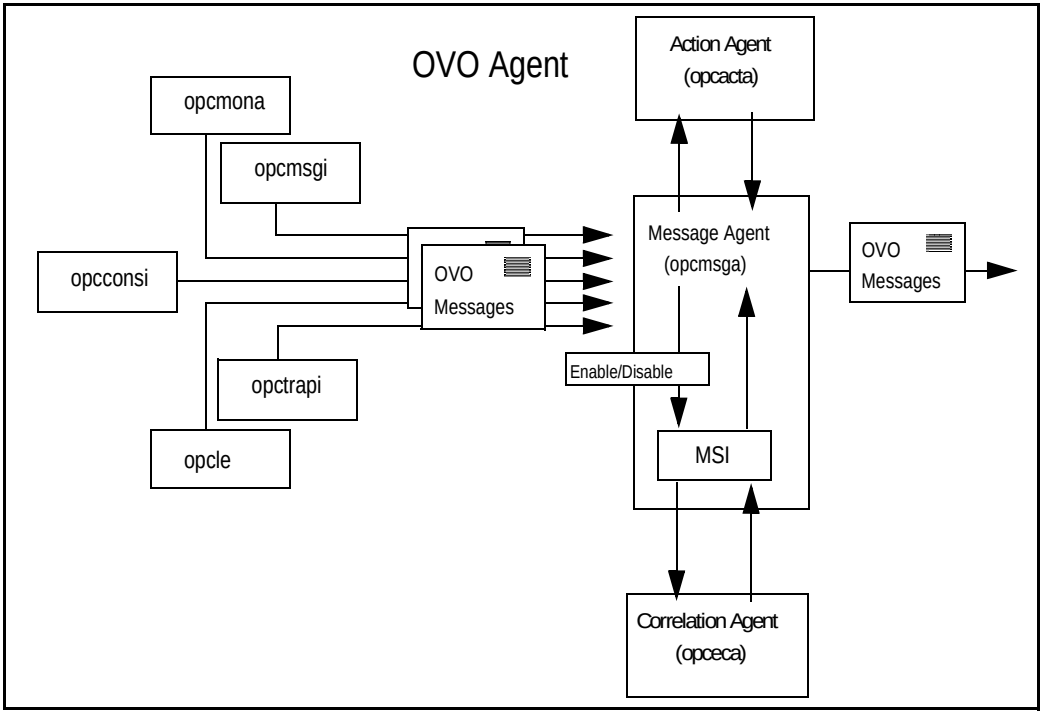
Like all agent processes, opceca is controlled by the Control Agent, opcctla, and if ECS configuration is available on the managed node, you can check `opceca`'s status by using the command: `opc(r)agt -status`. The process `opceca` is started if an ECS template is distributed to the managed node. The process `opceca` is stopped if no more ECS template configuration is available on the managed node.

Correlation templates are assigned to the managed node in the same way as any other OVO template. Although it is possible to assign unverified correlation templates to the managed nodes, only those templates that are successfully verified (using the ECS Designer GUI) will be activated.

NOTE

If any attempt is made to load a correlation circuit/template into a running correlation engine (for example when distributing new or modified correlation templates) the engine forwards all open messages to the message agent, `opcmsga`, stops the engine, loads the circuit, and restarts. The message agent does not subsequently send back the forwarded messages.

Figure 3-11 Message Flow on the OVO Managed Node



NOTE

Automatic actions are not carried out if they are associated with a message that is discarded in the correlation process *and* the `Immediate Local Automatic Actions` option is disabled in the `Message Conditions Advanced Options` window. This option is enabled by default and only available if you have *both* enabled output to the MSI *and* set the `Divert Messages` switch in the `Message Conditions Advanced Options` window.

Any new actions that are required must be defined and associated with a new message or by modifying existing messages during the correlation process. See “Automatic Actions and Trouble Tickets” on page 86 for more information on defining automatic actions from within a correlation circuit.

Correlating on the OVO Management Server

On the OVO management server you can correlate messages from different nodes that relate to the same problem. This reduces the number of messages appearing in the `Message Browser` window. This is particularly useful in an environment with distributed client/server applications, and shared network devices such as printers, backup devices, or NFS file servers. For example, if a database server is momentarily unavailable, the OVO administrator can discard the large number of similar message arriving from managed nodes which have lost contact with the database server. This can be achieved by configuring and assigning a correlation template to the OVO management server that correlates these similar messages and displays a single, relevant (and much more helpful) message in the `Message Browser` window. Figure 3-12 shows the correlation flow on the OVO management server.

Figure 3-12 Correlation on the OVO Management Server

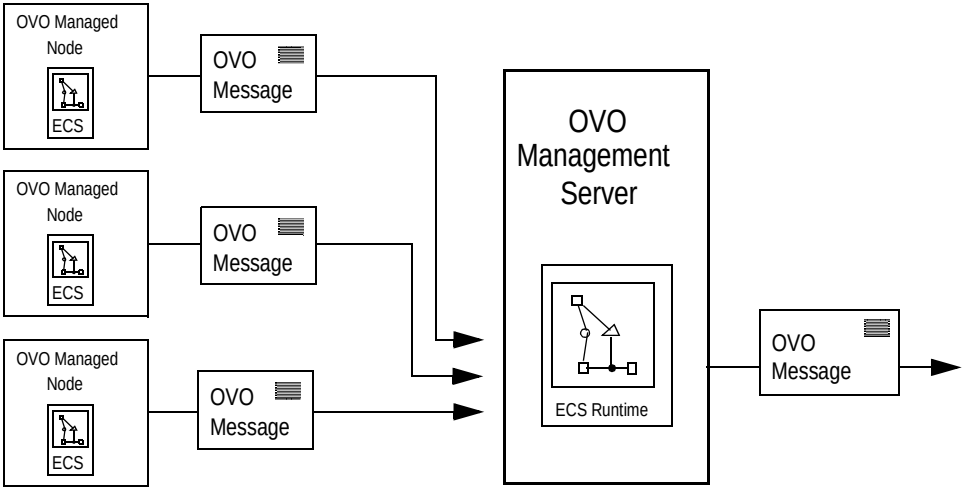
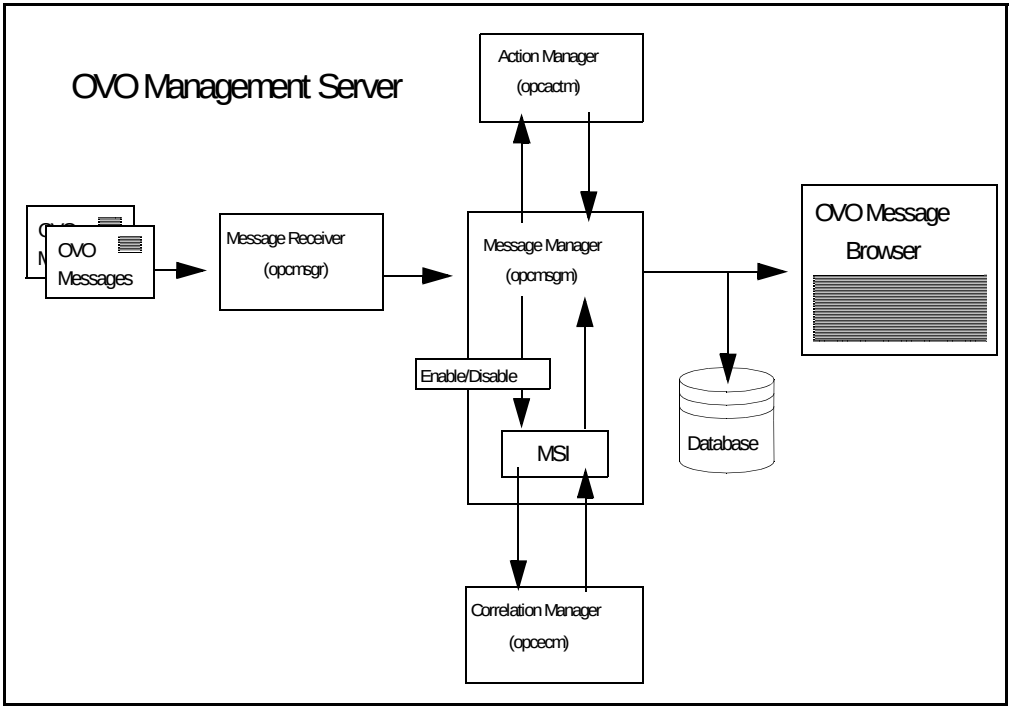


Figure 3-13 on page 73 shows how the OVO correlation server process, `opcecm`, connects to the server MSI to allow access to messages from the message stream. The messages are then correlated and written back to OVO's message manager. Messages that are created or modified by the correlation process appear in OVO `Message Browser` window with the message source `MSI: opcecm`. If a message does not match any of the rules and conditions specified in the correlation circuits/templates, it retains its original message source. Note that the OVO agent runs on the OVO management server, too.

Like all server processes, `opcecm` is controlled by the Control Manager, `opcctlm`. Consequently, if ECS configuration is available on the management server, you can check `opcecm`'s status with the command: `opcsv -status`. The process `opcecm` is started if an ECS template is distributed to the management server. The process `opcecm` is stopped if no more ECS template configuration is available on the management server.

Figure 3-13 Message Flow on the OVO Management Server



Correlation templates are assigned to the OVO management server using the `Management Server Template Configuration` window. Although it is possible to assign unverified correlation templates to the OVO management server, only those templates that are successfully verified (using the ECS Designer GUI) will be activated.

NOTE

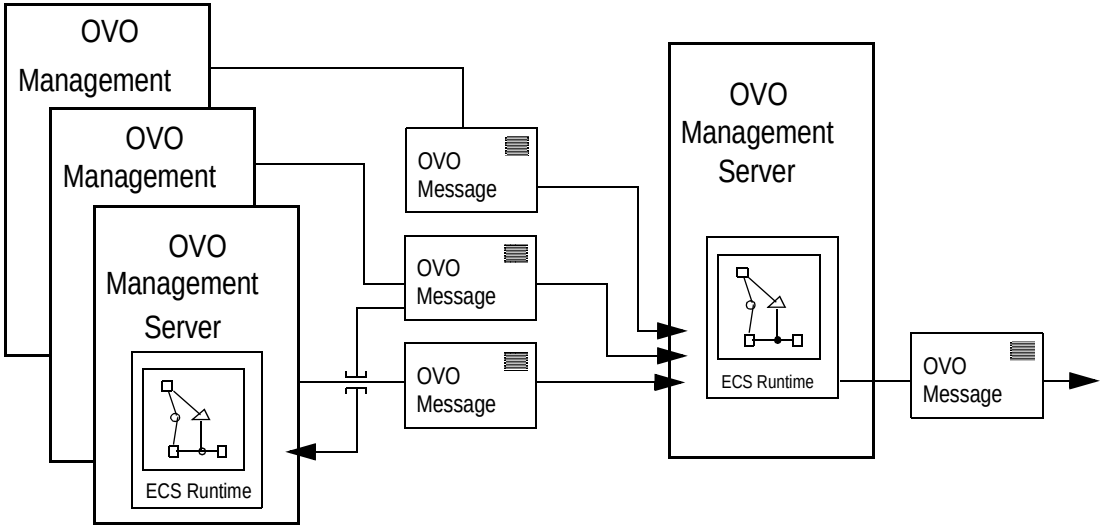
Automatic actions are not carried out if they are associated with a message that is discarded in the correlation process *and* the `Immediate Local Automatic Actions` option is disabled in the `Message Conditions Advanced Options` window. This option is enabled by default and only available if you have *both* enabled output to the MSI *and* set the `Divert Messages` switch in the `Message Conditions Advanced Options` window.

Any new actions that are required must be defined and associated with a new message or by modifying existing messages during the correlation process. See “Automatic Actions and Trouble Tickets” on page 86 for more information on defining automatic actions from within a correlation circuit.

Correlation in Flexible Management Environments

In larger corporate environments that make use of OVO's flexible-management configuration features such as message escalation and message forwarding, message correlation can be seen in a much broader context, which takes into account the relationships between the different levels in the management hierarchy. The correlation relationship between managed nodes and the management server in the OVO environment can be extended to the relationship between the management servers in the management hierarchy. You could, for example, reduce the number of similar escalated messages appearing in the OVO `Message Browser` window. Correlation may also be applied to the relationship between OVO management servers on the same hierarchical level, where message-forwarding has been set up.

Figure 3-14 Correlation with OVO's Flexible Management



Taken to its logical conclusion in a OVO environment where a corporate-wide correlation strategy has been implemented, managed nodes would send only correlated messages to their respective management servers. These management servers could then either correlate these messages and send a newly correlated stream on to the management servers they report to in the level above, or send an uncorrelated stream of messages to be correlated on arrival at the next management server up the chain.

The overall result is increased efficiency and reduced load both on the network and the OVO user. To enable this kind of message correlation in a flexible-management hierarchy, you would assign and distribute management-server correlation templates to OVO management servers on the various levels in the same way as you usually assign and distribute templates to any OVO management server.

OVO Message Attributes

Message attributes are the link between the OVO world and the ECS world. A correlation template can determine which correlation flow an OVO messages will pass through by matching the OVO `Message-type` attribute to the `Event Type` field of the correlation circuit's `Input` port. In addition, by utilizing the `Filter Condition` of an `Input` port, you can look at *any* of the OVO message attributes to determine which correlation flow a message will pass through.

Message attributes can also be referenced in ECS nodes. For example, a Filter node in an ECS circuit would refer to the OVO message attribute `OBJECT` in the following way: `input_event OBJECT`. Similarly, the `CREATION_TIME` message attribute could be accessed as: `input_event CREATION_TIME`. See Table 3-1 for more information on the OVO-specific event attributes that you may use for correlation in the context of OVO.

See the *HP OpenView ECS Designers Reference* for information on OVO's event-header attribute values.

Table 3-1 OVO Event Body Message Attributes

OVO Message Attribute	Type	Description
AACTION_ACK	Boolean	Defines whether or not the message is acknowledged automatically on the OVO management server after the corresponding automatic action has finished successfully.
AACTION_ANNOTATE	Boolean	Defines whether or not OVO creates “start” and “end” annotations for automatic actions.
AACTION_CALL	String	The command to use as automatic action for the OVO message. Default: empty string; max. length: 2000 chars.
AACTION_NODE	String	Defines the system on which the automatic action runs. Default value: the content ‘NODENAME’; max. length: 254 chars

Table 3-1 OVO Event Body Message Attributes

OVO Message Attribute	Type	Description
AACTION_STATUS	Integer	Defines the status of the automatic action belonging to the current message. Possible values are: • ACTION_UNDEF (default) • ACTION_DEF (default if AACTION_CALL is defined) • ACTION_STARTED • ACTION_FINISHED • ACTION_FAILED
APPLICATION	String	Application name to use for the OVO message. Default: empty string; max. length: 32 chars.
CREATION_TIME	Time	The time the message was created. The time is in UNIX format (seconds since Epoch). Default: the (local) time when the message was created.
FORWARDED	Boolean	Read only. Defines whether or not a message is forwarded in an environment configured with manager-to-manager forwarding.
GROUP	String	The OVO message group to use for the message. Default: empty string; max. length: 32 chars.
INSTR_IF	String	The name of the external, instruction-text interface. The external, instruction-text interface must be configured in OVO. Default: empty string; max. length: 36 chars.
INSTR_IF_TYPE	Integer	Defines whether the internal OVO instruction-text interface or an external interface is used to display instructions for the message. Possible values are: • INSTR_NOT_SET (default) • INSTR_FROM_OPC (instruction stored in OVO database) • INSTR_FROM_OTHER (instruction accessed via external instruction-text interface)

Table 3-1 OVO Event Body Message Attributes

OVO Message Attribute	Type	Description
INSTR_PAR	String	Parameters for the call to the external, instruction-text interface. Default: empty string; max. length: 254 chars.
MSGID	String	Read only. Unique identifier of the message. Modified or newly created messages will assume the ID: '00000....'
MSGKEY	String	Summarizes the important characteristics of the event that triggered the message.
MSGKEY_RELATION	String	Specifies the message key relation. This can contain patterns.
MSGKEY_RELATION_ICASE	Integer	Case sensitivity of message key relation: 0 - case sensitive, !=0 - not case sensitive
MSGKEY_RELATION_SEPS	String	Field separators for message key relations.
MSGSRC	String	Read only. This attribute specifies the source of the message, for example, the name of the encapsulated logfile if the message originated from logfile encapsulation or the interface name if the message was sent via an instance of the Message-Stream Interface. Default: empty string; no max. length.

Table 3-1 OVO Event Body Message Attributes

OVO Message Attribute	Type	Description
MSGSRC_TYPE	Integer	Read only. Specifies the source type of the message. Each source is represented in one bit, e.g. a message that was generated by the logfile encapsulator and then modified at the Agent MSI will have 'bit' for LOGFILE_SRC and AGTMSI_SRC set. Possible values are: • OPCMSG_SRC (Message submitted by `opcmsg(1/3)` API) • LOGFILE_SRC • MONITOR_SRC • SNMPTRAP_SRC • SCHEDULE_SRC • CONSOLE_SRC (MPE/iX source) • SVMSI_SRC (MSI on OVO management server) • AGTMSI_SRC (MSI on OVO managed node) • LEGLINK_SRC (Legacy Link interface)
MSGTEXT	String	Message text. Default: empty string; no max. length.
MSGTYPE	String	This attribute is used to group messages into subgroups, e.g., to denote the occurrence of a specific problem. Default: empty string; max. length: 36 ASCII chars, no spaces.
MSG_LOG_ONLY	Boolean	Inserts the message immediately into the history-message log when the message is received on the management server. The message is not sent to any operator. An Operator will only be able to see the message when using the OVO history message browser

Table 3-1 OVO Event Body Message Attributes

OVO Message Attribute	Type	Description
MSI_OUTPUT	Integer	Defines the handling of the message in the OVO Message Stream Interface. Each value is representing one bit that can be bitor'ed. Possible values are: • SV_MSI_NO_OUTPUT (default) • SV_MSI_DIVERT • SV_MSI_COPY • AGT_MSI_NO_OUTPUT (default) • AGT_MSI_DIVERT • AGT_MSI_COPY
NODENAME	String	The name of the system on which the message was created. The message is only handled by the OVO management server if NODENAME is part of the OVO managed environment (OVO node bank). Default: local node name; max. length: 254 chars.
NOTIFICATION	Boolean	Forwards OVO messages from the OVO management server to the OVO Trouble-ticket Notification Service interface(s), if the appropriate notification interface(s) is/are configured and active.
OBJECT	String	The "object" name to use for the OVO message. Default: empty string; max. length: 254 chars.
OPACTION_ACK	Boolean	Defines whether the message is acknowledged automatically on the OVO management server after the corresponding operator-initiated action has finished successfully.
OPACTION_ANNOTATE	Boolean	Defines whether or not OVO creates "start" and "end" annotations for the operator-initiated action.
OPACTION_CALL	String	Command to use as operator-initiated action for the OVO message. Default: empty string; max. length: 2000 chars.
OPACTION_NODE	String	Defines the system on which the operator-initiated action should run. Default value: NODENAME; max. length: 254 chars.

Table 3-1 OVO Event Body Message Attributes

OVO Message Attribute	Type	Description
OPACTION_STATUS	Integer	Read only. Defines the status of the operator-initiated action belonging to the current message. Possible values are: • ACTION_UNDEF (default) • ACTION_DEF (default if OPACTION_CALL is defined) • ACTION_STARTED • ACTION_FINISHED • ACTION_FAILED
ORIGMSGID	String	Read only. Original message identifier. Multiple messages may have the same identifier if they have the same source message.
ORIGMSGTEXT	String	The original message text. This attribute allows you to set additional source information for a message. It is useful if the message text was reformatted but the OVO operator needs to have access to the original text. Default: empty string; no max. length.
READ_ONLY	Boolean	Read only. Defines whether or not a message is forwarded as a “notification” in an environment configured with manager-to-manager forwarding.
RECEIVE_TIME	Time	Read only. Defines the time the message was received by the management server. The time is in UNIX format (seconds since Epoch). This value is set by the management server.
SERVICE_NAME	String	Defines whether or not an attribute influences the status of a service.

Table 3-1 OVO Event Body Message Attributes

OVO Message Attribute	Type	Description
SEVERITY	Integer	The severity of the message. Possible values are: • SEV_UNKNOWN • SEV_NORMAL • SEV_WARNING • SEV_MINOR • SEV_MAJOR • SEV_CRITICAL
TIME_ZONE_DIFF	Integer	Defines the difference in seconds between UTC and local time at the time a message is created.
TROUBLETICKET	Boolean	Defines the forwarding of OVO messages from the OVO management server to the OVO trouble-ticket (TT) interface, if the TT interface is configured.
TROUBLETICKET_ACK	Boolean	Defines that the OVO management server acknowledges the message automatically if forwarding of the message to the trouble ticket system was successful.
UNMATCHED	Boolean	Read only. Defines whether or not the message matched a condition.

All attributes can be read in a correlation circuit. Those that are not marked read-only can also be altered (by a Create or Modify node).

Using Event Attributes

OVO event attributes can be addressed as String data types, or as Tokens. This means that you can, for example, address the OBJECT message attribute using the String value `"OBJECT"` or the Token value `OBJECT`.

Using the Token address form has the advantage that a misspelled token name is picked up when the circuit is validated (compiled) rather than at run time. Tokens are also more efficient than Strings. You should also remember that case is important with Strings, but is not important when Tokens are used. So `OBJECT`, `object` and `"OBJECT"` are acceptable, but `"object"` will raise a run time error.

All other ECS documentation assumes that the attributes of events are addressed as String values.

Examples

To filter events based on their severity you could enter a Filter node `Condition` parameter such as:

```
input_event SEVERITY = SEV_UNKNOWN
```

To change messages with a severity of SEV_UNKNOWN to SEV_NORMAL you could use a Modify node after the Filter node above, with a `Modify Spec` parameter such as:

```
input_event alter (SEVERITY => SEV_NORMAL)
```

To retain just the latest event from a particular OBJECT in a Table node, you could enter a Delete Condition parameter such as:

```
current_event OBJECT = retained_event OBJECT
```

Custom Message Attributes

Custom message attributes allow the user to add individual attributes to an OVO message. This effectively allows the user to extend the OVO message format with additional fields. For example a custom field could be created for SLA type, customer name, and so on. Custom message attributes are defined in the `Custom Message Attributes` window, where you can enter name/value pairs.

Custom message attributes can be accessed from within ECS by specifying the attribute name as a String data type. It is also possible to create new and/or modify existing custom message attributes from within ECS.

Examples

To filter events based on a custom message attribute called `customer` you could enter a Filter node `Condition` parameter such as:

```
input_event "customer" = "CustomerA"
```

To change the custom message attribute `SLA_type` to `Level1` you could use a Modify node after the Filter node above, with a `Modify Spec` parameter such as:

```
input_event alter ("SLA_type" => "Level1")
```

Creation of new custom message attributes can be done in exactly the same way. To create a new attribute called `fail_type` with a value of `Primary` you could use a Modify or Create node, with a `Modify Spec/Create Spec` parameter such as:

```
input_event alter ("fail_type" => "Primary")
or
created_event alter ("fail_type" => "Primary")
```

NOTE

In the above example if the `fail_type` custom message attribute does not exist, it is automatically created for the given message, and assigned the value `Primary`.

Automatic Actions and Trouble Tickets

An automatic action is an external action that takes place automatically when a message is received by OVO. Automatic actions are generally configured in the message source templates, but they can also be configured in a correlation circuit. This is very useful, as it allows you to define automatic actions for new messages that are created by a correlation circuit. It is also possible to specify whether a message should be sent to the defined trouble ticketing interface.

Example

Let us assume you have a correlation circuit that creates a message indicating a database has gone down, where the `MSGTEXT` attribute contains a list of all the affected applications. You wish to use an automatic action to e-mail the `MSGTEXT` to the system administrator. In addition, you can forward the message directly to the trouble ticketing system to ensure the problem gets fixed as soon as possible. The Modify node `Modify Spec` could be configured something like the following:

```
let
   val messageText : string = input_event MSGTEXT
   val aaCommand : string = "/var/opt/OV/bin/OpC/actions/sendMail.sh DB_Down"
in
   input_event alter (
     AACTION_CALL => aaCommand + " sysadmin@a.network " + messageText,
     AACTION_NODE => "ovo_server1.a.network.com",
     AACTION_ANNOTATE => true,
     AACTION_ACK => false,
     TROUBLETICKET => true,
     TROUBLETICKET_ACK => false)
end
```

See "OVO Message Attributes" on page 77 for more detail on the automatic action and trouble ticket attributes.

NOTE

This functionality requires that externally defined automatic actions are enabled. Select the following menu `Actions:Node -> Modify -> Advanced Options`, and tick the `Allow Externally Defined: Automatic Actions` check box.

Accessing External Data

It is often necessary for a correlation circuit/template to access information from an external source. This external information might include parameters that alter the behavior of the correlation circuit, thus allowing you to modify the circuit operation without re-compiling it in the ECS Designer. For example, you may create a correlation circuit for transient faults that is applicable for many different message types. The list of message types could be kept external to the correlation circuit, allowing you to add additional message types without having to re-compile the circuit.

External information may also be required to assist in making correlation decisions or possibly to enrich the information output from the correlation circuit. For example, on detection of a given error message, you could query an external database to obtain service level agreement (SLA) information. The SLA details could then be added to the message before it is output to the `Message Browser` window.

There are two methods of accessing external data from within a correlation circuit:

- Data and Fact Stores.
- Annotate Mechanism.

Data and Fact Stores

The ECS data store and fact store allow you to separate the environmental aspects of correlation from the rules and basic logic that are “hard-coded” into an OVO correlation template. For example, you could configure a generic correlation template for one OVO managed node and, with the help of the data store, use the same template on other managed nodes.

The data store can be used to hold key-value pairs of information. This might include system-specific information relating to the correlation template as well as external network details such as intervals, threshold values, and other constants if you know that these details might change. For example, you could configure one generic correlation template to monitor user switches (using the `su` command) and then run the template on many different OVO managed nodes by storing the names of “trusted users” (which might change from system to system) in the data store.

The fact store is a data structure that defines relationships. These relationships could describe the hierarchy of an organization, for example, or the relationships between one organization and another, or between services and service providers. For example, if three application servers, A, B, and C, are linked to the same database server DBserver01 and a fourth application is linked to a different database server DBserver02, you could use the fact store to define these relationships. If DBserver01 goes down for any reason, the information in the fact store could be used to correlate the database server down message with the messages concerning lost contact with application servers A, B, and C.

Fact and data stores are text files that are loaded into the ECS engine when an EC template is distributed to the managed node or management server. They have a file extension of `.ds` for the data store and `.fs` for the fact store. Note that fact and data store files are not distributed with the EC template. They must be manually created and copied to the following location *prior* to distributing the EC template.

- management server: `/var/opt/OV/conf/OpC/mgmt_sv/`
- managed node (UNIX): `/var/opt/OV/conf/OpC/`
- managed node (Windows): `\usr\OV\conf\OpC\`

Each correlation circuit can access only one fact store and one data store file. There are two types of fact and data store in OVO:

- Specific Data Stores and Fact Stores.
- Global Data Store and Fact Store.

Global fact and data stores can be shared across multiple correlation circuits. Specific fact and data stores can only be accessed by a single correlation circuit.

Specific Data Stores and Fact Stores

Each correlation template has a compiled ECS circuit file (`.eco`) associated with it. The name of the compiled circuit file is automatically generated by OVO when the correlation template is first created, and will look something like `EAAAa03015.eco`.

When a correlation template is distributed to a managed node or the management server, it checks to see if a fact or data store exists (in the directory defined above) with the same name as the compiled circuit file. If they do *not* exist, the global fact or data store will be used.

To obtain a list of EC template names and the corresponding compiled circuit name, run the following command on the management server that has the ECS Designer installed:

```
opc_chg_ec -list
```

The `opc_chg_ec` command can also be used to rename a compiled circuit file. This is useful, as the automatically generated name is often difficult to remember. The format of the command is

`opc_chg_ec` *<generated_circuit_name> <new_name>*

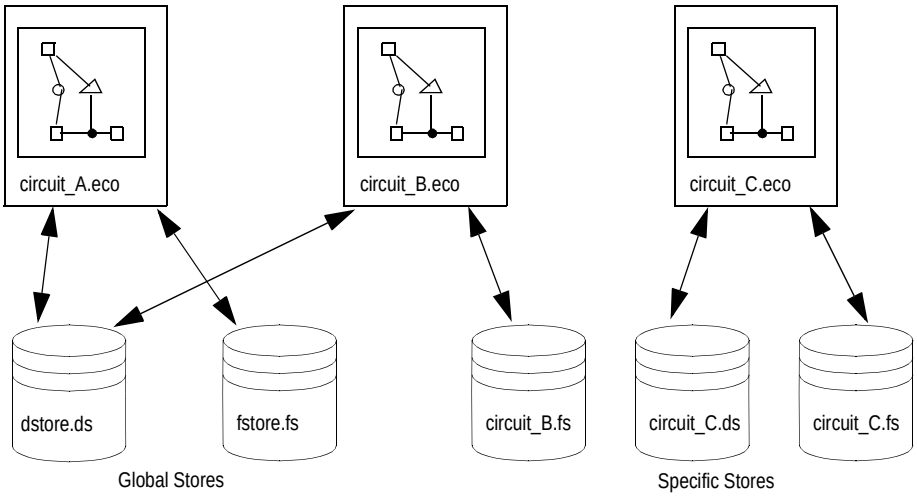
For more information on the `opc_chg_ec` utility, refer to the man page *opc_chg_ec(1M)*.

Global Data Store and Fact Store

The global data store file is called `dstore.ds`, and the global fact store file is called `fstore.fs`. These stores can be shared across multiple correlation circuits. If a specific fact store does not exist for a given correlation template, the global fact store will be loaded. If a specific data store does not exist for a given correlation template, the global data store will be loaded.

Figure 3-15 on page 90 demonstrates correlation circuits accessing global and specific fact and data stores.

Figure 3-15 Data Stores and Fact Stores in OVO



Example

Let's assume you have a correlation template called "DatabaseDown" that will be distributed to a UNIX managed node. The correlation template accesses configuration data from a *specific* data store, and database relationships from the *global* fact store. You run the `opc_chg_ec -list` command which tells you that the compiled circuit for the DatabaseDown correlation template is called `EAAAa03016`. You can then change the compiled circuit name to be the same as the template name by typing the following command:

```
opc_chg_ec EAAAa03016 DatabaseDown
```

The specific data store should then be copied to `/var/opt/OV/conf/OpC/DatabaseDown.ds` on the managed node.

The global fact store file should be copied to `/var/opt/OV/conf/OpC/fstore.fs` on the managed node.

When the DatabaseDown correlation template is distributed, it will load the `DatabaseDown.ds` data store file, as it has the same name as the compiled circuit file. Since there is no `DatabaseDown.fs` file, it will load the global `fstore.fs` fact store file.

Updating Fact Stores and Data Stores

When OVO is re-started, or when a new or modified correlation template is distributed to the managed node or management server, the fact store and data store files are re-loaded. It is also possible to manually force individual fact and data store files to be re-loaded using the `ecsmgr` command-line utility. The `ecsmgr` utility exists in the following location:

- management server: `/opt/OV/bin/OpC/`
- managed node (UNIX): `/opt/OV/bin/OpC/utils/`
- managed node (Windows): `\usr\OV\bin\OpC\install\`

When using `ecsmgr` you need to specify whether you are communicating with the managed node ECS engine or the management server ECS engine. The management server is instance `11`. The managed node is instance `12`.

The following two commands can be used to update a correlation template with a new fact store and/or data store:

- **Data Store**

  ecsmgr -i *<instance>* -data_update *<datastore_name>* \
  <filename>

  Using the above example, this would be

  ```
  ecsmgr -i 12 -data_update DatabaseDown
  /var/opt/OV/conf/OpC/DatabaseDown.ds
  ```

- **Fact Store**

  ecsmgr -i *<instance>* -fact_update *<factstore_name>* \
  <filename>

  Using the above example, this would be

  ```
  ecsmgr -i 12 -fact_update fstore
  /var/opt/OV/conf/OpC/fstore.fs
  ```

NOTE This command will only update dynamic circuit parameters. Statically evaluated parameters will not be affected. For more information on static and dynamic ECS parameters, refer to the *HP OpenView ECS Designers Reference*.

If the fact or data store files are exceptionally large, it is recommended to perform incremental updates using smaller files. This is to prevent the correlation process from blocking for a long period of time.

For more information on how both the fact and data store work and the syntax requirements that exist for their respective contents, see the *HP OpenView ECS Designers Reference*. For more information on the `ecsmgr` utility, refer to the man page *ecsmgr(1M)*.

Automating Distribution of Fact Stores and Data Stores

Although fact stores and data stores are not distributed with the correlation template, it is possible to partially automate their distribution. The path names in the following example assumes distribution to a UNIX managed node. The first step is to create the fact and data store files in the OVO commands (`cmds`) directory on the management server (`/var/opt/OV/share/databases/OpC/mgd_node/customer/<arch>/cmds`). When you distribute the commands to the managed node, it will cause the fact and data store files to be installed in the `/var/opt/OV/bin/OpC/cmds` directory.

You could then write a simple script (also in `/var/opt/OV/share/databases/OpC/mgd_node/customer/<arch>/cmds`) that copies the fact and data store files from `/var/opt/OV/bin/OpC/cmds` to `/var/opt/OV/conf/OpC/`.

When you wish to distribute the fact and data stores you can simply distribute “commands” to the managed node and run the script by using the OVO broadcast command mechanism.

In some circumstances the ECS Annotate node feature may be more suitable for accessing volatile data than fact stores or data stores. This is fully described in the following section.

Annotate Mechanism

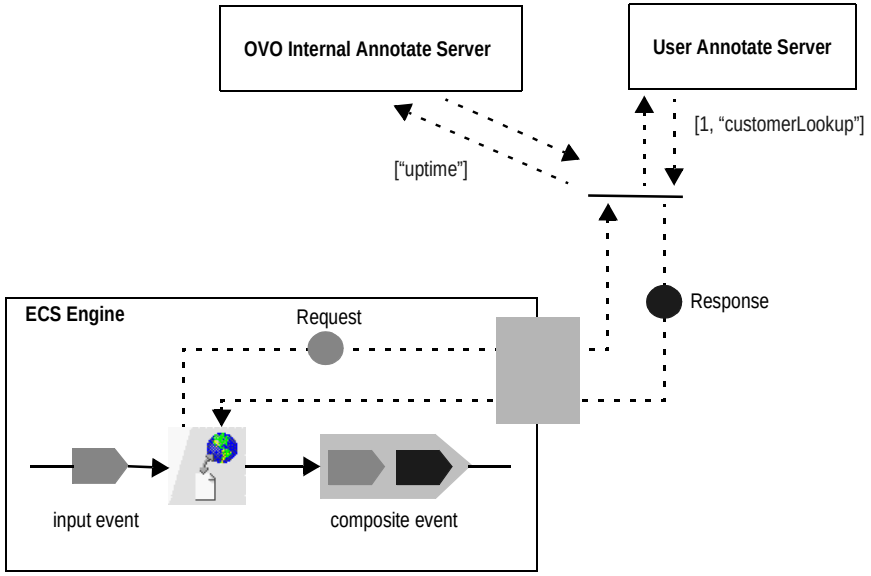
The ECS `Annotate` node allows you to access external sources of information from within a correlation circuit. The annotate node is generally used in preference to the fact store or data store when accessing information that is liable to change on a regular basis.

The annotate node makes a call outside the ECS engine to an external process. The external process is known as an **Annotation Server**. The annotation server performs the appropriate processing and returns the information to the annotate node. This information can then be used within the correlation circuit to assist in decision making or to enrich the information output from the circuit.

There are two types of annotation server available in OVO:

- Internal Annotation Server
- User-built Annotation Server

Figure 3-16 Annotate mechanism in OVO



Selecting the Appropriate Annotation Server to Process a Request

The `Annotate_Spec` parameter of the annotate node is used to send requests to the annotation server. By default, if the first element of the list is a string, the OVO internal annotation server will intercept the request and execute the string as a command. If the first element of the list is any data type other than a string (an integer for example), the OVO internal annotation server will *not* intercept the request. Instead, the request will be available for any user-built annotation servers.

It is possible to override the default behavior described above. A more elegant solution involves each annotation server registering to receive requests only from annotate nodes with specific *names*. You can set the `OPC_ECS_ANNO_NODE` variable in the following file:

- management server: `/opt/OV/bin/OpC/install/opcsvinfo`
- managed node (UNIX): `/opt/OV/bin/OpC/install/opcinfo`
- managed node (Windows): `\usr\OV\bin\OpC\install\opcinfo`

If the `OPC_ECS_ANNO_NODE` variable is set, only annotation requests from nodes with the specified names will be processed by the OVO internal annotation server. The format of the variable is as follows:

`OPC_ECS_ANNO_NODE <name1>[,<name2>...]`

For instance, to configure the Internal Annotate Server so that it only processes annotate requests from annotate nodes named `OVOEXE`, the entry would appear as:

```
OPC_ECS_ANNO_NODE OVOEXE
```

NOTE Future releases of OVO may include this definition by default. As such, it is recommended that this configuration be used.

Note that if you wish to have multiple annotate nodes in your correlation circuit with the same name, you will need to place each annotate node within its own compound node to avoid naming conflicts. Refer to the *HP OpenView ECS Designers Reference* for more information on compound nodes.

Refer to the *HP OpenView ECS Developer's Guide and Reference* for information on configuring a user-built annotation server to accept requests from specific node names.

Internal Annotation Server

The internal annotation server is used to execute commands or scripts, returning the results to the correlation circuit. The `Annotate_Spec` parameter of the annotate node must contain the full path to the command/script that is to be executed. Any parameters must also be included. The OVO internal annotation server will return both the exit code and the standard output of the command/script.

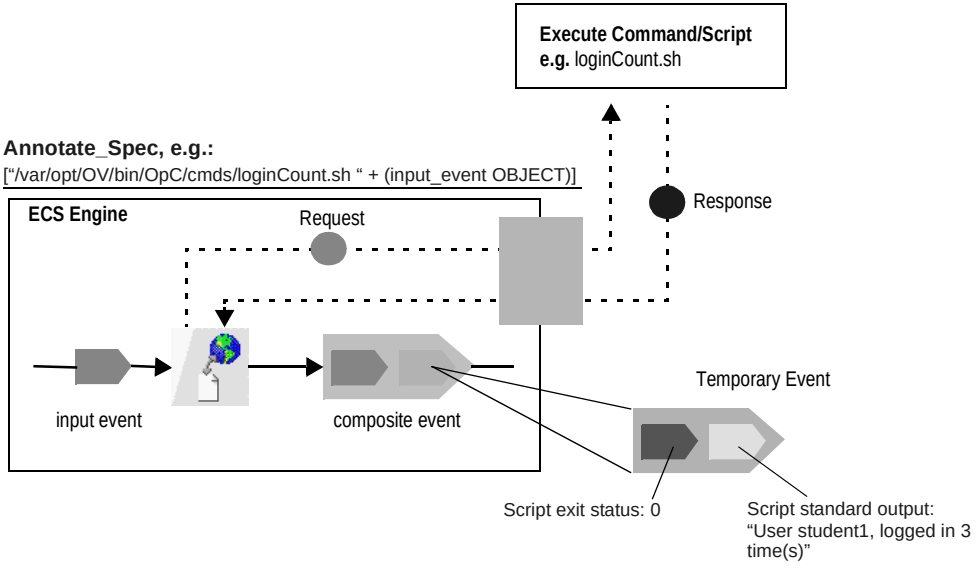
Data returned from all annotation servers is placed in the second sub-event of the composite event that is output from the annotate node. As can be seen in Figure 3-17, "OVO Internal Annotation Server," the data returned from the OVO internal annotation server contains two pieces of information; the exit code and the standard output of the command/script. To obtain the exit code in a filter node you would use:

```
input_event 2 1
```

To obtain the standard output of the command/script:

```
input_event 2 2
```

Figure 3-17 OVO Internal Annotation Server



- **Annotation Script/Command Parameters**

  When sending an annotation request to the OVO internal annotation server, the `Annotation_Spec` must be a list containing a single string. The string value is the fully qualified filename of the script/command to execute, followed by any parameters. For example:

  ```
  ["/var/opt/OV/bin/OpC/cmds/loginCount.sh student1"]
  ```

  This will cause the `loginCount.sh` script in the given location to be executed, passing a single parameter `student1`. This parameter would equate to the first positional parameter ($1) in the script. Additional parameters would be subsequent positional parameters in the script ($2, $3, ...)

  Commonly, parameters passed to the script/command are obtained from the message attributes of the input message. For example:

  ```
  ["/var/opt/OV/bin/OpC/cmds/loginCount.sh " + (input_event
  OBJECT)]
  ```

- **Script/Command Location**

  The script/command that is executed when the annotation request is made is not distributed/installed automatically as part of the EC template distribution/installation. There is no fixed location for the script, so the absolute path should be provided in the `Annotation_Spec`.

  As a general guideline, it is suggested to place the script files in the OVO commands (`cmds`) directory (`/var/opt/OV/share/databases/OpC/mgd_node/customer/`*`<arch>`*`/cmds`). This will cause these files (on a distribution/install) to be installed in the `/var/opt/OV/bin/OpC/cmds` directory on the managed nodes. This path would then be used in the `Annotation_Spec` for the command to execute. For example:

  ```
  ["/var/opt/OV/bin/OpC/cmds/loginCount.sh
  <rest_of_command_line_parameters>"]
  ```

- **Annotation Script Results**

  Since the annotation response contains the script exit status and the standard output of the script, the script would typically use `exit` and `echo` commands (or equivalent) respectively to produce the desired response.

- **Example Script**

  The following script shows that for a user name passed in as the first parameter, it will product a text string stating how many times that user is currently logged on.

```
#!/bin/sh
# loginCount.sh script
COUNT=`who | grep $1 | wc -l`
echo "User $1, logged in $COUNT time(s)"
exit 0
```

User-built Annotation Server

An annotation API is provided for users that wish to develop their own annotation server process. User-built (custom) annotation servers are separate processes that register with the ECS engine and listen for annotation requests. Once a request is received, the annotation server must process the request, and send a response back to the correlation circuit using the annotation API's. User-built annotation servers can be used for many purposes, such as accessing an SLA database, querying MIBs on a network device, or querying topology information from another application. For more information on the development of a custom annotation server, refer to the *HP OpenView ECS Developer's Guide and Reference*.

For more information on using the annotate node in your correlation circuit refer to the *HP OpenView ECS Designers Reference*.

Simulating the Annotate Node in the ECS Designer

The ECS Designer has a simulate mode that can be used to test a correlation circuit before deploying it in a live environment. It is not possible to communicate directly with annotation servers from the ECS Designer, but it is possible to create simulated annotation responses, allowing you to test your circuit. To create the simulated annotation server responses, follow these steps:

1. Temporarily alter the circuit so that no events are output.
2. Enter simulate mode of the ECS Designer and run some sample events through the circuit containing the annotate node. The annotation requests will be placed into the output event browser. These can be saved by using the `[Save]` button at the bottom of the browser.

   The requests will have a format similar to:

   ```
   0

   20010424044754.000000Z

   ["Data sent from Anno_Spec"]

   % anno:request:

   1
   ```

3. To turn these requests into responses, alter the line `%anno:request:` to `%anno:response:`. Modify the list to contain the data to be sent back to the annotate node.
4. In order to add a (for example) 2 second delay to the response, alter the line `%anno:response:` to `%anno:response:2`.
5. The annotation responses can then be loaded into the simulator.

Logging Correlation Events in OVO

Figure 3-18 on page 101 shows how the OVO correlation process on the management server and the managed node log messages going into and out of the ECS engine in the files `ecevilg` and `ecevolg` respectively. These logfiles are used for testing and debugging and reside in the following location:

- management server: `/var/opt/OV/log/OpC/mgmt_sv/`
- managed node (UNIX): `/var/opt/OV/log/OpC/`
- managed node (Windows): `\usr\OV\log\OpC\`

The ECS engine uses the logfile `ecevilg` to store input events. The ECS Designer uses the logfile `ecevilg` as its simulation input. If you wish to simulate the operation of a circuit running on the managed node, you have to transfer the logfile manually from the managed node to the OVO management server on which the ECS Designer is running.

NOTE

The ECS Designer requires a `.evt` suffix for the ECS input logfiles. You have to add this suffix manually. For example, on UNIX you could simply type the following command:

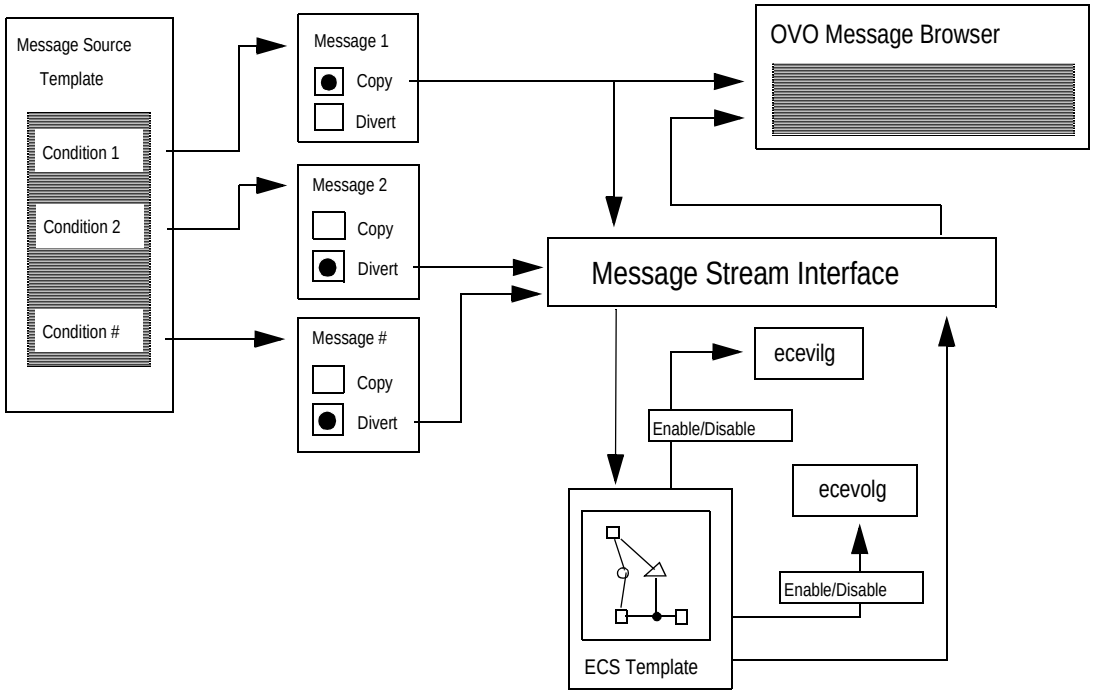
```
cp ecevilg ecevilg.evt
```

In order to rebuild an original OVO message from the content of the ECS logfile, the logfile must contain as much information as possible relating to messages passing through the correlation process. Consequently, the structure of all standard OVO message logfiles (`opcmsglog`) is designed to ensure that they contain the required message attributes in the appropriate format.

NOTE

Custom message attributes are *not* currently logged in the event logs. Therefore it is not possible to simulate custom message attributes in the ECS Designer.

Figure 3-18 ECS Event Logs in OVO



You can switch on the logging of input and output events for a given correlation template in the `Options` window, which you access using the `[Options...]` button in OVO's `Message Source Templates` window.

NOTE Switching on logging for any of the EC templates you assign and distribute to a managed node will switch on logging for *all* EC templates on that managed node.

Event Log Format

The format of OVO messages in the event input (ecevilg) and output (ecevolg) logfiles is shown below:

```
> bd6bb7da-e32a-71d5-014a-0f887fe20000
> 1006860878 ; Tue Nov 27 12:34:38 2001
> karton.bbn.hp.com (IP) ; Normal ;
> Matched ; succeeded_su ; Logfile: Su (10.x HP-UX)
> Security ; /usr/bin/su(1) Switch User ; ejike
>  ; :
> INSTR_NOT_SET:  ;
> AA:  ;  ;  ; Undef
> AA:
> OA:  ;  ;
> OA:
> Succeeded switch user to oracle by ejike
> SU 03/19 16:43 + ttyp6 ejike-oracle
>
> -3600
>
>
>
> MKR_ICASE ; ,
>
% OpC_Msg::
# 1006860878
+0
```

The values shown in the message above are matched to OVO message attributes as shown below. The notation [OPACTION_ACK: "ACK" | ""] means: depending on the OPACTION_ACK attribute, the log contains either ACK or nothing; no quotes are present in the logfile itself.

```
> [MSGID[;ORIGMSGID]]
> [CREATION_TIME] ; [CREATION_TIME (in ASCII)]
> [NODENAME] ([net_type: usually IP]) ; [SEVERITY] ;
  [MSG_LOG_ONLY: "LOG_ONLY"|""]
> [UNMATCHED: "Matched"|"Suppressed"|"Unmatched"] ; [MSGTYPE] ;
  "Console"|"OpC"|"Logfile"|"Monitor"|"SNMP"|"MSI": [MSGSRC]
> [GROUP] ; [APPLICATION] ; [OBJECT]
> [NOTIFICATION: "NOTIFICATION"|""] ; [TROUBLETICKET: "TT"|""]:
  [TROUBLETICKET_ACK: "ACK"|""]
> [INSTR_IF_TYPE:
  "INSTR_NOT_SET"|"INSTR_NOT_SET"|"INSTR_FROM_OTHER"]:
  [INSTR_IF] ; [INSTR_PAR]
> AA: [AACTION_NODE] ; [AACTION_ACK: "ACK"|""] ;
  [AACTION_ANNOTATE: "ANN"|""] ; [AACTION_STATUS:
```

```
  "Undef"|"Def"|"Started"|"Finished"|"Failed"]
> AA: [AACTION_CALL]
> OA: [OPACTION_NODE] ; [OPACTION_ACK: "ACK"|""] ;
  [OPACTION_ANNOTATE: "ANN"|""] ; [OPACTION_STATUS:
  "Undef"|"Def"|"Started"|"Finished"|"Failed"]
> OA: [OPACTION_CALL]
> [MSGTEXT]
> [ORIGMSGTEXT]
> [SERVICE_NAME]
> [TIME_ZONE_DIFF]
> [READ_ONLY: "READ_ONLY"|""]
> [FORWARDED: "FORWARDED"|""]
> [MSGKEY]
> [MSGKEY_RELATION_ICASE: "MKR_ICASE"|""] ; [MSGKEY_RELATION_SEPS]
> [MSGKEY_RELATION]
% OpC_Msg::(time difference between log time and CREATION_TIME)
# (log time in seconds since epoch)
(time difference to next message)
```

Where:

`>`	Marks the start of a new line.
`;`	Separates attributes on the same line.
XXX	Literal text that appears in the message.
`[...]`	A message attribute value, as described in Table 3-1 on page 77. The square brackets may enclose a list of literal values, but are not present in the message itself.
`"..."`	A named value (the quotes are not present in the message itself – empty quotes "" indicate that a missing value is acceptable.).
|	Separates alternative values.
(...)	Encloses a descriptive comment.

Troubleshooting

There are many techniques you can use to investigate correlation-related problems in the context of OVO. If correlation is not working as expected it could be a problem with the message source templates, a template distribution issue, or there could be a problem with the correlation circuit logic. You should follow the steps below to track down the location of the problem.

1. Ensure the Correlation Process is Running.
2. Confirm EC Template Distribution.
3. Look for ECS-related Errors in opcerror File.
4. Verify Diversion to MSIVerify Diversion to MSI.
5. Ensure Message-type Matches Event Type.
6. Check if Message is Being Generated by the OVO Message-source Template.
7. Examine Event Flow in Trace File.
8. Debug Correlation in ECS Designer Simulator.
9. Use ecsmgr Command to Debug Correlation.

Ensure the Correlation Process is Running

- For local correlation on the managed node, run the following command:

  ```
  opc(r)agt -status
  ```

  — If the correlation process is not running refer to “Confirm EC Template Distribution” on page 105.

- For central correlation on the management server, run the following command:

  ```
  opcsv -status
  ```

  — If the correlation process is not running refer to “Confirm EC Template Distribution” on page 105.

Confirm EC Template Distribution

You can use the `opctemplate` and `ecsmgr` commands to verify that the EC template(s) have been distributed as intended. Run the following command to list all the templates that are installed:

```
opctemplate -l
```

You can check the output of this command to ensure the EC template(s) have been distributed. For more information on the `opctemplate` command, refer to the man page *opctemplate(1M)*.

The `ecsmgr` utility is a command-line utility that can also be used to verify the EC template distribution. It exists in the following directory:

- management server: `/opt/OV/bin/OpC/`
- managed node (UNIX): `/opt/OV/bin/OpC/utils/`
- managed node (Windows): `\usr\OV\bin\OpC\install\`

When using `ecsmgr` you need to specify whether you are communicating with the managed node ECS engine or the management server ECS engine. The management server is instance `11`. The managed node is instance `12`. Run the following command:

`ecsmgr -i <instance> -info`

This command will provide you with detailed information on the ECS engine. This includes information on all circuits/templates that are currently loaded in the ECS engine and the time they were loaded.

For more information on the ecsmgr utility, refer to the man page *ecsmgr(1M)*.

Look for ECS-related Errors in `opcerror` File

There is not a separate file for the ECS engine log. All engine log data is written to the OVO error logfile, `opcerror`. This logfile contains ECS engine operation information such as warnings and errors in processing, and output from the ECS `system.audit_log` function. The logfile exists in the following location.

- management server: `/var/opt/OV/log/OpC/mgmt_sv/opcerror`
- managed node (UNIX): `/var/opt/OV/log/OpC/opcerror`
- managed node (Windows): `\usr\OV\log\OpC\opcerror`

- **Managed node**

  1. Look for the following errors in the managed node logfile:

     ```
     05/15/97 08:54:22 ERROR opceca    (Event Correlation Agent)(7163)
     [mpiclt.c:714]: MSI instance 'opceca' already exists. (OpC20-2048)

     05/15/97 08:54:22 ERROR opceca    (Event Correlation Agent)(7163)
     [opceca.c:407]: Cannot open MSI. Return error code -34. (OpC30-1351)

     05/15/97 08:54:22 ERROR opcctla   (Control Agent)(7156) [genctla.c:6816]:
     Event Correlation Agent aborted; process did an exit 1 (OpC30-1040)
     ```

  2. If the above errors are present in the logfile you have to remove the queue and pipefiles in the temporary directory:

     ```
     rm /var/opt/OV/tmp/OpC/opcecaq
     ```

     ```
     rm /var/opt/OV/tmp/OpC/opcecap
     ```

  3. Re-start the correlation process (`opceca`):

     ```
     opcagt -start opceca
     ```

- **Management server**

  1. Look for the following errors in the management server logfile:

     ```
     05/15/97 14:21:55 ERROR opcecm    (Event Correlation Manager)(9689)
     [mpiclt.c:714]: MSI instance 'opcecm' already exists. (OpC20-2048)

     05/15/97 14:21:55 ERROR opcecm    (Event Correlation Manager)(9689)
     [opcecm.c:407]: Cannot open MSI. Return error code -34. (OpC30-1351)

     05/15/97 14:21:55 ERROR opcctlm   (Control Manager)(9681)[procctl.C:337]:
     Process 'Event Correlation Manager' terminated with exit code 1.
     (OpC20-2834)
     ```

  2. If the above errors are present in the logfile you have to remove the queue and pipefiles in the temporary directory:

     ```
     rm /var/opt/OV/share/tmp/OpC/mgmt_sv/opcecaq
     ```

     ```
     rm /var/opt/OV/share/tmp/OpC/mgmt_sv/opcecap
     ```

  3. Re-start the correlation process (`opcecm`):

     ```
     opcsv -start opcecm
     ```

Verify Diversion to MSI

MSI output must be enabled on the management server and/or managed node. This allows messages to be diverted or copied to the MSI, thus making the messages available to the ECS engine. See "Enabling MSI" on page 59 for information on verifying that output to MSI is enabled.

The option `Divert/Copy Messages` to MSI must also be set for *every* message condition that outputs messages to the correlation engine. See "Diverting Messages to MSI" on page 61 for more information on diverting messages to the MSI.

Ensure `Message-type` Matches `Event Type`

The OVO `Message-type` attribute must be set correctly for each message condition. The OVO administrator or template administrator generally defines message types: a message type is usually the name of the sub-group to which the message belongs. You can set the `Message-type` attribute by opening the following sequence of windows:

`Window:Message Source Templates -> Conditions -> Condition Number #: Message Type`

The `Message-type` for a given message condition should match the `Event Type` field of the appropriate correlation `Input` port. The "appropriate" `Input` port is the one that starts the correlation flow that processes the message.

See "External Event Filtering Details" on page 56 for more information on `Message-type` and `Event Type`.

Check if Message is Being Generated by the OVO Message-source Template

ECS Input/output Logging

One way to find out whether a message is being generated by the OVO message-source template is to use ECS input logging. If a message reaches the ECS engine it will be logged in the ECS input log (`ecevilg`).

You can switch on the logging of input (and output) events for a given correlation template in the `Options` window, which you access using the `[Options...]` button in OVO's `Message Source Templates` window.

You can also use the ECS output log (`ecevolg`) to determine if a message was output by the ECS engine. This can help you determine if the problem is in the correlation template, or an OVO configuration problem such as message browser configuration or other OVO message flow problems.

Disable Output to MSI

Another method of determining if a message has been generated by the OVO message-source template in the first place (since if it were indeed being generated by the template but then correctly discarded by the correlation process, it would not appear in the `Message Browser` window) is to disable the output to the MSI.

- **Managed node**

  De-select the `Enable Output` check box for the `Message Stream Interface` in the `Node Advanced Options` window.

  `Actions:Node -> Modify -> Advanced Options`

- **Management server**

  De-select the `Enable Output` check box for the `Message Stream Interface` in the `Configure Management Server` window.

  `Actions:Server -> Configure`

This ensures that the message passes directly to the `Message Browser` window. Alternatively, you can set the `Divert/Copy message to MSI` switch to `Copy`.

NOTE Make sure that you redistribute the OVO message source template after changes such as toggling the `Divert/Copy message to MSI` switch.

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 `Message Browser` window, then you know the message conditions in the OVO message-source templates are generating messages as expected and that, if there is a problem, it lies in the correlation template. As a consequence, you can concentrate further investigation on the event-correlation process itself, which means examining and testing the correlation circuit.

Examine Event Flow in Trace File

Checking the trace files on the managed node or management server allows you to look for the right message flow to and from the event correlation engine.

- **Switch on tracing for local correlation on managed node.**

  First you need to modify the `opcinfo` file. It is located in the following directory:

  UNIX: `/opt/OV/bin/OpC/install/opcinfo`

  Windows: `\usr\OV\bin\OpC\install\opcinfo`

  Insert into the `opcinfo` file

  ```
  OPC_TRACE TRUE
  OPC_TRACE_AREA ALL
  OPC_TRACE_TRUNC FALSE
  ```

  Inform the agent processes that tracing is switched on:

  ```
  opcagt -trace
  ```

  Tracing is reported in the following file:

  UNIX: `/var/opt/OV/tmp/OpC/trace`

  Windows: `\usr\OV\tmp\OpC\trace`

- **Switch on tracing for central correlation on management server.**

  Insert into `/opt/OV/bin/OpC/install/opcsvinfo`

  ```
  OPC_TRACE TRUE
  OPC_TRACE_AREA ALL
  OPC_TRACE_TRUNC FALSE
  ```

  Inform the server processes, that tracing is switched on:

  ```
  opcsv -trace
  ```

  Tracing is reported in the file

  ```
  /var/opt/OV/share/tmp/OpC/mgmt_sv/trace
  ```

  For example, assume a message of type “SU Succeeded” was set up for correlation. Entries like the following would be found in the tracefile:

1. `05/15 16:32:12.193 opcmsga(8678:001)[MSG]: Write message to MSI: 0529f926-cd30-71d0-0bd3-0f887b9d0000 Security 'Succeeded switch use' 15.136.123.157`
2. `05/15 16:32:12.295 opceca(8683:001)[MSG]: Put msg into ECS engine:0529f926-cd30-71d0-0bd3-0f887b9d0000 Succeeded switch user to root by bstefan`
3. `05/15 16:32:12.297 opceca(8683:001)[MSG]: Got msg from ECS engine:0529f926-cd30-71d0-0bd3-0f887b9d0000 Succeeded switch user to root by bstefan`
4. `05/15 16:32:12.305 opcmsga(8678:001)[MSG]: Message received from MSI: 0529f926-cd30-71d0-0bd3-0f887b9d0000 Security 'Succeeded switch use' 15.136.123.157`

— If messages are registered in the tracefile till point 2 but not in point 3, they are lost in the event correlation engine and you can assume that the cause of the problem is an ECS engine problem or a design problem in the correlation circuit.

— If messages are registered in the tracefile in point 2, 3 and 4 but are not shown in the OVO message browser, you can assume that this is an OVO configuration problem such as message browser configuration or other OVO message flow problems.

For more information on trace file messages, refer to the *HP OpenView Operations Administrator's Reference Volume I*.

Debug Correlation in ECS Designer Simulator

If you find there is a problem with your correlation circuit you can use the ECS Designer's Simulator to test your correlation logic.

Using the Simulator, you can load the ECS input log, allowing you to trace the messages through your circuit. See "Logging Correlation Events in OVO" on page 100 for information on obtaining the ECS input log. For more information on simulating and debugging a correlation circuit, refer to the *HP OpenView ECS Designers Reference*.

Use ecsmgr Command to Debug Correlation

The `ecsmgr` utility is a command-line utility that can be used to assist in the debugging of correlation circuits. It exists in the following directory:

- management server: `/opt/OV/bin/OpC/`
- managed node (UNIX): `/opt/OV/bin/OpC/utils/`
- managed node (Windows): `\usr\OV\bin\OpC\install\`

When using `ecsmgr` you need to specify whether you are communicating with the managed node ECS engine or the management server ECS engine. The management server is instance `11`. The managed node is instance `12`. The following two commands can be particularly useful:

- `ecsmgr -i <instance> -info`
  This command will provide you with detailed information on the ECS engine. This includes information on all circuits that are loaded in the engine and also the time they were loaded.
- `ecsmgr -i <instance> -stats verbose`
  This command will provide you with detailed information on messages that have been input to and output from every node in every correlation circuit that is loaded in the ECS engine. This can be very useful for tracking the path that a particular message has followed in the ECS circuit.

For more information on the ecsmgr utility, refer to the man page *ecsmgr(1M)*.