HP Operations Manager for UNIX: Correlation Techniques

White Paper

Executive Summary

A common problem in network, systems, and application management is the inappropriate handling of events. If there is a fault or incident in the managed environment, such as a network outage or a performance bottleneck, operators can be flooded with a burst of events indicating symptoms of an underlying issue. The aim of event correlation is to reduce the number of events, and to enrich the meaning of the events shown to operators. Ideally, the event correlator condenses the received events to a single event, directly indicating the problem in the managed environment.

As an acknowledged leader in IT event correlation and analysis, HP Operations Manager (HPOM) provides several mechanisms to address this space:

- **Visual Correlation**

  Provides the context of an event, such as the services affected by the event and the impact the event has on service health. It also provides a quick way to show a relationship between multiple events that are affecting a given service.

- **Message Correlation**

  Provides built-in HPOM correlation, based on duplicate message suppression and smart message correlation techniques.

- **Event Correlation**

  Provides rich information content that can tie together multiple related symptoms into a reduced volume of events, and possibly even a single root cause event.

These techniques provide correlation at both the local managed node and centrally, offering multiple points of correlation. Local correlation, recommended wherever possible, allows optimal scalability, reducing network traffic and processing overhead at the central management server.

HPOM and HP Network Node Manager (HP NNM) include the HPOM Correlation Composer and Event Correlation Services (ECS) run-time components for configuring event correlations across combined network, application, and platform management environments. Taken together, these tools provide a powerful cross-domain correlation capability for root cause analysis and classification. For even more sophisticated event correlation, the ECS Designer provides a development environment for additional correlations, which can be applied to the ECS runtime.

From simple policy-based message correlation, it is a small learning curve to correlate the majority of "well known" problems with the GUI-based Correlation Composer, and to use the ECS Designer's more flexible, developer-oriented GUI-based tool for more complex correlation requirements.

Introduction

With the proliferation of server partitioning, clustering, and web-based applications with dynamic dependencies across multiple platforms and components, coupled with ever increasing quantities of servers and constant change, a fault or performance bottleneck can result in a large burst of events, each of which represents different symptoms of the root cause.

Presenting all these raw events to operators results in significantly increased effort on the part of users to investigate each event, and places an expectation on users to mentally correlate the different events to an underlying root cause. Alternatively, operators may simply choose to ignore certain events, or miss important events altogether, in a message storm that may result in longer than necessary periods of service unavailability.

HPOM is a leader in IT event correlation and analysis (for details, refer to "Magic Quadrant for IT Event Correlation and Analysis, 2006," Gartner). HPOM provides a multi-pronged approach to correlation, which helps you to better control and shape the myriad events that occur in the IT infrastructure. HPOM uses visual correlation, message correlation, and event correlation to provide more meaningful content and context to the consuming user or application, such as a help desk.

Visual Correlation

Visual correlation provides the context of an event. It shows the ripple effect of an event on the IT environment by using the modeled relationships between dependent components, applications, and services. This context provides users with information about the priority and types of end users affected by an event. It also provides operators with a mechanism to visually see relationships between events that are affecting a particular application or service, even if they have not been consolidated through event correlation.

HPOM Service Navigator is an integral part of the HPOM Java-based GUI. Its purpose is to map the events detected by HPOM event management into service views, as defined by the HPOM administrator, coupled with auto-discovered infrastructure components. Discovery is performed by the HPOM agents and by integration with HP BAC Application Mapping. Service views are used to show how the services relate to infrastructure elements and organizations.

Instead of focusing on individual events within a complex IT environment, and wondering what impact they have on business-critical IT services and priorities, the service view shows the relationship between the event, the part of the IT infrastructure affected, and all IT services (and any parent IT services) that rely on that infrastructure.

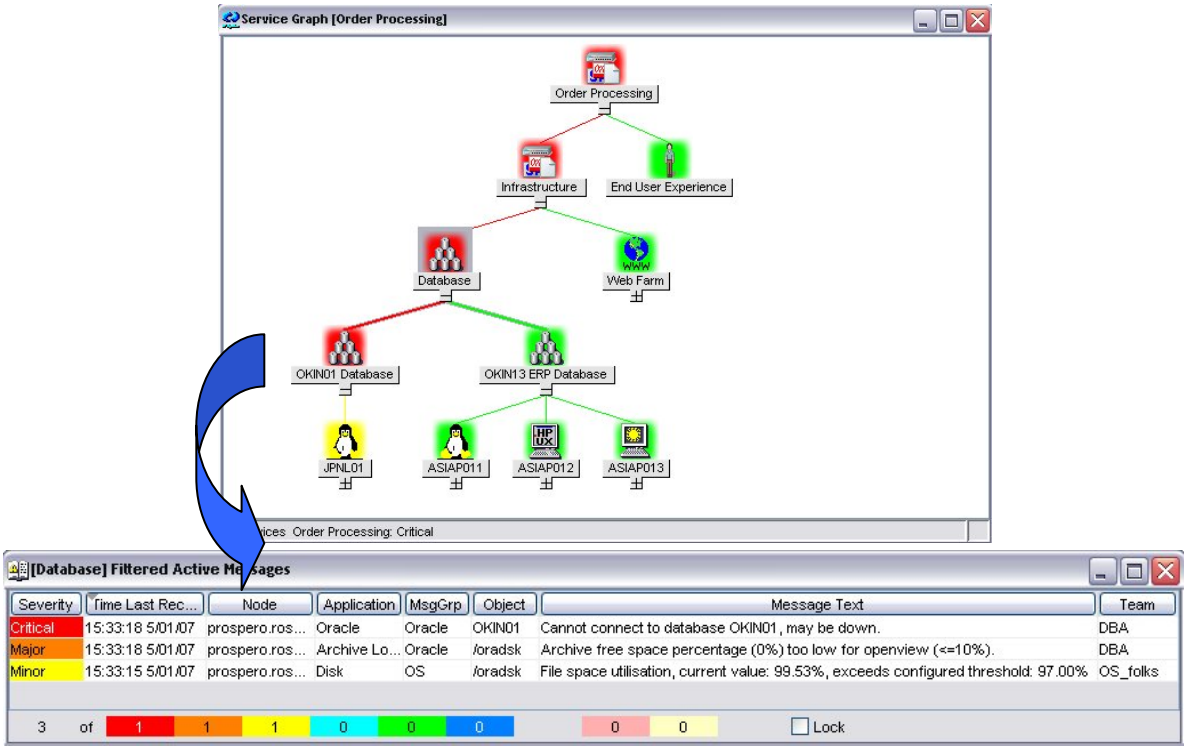
For example, you may detect events about a full file system, an unavailable database, and a full archive log. These events may be correlated, by using event correlation, into a single event that indicates that a database is unavailable because there is insufficient space on the file system for archive logging. However, this event correlation does not tell you which IT services depend on this database, and the impact this dependency has on the delivery of the business service.

The Service Navigator service views provide this information by doing the following:

- Modeling the relationships between infrastructure and higher-level services in a hierarchical fashion
- Applying calculation and propagation rules to realistically represent impact, priority, and service health

Figure 1 shows a service view that points to underlying events.

Figure 1. Service view that points to underlying events



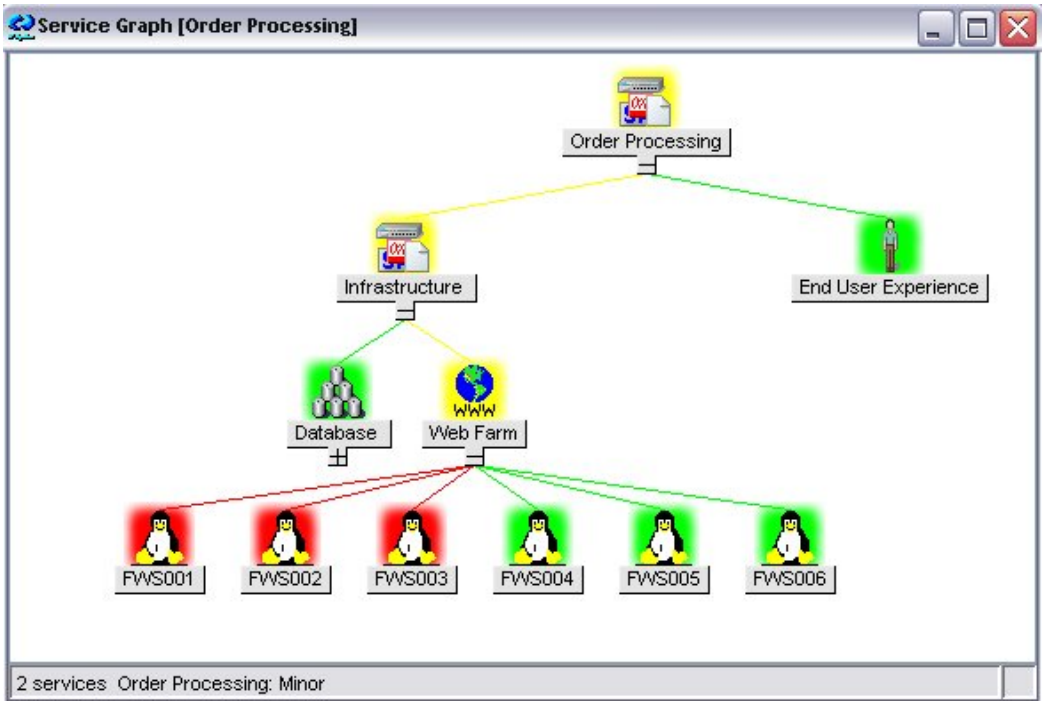
In the example mentioned above, the service views may show that the database is part of a business-critical online order processing service, which draws operator attention to the need to respond with the highest priority. In addition, when this online order processing service becomes critical, an automatic action may be generated to alert line-of-business (LOB) managers through email of the situation. Or they and other stakeholders may see the same service view of service health in their operational dashboard (for example, in HP Operations Dashboard).

The calculation and propagation rules provide a more sophisticated way to model service impact closer to reality. For example, a service view may represent a web server farm that consists of 10 web servers. That farm is used by the online order processing service. If one web server fails, the online order processing service is not affected. If half of the web servers are down or experiencing significant performance issues, the service may be impaired. If 90% of the servers are down or overloaded, the service may be significantly impaired. Service Navigator's calculation and propagation rules model this requirement. The online order processing service may change the state from "Normal" to "Warning," or to a higher severity, as the number or percentage of web servers are affected.

In other words, one unavailable web server does not cause the online order processing service to become critical. But 90% or 100% of unavailable web servers does.

Figure 2 shows a service view with a calculation rule that represents degrees of impact to a web farm, based on the percentage of underlying degraded web servers.

Figure 2. Service view that shows service impact to a web farm



Service views provide a visualization of the relationships between IT components and services, as well as the relative impact of availability- and performance-related issues.

Service views result in the following:

- Increased operator efficiency because root cause can be pinpointed faster
- Reduced downtime because highest-impact events can be prioritized faster
- Increased visibility and awareness among stakeholders (for example, LOB managers) using real-time notification, operational dashboard updates, and service availability reports

Correlating Messages and Events

Message replacement is achieved through message correlation and event correlation.

Message Correlation

Message correlation can be achieved with built-in HPOM mechanisms, offering basic correlation techniques. Message correlation is recommended as the starting point before proceeding to a more sophisticated event correlation mechanism.

Available message correlation techniques include the following:

- **Duplicate Message Suppression**

  Suppressing duplicate messages occurs at the local managed node and at the HPOM management server. This suppression can be controlled by a timer, a counter, or both. If duplicate message suppression is enabled on the management server, HPOM also keeps a counter of the suppressed messages. HPOM stores information about suppressed duplicate messages as annotations to the first message. In addition, duplicates are not passed to external notification or trouble ticket systems.

- **Smart Message Correlation**

  Often referred to as the "state-based" browser or "good/bad" message correlation because HPOM operators see only the current state of a monitored object. For example, if a CPU threshold is exceeded, and then later falls below the threshold, one message is generated indicating the violation. A subsequent message indicates a back-to-normal status, which automatically acknowledges the previous threshold-exceeded status. This smart message correlation automatically keeps HPOM operators up to date with the current health status of the CPU-monitored object for the managed node, and reduces the volume of messages within their responsibility.

Event Correlation

Event correlation allows real-time processing of event streams to identify relationships between events and, where possible, generate new and smaller streams with more useful and manageable information. In other words, event correlation uses the relationships between input events to produce a smaller output event stream containing richer information content.

To configure event correlation, HPOM provides the following tools:

- **Correlation Composer**

  Part of HPOM and HP NNM, which provide a GUI for configuring correlations.

- **ECS Designer**

  Separate product, which provides an even more sophisticated GUI-based tool for designing correlations.

- **HPOM for UNIX Developer's Toolkit**

  HPOM C/C++ APIs, which allow you to create correlations.

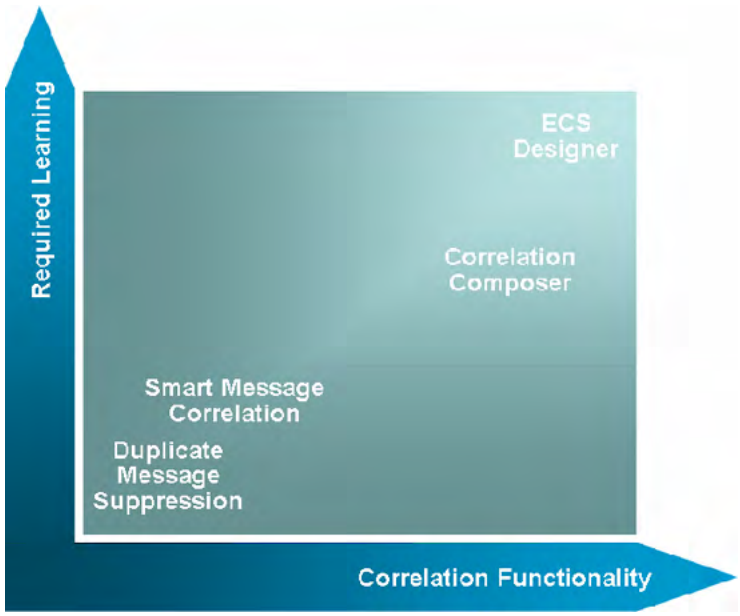
- **NNM Layer 2 monitoring**

  Includes additional correlation, based on network topology awareness. The correlated event stream is fed into HPOM, and can participate in any of the correlations mentioned above.

NOTE:
This document expands on the capabilities of Correlation Composer and ECS Designer. HP NNM is a tightly integrated component of HPOM, but details about its correlation capabilities are outside the scope of this document.

Figure 3 shows the relationship between functionality and ease of learning for HPOM message and event correlation.

Figure 3: Functionality versus ease of learning



Message Correlation Techniques

To correlate messages, HPOM provides duplicate message suppression on the managed node and the management server, as well as smart message correlation.

Duplicate Message Suppression on the Managed Node

Duplicate messages are those that report the same event or a similar event. HPOM lets you decide how often, or for how long, further messages are suppressed before a "new" duplicate message is sent. HPOM can suppress duplicate messages on the managed node or on the management server. Filtering (or suppressing) messages on the managed node reduces network traffic, and keeps the management server free for other tasks.

Duplicate suppression at the managed node is configured within HPOM policies. HPOM policies define what is to be monitored (for example, log files, processes, performance thresholds, SNMP traps, and so on). Within each HPOM policy, there are conditions that are compared with the monitored object to determine which action, if any, to take.

A simple example is an HPOM policy to monitor the `syslog` file. The policy contains a condition to match a file system full event and to report a formatted message to HPOM. It also includes an automatic action to include the output of a `df` command with the message.

Within an HPOM policy, there are two attributes of duplicate suppression:

- Definition of a duplicate message
- Type of suppression

Duplicate Messages

Duplicates can be one of the following:

- **Identical Input Events**

  Input events that are identical are considered to be duplicates. For example, log file entries that are exactly the same are treated as duplicates.

- **Identical Output Events**

  An output event is how you have defined the message to appear in the HPOM console. A message key summarizes the important characteristics of the event that triggered the message.

  For example, the policy condition for reporting a full file system may have a message key like this:

  `<$MSG_NODE_NAME>:<$MSG_OBJECT>:FS_Full`

  These variables are resolved at the time the condition is matched. (For example, the full file system event is detected by the HPOM policy.) The resulting output message that is created on the managed node would contain a message key `mynode.hp.com:/var:FS_Full`. This means that there could be a unique message key for each file system, ensuring that you track the duplicates for each distinct object (that is, file system) separately.

  If the message does not have a message key, HPOM determines a duplicate, based on seven attributes of the HPOM message: severity, node, application, message group, object, message text, and service name.

- **Messages Matching the Same Condition**

  Messages matching the same condition are considered to be duplicates.

  For example, if the policy condition matches all `xntpd<*>permission denied error` messages, all of the messages would be considered duplicates, despite the different time stamps and PIDs:

```
Jun 30 20:08:54 devs xntpd[896]: server returns a permission denied error
Jun 30 20:08:55 devs xntpd[896]: server returns a permission denied error
Jun 30 20:08:59 devs xntpd[1057]: server returns a permission denied error
```

  To suppress duplicate messages that are generated by different conditions, you enable this functionality on the management server.

Types of Suppression

You can specify three types of suppression:

- **Time Interval**

  Specify a time interval during which duplicate events are ignored, and a time period after which you want to start sending messages again.

- **Counter**

  Specify a threshold for a duplicate message counter. HPOM increments the counter until it equals or crosses the threshold. At that point, HPOM allows the transmission of the duplicate message.

- **Both**

  If you use the time interval and counter together, events are evaluated first by the timer. If an event passes the timer, it is then evaluated by the counter, which either suppresses it, or sends a message to the management server.

Duplicate Message Suppression on the Management Server

Suppressing duplicate messages on the management server significantly reduces high system loads caused by large numbers of messages. It also allows for correlating messages across multiple managed nodes.

The HPOM server determines duplicate messages on the same basis as the identical output message detection mentioned in "Duplicate Message Suppression on the Managed Node" on page 7. In other words, if the incoming message has a message key, HPOM checks the active message browser for an identical message key to determine if this is a duplicate message. If the incoming message does not have a message key, HPOM determines that the message is a duplicate only if an existing message contains identical values for the seven attributes of the HPOM message: severity, node, application, message group, object, message text, and service name.

HP recommends that you configure message keys within your policies as the basis for determining duplicates, for performance reasons. Many HPOM policies are provided with message keys already configured.

If an identical message already exists, HPOM suppresses the duplicate message and starts a counter of duplicates on the first message. The counter gives an indication of how often the problem occurred. The message also shows when the last duplicate message was received (and suppressed). If the message text, severity, or both of a duplicate message is different from that of the original, it is possible to have the latest information updated in the message.

Smart Message Correlation

At times, you may want to acknowledge messages automatically, such as in the following scenarios:

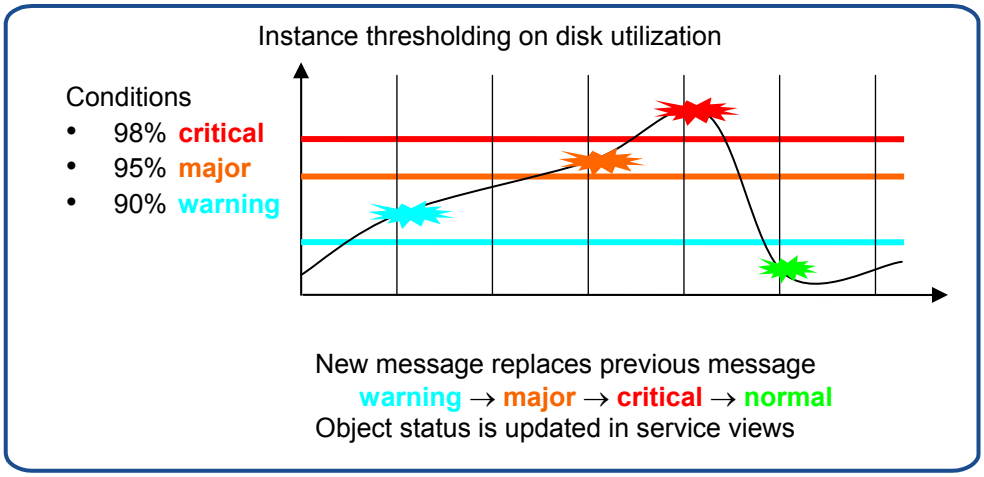
- **Problem Resolution**

  A first message might report a problem. Then, a second message might report that the problem has been solved (for example, if a server goes down and is restarted). Or the message might report that the problem has deteriorated. In either case, the first message would no longer be relevant. For this reason, you would want the second message to acknowledge the first message automatically. Also, you would want the second message to be cleared automatically. Both actions are achieved through smart message correlation.

- **Problem Escalation**

  HPOM might send a "Warning" severity message that the disk space on a managed node has exceeded its 90% threshold. When the disk space exceeds 95%, HPOM might send a second message with a severity of "Major." When the disk space exceeds 98%, HPOM might send a third message with a severity of "Critical." With smart message correlation, the second ("Major") message automatically acknowledges the first ("Warning") message, and the third ("Critical") message acknowledges the second ("Major") message, resulting in a less cluttered view for HPOM operators. In addition, operators save time because messages that have become irrelevant are acknowledged automatically. HPOM automates such scenarios, as shown in Figure 4.

Figure 4: Smart message correlation example with disk utilization threshold monitoring



When you acknowledge messages automatically, a maximum of one message per managed object is in the browser. This message reflects the current status of the object. In effect, the message browser has become a state-based browser.

When a message is acknowledged automatically by another message, both messages are annotated with information to identify the acknowledgement relationship between the messages.

When you work with related messages, the relationship between the first and the second (or the second and the third) message is established through message keys and message key relations. A message key relation acknowledges a message by matching the message key of that message. Because state-based correlation is particularly suited to threshold monitoring (for example, CPU, memory, and disk utilisation), HPOM can generate default message keys for these types of policies to minimize effort for the HPOM administrator.

HPOM also acknowledges the last message of a monitored object automatically by sending a reset message if a threshold was first exceeded, and then the monitored value drops below all possible reset values. The reset message acknowledges the last message sent for a given threshold monitor.

Event Correlation Techniques

Event correlation allows real-time processing of event streams to identify relationships between events, and then generate new and smaller streams with more useful and manageable information.

Real-time Processing of Event Streams

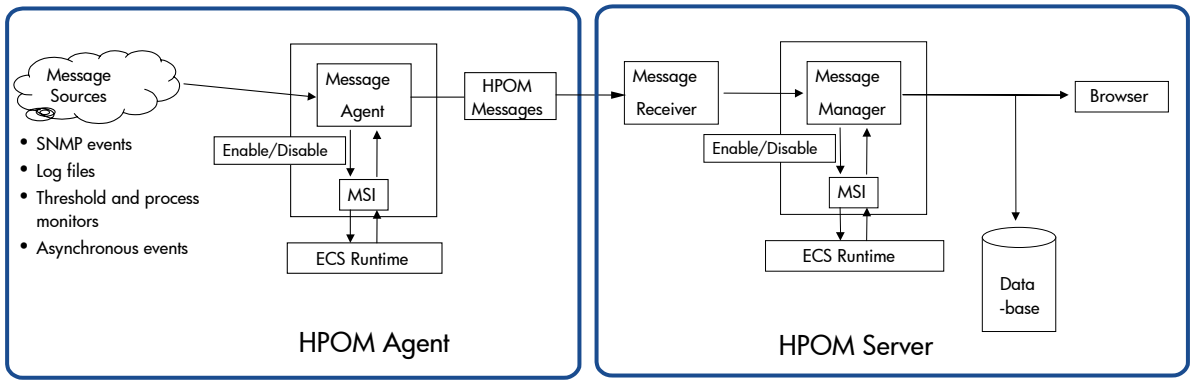
The correlations configured through Composer and ECS Designer are executed within an ECS run-time engine. This ECS run-time engine is part of the HPOM agent and the HPOM management server. As a result, correlations can take place on the local managed node and on the central management server. Local correlation reduces network traffic and minimizes overhead on the central management server. Correlation at the management server is useful for correlating events across different managed nodes and for enriching events with centrally provided additional information (for example, a centrally maintained data store for appending customer name or SLA information).

Both ECS Designer and Composer correlations, working from the HPOM event stream, can use external reference calls to history events and topology, as well as shared information model information, to derive more accurate and enriched information for HPOM operators.

Correlations are deployed from the central management server by using the GUI or command line.

The input events to the ECS run-time engine are messages that have been generated from monitoring through HPOM policies. The HPOM policies monitor for events of interest in log files, SNMP traps, threshold monitors, scheduled commands, and asynchronous events, as shown in Figure 5.

Figure 5: Event correlation message flow



Message Storm Detection: an Example

HPOM includes an ECS circuit and documentation for implementing message storm detection. ECS Designer is not required to use this circuit. The circuit is deployed to the management server. All messages that arrive at the management server are directed through this ECS circuit. For each node, the rate at which these messages arrive is measured with a moving interval. If the rate of messages received exceeds the allowed message rate, a configurable action is started. The default action calls a script that stops the node from causing a message storm.

Immediately after this action has been executed, a critical message is generated. This message informs you that node X caused a message storm, and that the configured action has been executed. In parallel, the messages received rate for this node is reset to zero. This reset allows the messages to pass through to the management server again. The reset is performed after a configurable delay that

allows the management server to process the pending requests. As soon as the circuit detects the first message coming from this node after a storm detection and reset, a message is generated to advise you that the message storm is over.

You can configure the circuit so that it does not send the messages that are received by the management server to the message browser until the message storm is stopped. If you decide to suppress them, you receive a message telling you how many messages have been suppressed, as well as a message informing you that the message storm is over.

Network Event Correlation

HP Network Node Manager (HP NNM) is the primary source of network events for HPOM.

HP NNM can correlate network events using the following methods:

- **Correlation Composer**

  User-defined correlation and out-of-the-box correlations:

  – *Cisco Chassis Failure*
  Monitors Cisco traps for three error conditions: temperature, fan failure, and power supply fault. Generates a new alarm if the condition persists for a specified time period.

  – *Multiple Reboots on Routers/Switches*
  Listens for coldStart and warmStart traps. Creates a new alarm when more than N coldStart or warmStart traps are received within M minutes from a specified SNMP agent and IP address pair.

  – *Router/Switch Intermittent Status Changes*
  Listens for OV_IF_Down alarms from routers and switches. Replaces them with a new alarm when more than N interface-down events are received within M minutes from a specific interface.

  – *Router/Switch Health*
  Group of three correlators to correlate interface status alarms with their related router or switch node status alarm. OV_IF_Unknown and OV_IF_Down status alarms from interfaces within routers or switches are suppressed and nested beneath the node status alarm.

- **Advanced Problem Analyser**

  Built into HP NNM Advanced Edition, which leverages HP NNM's topology for better root cause analysis.

- **ECS circuits**

  Four configurable built-in ECS circuits:

  – *Network Connector Down Correlation*
  Distinguishes between physical failures of network devices (primary failures) and failures that result from failure of a connector device (secondary failures).

  – *PairWise Events Correlation*
  Good-bad message correlation (for example, Node up and Node down).

  – *Repeated Events Correlation*
  Suppresses and groups events that are repeated logically within a given time window (for example, several "node add" events).

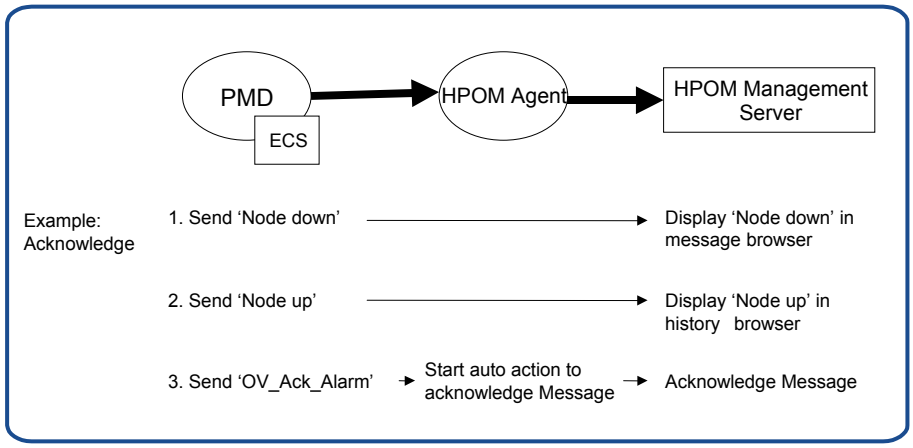
  – *Scheduled Maintenance Correlation*
  Suppresses events during periods when portions of the network are down for scheduled maintenance.

Each of these mechanisms produces a correlated event stream that, by default, is fed into HPOM for any additional message and event correlation. It is possible to forward the raw event stream to

HPOM, if needed. From HPOM, it is possible to inspect the correlated events from the HP NNM event database.

If an event is acknowledged or deleted in the HP NNM alarm browser, a new event is generated (OV_Ack_Alarm or OV_Delete_Alarm) containing the HP NNM UUID of the original event. HPOM includes a policy that uses this new event to acknowledge messages automatically as soon as the corresponding events are acknowledged in HP NNM. This process synchronises the network events from HP NNM with those presented into HPOM, as shown in Figure 6.

Figure 6. Example flow of HP NNM events to HPOM with automatic acknowledgement



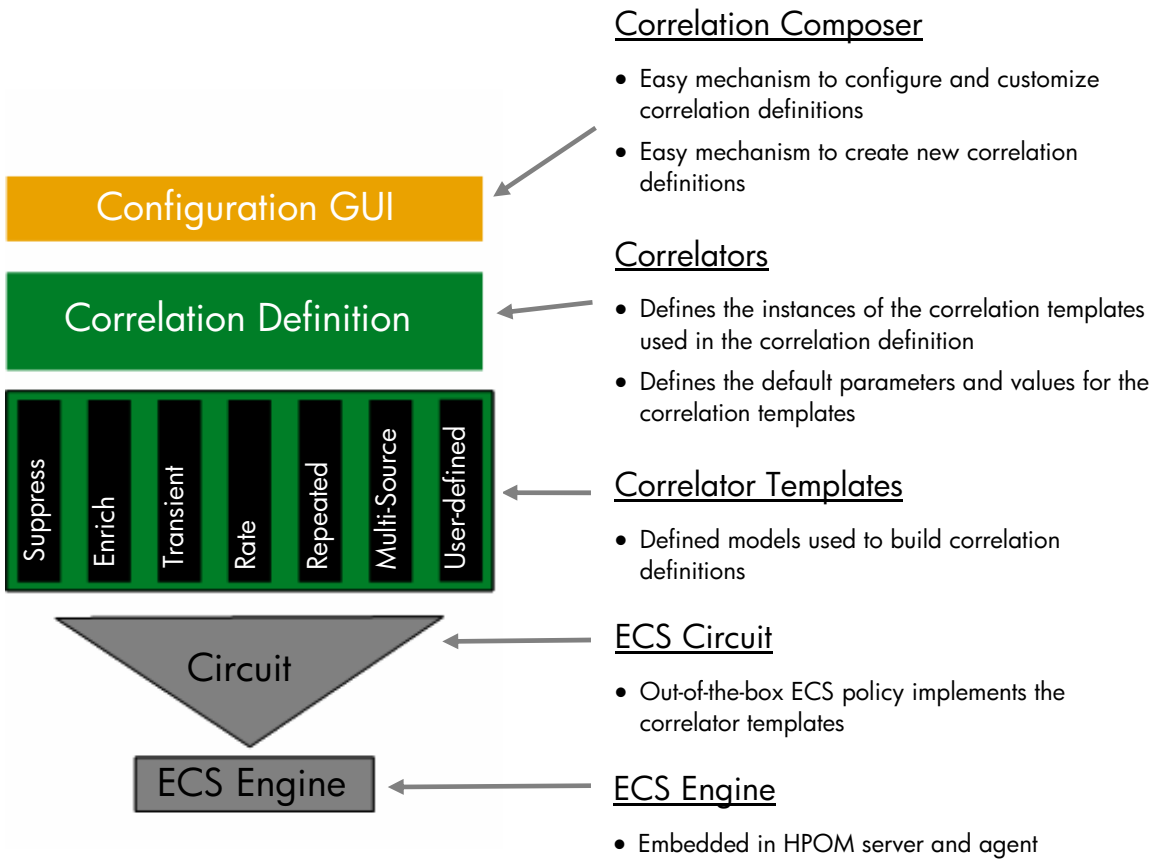
Flexible Management Correlation

In larger environments that make use of an HPOM flexible management configuration, message correlation can be seen in a much broader context, taking into account the relationships between the different levels in the management hierarchy. The relationship between managed nodes and the management server in the HPOM environment can be extended to the relationship between the management servers in the management hierarchy. Management servers can then correlate the messages they receive from the managed nodes, and send a newly correlated stream to the management servers to which they report. Or the management servers can send an uncorrelated stream of messages to be correlated on arrival at the next management server up the chain.

Correlation Composer

Correlation Composer provides easy access to ECS functionality without the complexity of the ECS Designer product. Correlation Composer is an integral part of HPOM. It consists of a GUI and ECS circuit that enable users to tailor the event correlation behavior for correlations that are shipped with HPOM products, as shown in Figure 7. Correlation Composer simplifies the development of user-developed correlations. You can use correlations out of the box. Or you can fine-tune them to fit your specific environment easily, without any programming knowledge.

Figure 7. HPOM Correlation Composer concept



The Composer ECS circuit is based on a "generic event flow" model. It comes prepackaged with six correlator templates: Enhance, Multi-Source, Rate, Repeated, Suppress, and Transient. If the prepackaged correlations do not meet your needs, you can implement a User-Defined correlator template in C or Perl. These templates are described in Table 1.

Table 1. Correlator templates

Correlator Template	Description
Enhance	Modify and enrich messages.
Multi-Source	Define a relationship between messages. Discard or modify a subset of messages.
Rate	Measure the number of messages occurring within a specified time period. Discard all messages. Emit only the newly created message.
Repeated	Discard duplicate messages received within a specified time period.
Suppress	Discard a specified category of messages.
Transient	Detect transient failures. On detection, discard both messages.
User-Defined	Define your own templates if none of the predefined templates meet your needs.

The GUI is the interface to specify the rules of correlation. The rules of correlation, in turn, are picked up by the circuit and realized. The semantics of each correlation model can be further fine-tuned by using controls in the Correlator Composer GUI.

Configuring Correlators

A correlator uniquely identifies a unit of correlation logic to be applied to an event or a set of events.

A correlator is configured in three parts:

- Alarm definition
- New alarm creation
- Callback functions

Note:
The terms "Alarm" and "Event" are used interchangeably. Correlator Composer often uses the term "Alarm."

Alarm definition

Alarm definition consists of alarm filters, variables, message keys, and parameters.

Alarm filters

An alarm is input to the correlator for processing if it passes through two levels of filtering. The primary filter is the Alarm Signature, where alarms whose attributes match the attribute specification in the Alarm Signature are processed further. The Alarm Signature is a set of tuples consisting of `Attribute` name, `Operator`, and `Value`.

For example, a failed `su` alarm signature might be just one tuple, such as this:

```
MSGTEXT matches "Bad switch user to <*> by <*>"
```

You can filter events in a correlator further, based on the Advanced Filter (secondary filter) condition. While the primary filter is limited to the attributes of the input alarm, the secondary filter is applied after additional processing or manipulation of the data that passes through the primary filter.

For example, suppose there is a requirement to generate a lower severity message if a database-related problem comes from a development server rather than from a production server. By simply

examining the event attributes, it is not possible to deduce whether the alarm is emitted from a development server. So you define a variable, called isDev, which is bound to the return value of a function GetIsDev. This function takes the hostname as its parameter. It returns 1 if the server is a development server. Otherwise, it returns 0. In the Advanced Filter, you define a correlator that checks if the variable, isDev, is set to 1, which ensures that the correlator is applied to development servers only.

Variables

Defining variables is an important part of filtering and refining alarms. You can define variables within correlators and as Global Constants. You can access all attributes in the alarms, including custom message attributes, as variables. After you assign variable names, you can use them in other sections of Correlator Composer. Variables can be any of the types listed in Table 2.

Table 2. Variable types

Type	Description
Constant	Value that you enter. This value is bound to the variable name.
Extract	Matched string assigned to a variable.
Function	Data returned from a function. You can use built-in functions or make external function calls (with Perl or C).
Combine	Combination of two or more variables.
Lookup	Data returned from a datastore lookup. Typically, the datastore is used to hold static topological information. (For example, you could use scripts that run once a day, at midnight, to create the datastore file, and update the ECS engine with the newly created file.)

Message keys

If two or more alarms need to be related, a message key is required. The message key identifies the instance of the correlator under which the alarm is correlated and evaluated for each incoming alarm that passes the primary and secondary filter. Alarms with identical message keys are correlated under the same instance of the correlator.

For example, consider a Multi-Source correlator that has two alarms defined: `good_su` and `bad_su`. The correlator is configured in such a way that the `good_su` alarm suppresses individual `bad_su` alarms. If a user performs a successful `su` soon after a failed `su`, the correlator discards the failed `su` because it was a user mistake rather than a potential security issue. However, the correlator is generic in the sense that it applies to all `good_su` and `bad_su` alarms. A `bad_su` alarm should be suppressed only if the user performs a `su` to the same target user. The mechanism that ties alarms together (in this case, the good `su` alarm to the bad `su` alarm) is the message key. The message key is a combination of hostname, the user performing the `su`, and the target user.

Parameters

If two or more alarms need to be related, there is a timeframe during which all alarms of the set need to arrive for the set to be considered complete. For example, a good `su` message must arrive within one minute of the corresponding bad `su` message for the set to be complete. If the good `su` arrives within one minute, the bad `su` is discarded. Otherwise, the bad `su` is not discarded.

There is also a threshold count. For example, with the Rate correlator, if the number of alarms exceeds this value within the specified timeframe, the rate threshold is considered breached, and you can choose between discarding the alarms or creating a new alarm.

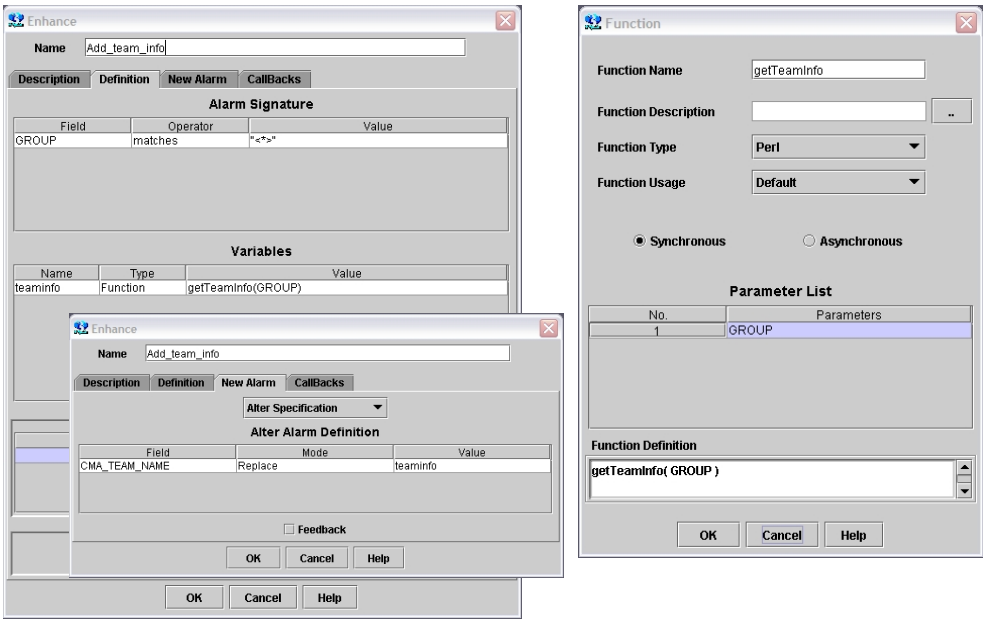
New alarm creation

If you want to generate an alarm, you can either create a new alarm, or modify an existing alarm. All HPOM message attributes and alarm definition variables are accessible.

Callback functions

A correlator can result in events being discarded or new events being created. You can invoke a user-defined callback function for either situation. This might be useful for creating an audit trail.

Figure 8: Enhance correlator enriching a message with external data retrieved from a user-defined Perl function



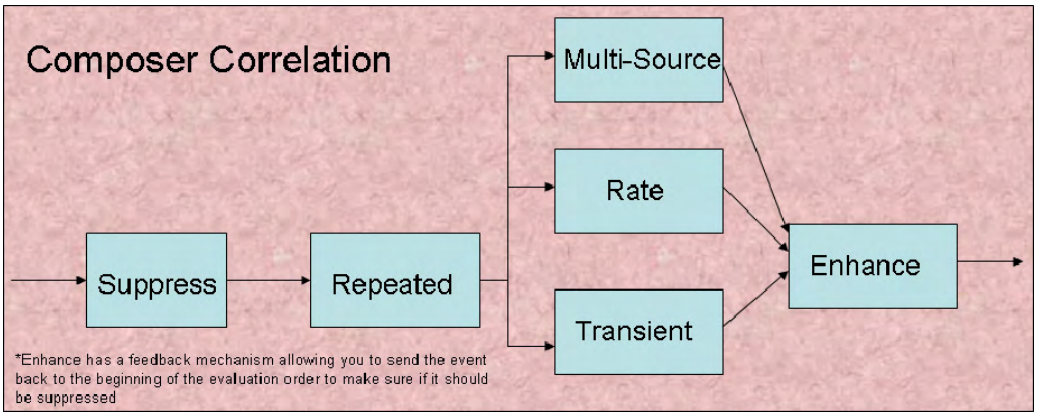
Correlator Template Evaluation Precedence

Each correlator is implemented by a discrete decision-making mechanism, based on the correlator template used. If the filters of two correlators are defined to admit the same alarm, both correlators are applied to the alarm.

If an event participates in multiple correlators, the following rules are applied to determine the outcome:

- Order of correlator evaluation is Suppress, Repeated, all other correlators (in parallel), and Enhance.
- If the Suppress and Repeated correlators discard the alarm, you can choose to allow the event to participate in other correlators before it is discarded.
- Enhance correlation is run last, enhancing the event if no other correlator discards it except when `Enhance Always` is enabled in the Enhance correlator template.
- Event is output *only if* no other correlator has discarded it.

Figure 9: Correlator template evaluation precedence



Composer Versus Message Correlation

The functionality of Correlation Composer and standard HPOM overlap. Table 3 provides example configurations and recommended approaches.

Table 3. Example configurations and recommended approaches

Correlation Type	Recommended
Filtering Pinpoint the object that has failed from a stream of related events from different sources.	Correlation Composer
Transient and repeated alarm correlation Alarms coming from the same managed object class and managed object instance ID, with the same probable cause and specific problem, but with different severity levels, are repeated within a specified time period.	Correlation Composer
Transient and repeated alarms Same alarms come from the same managed object class and managed object instance ID.	HPOM
Root cause and related alarm correlation Two alarms, A and B, are defined as root-cause and related alarms, respectively, if the fault that triggers A causes the fault that triggers B. B is recognized as an alarm related to A only if B is received within a specific time period after A was received.	Correlation Composer
User-defined threshold filtering alarm correlation Threshold filtering is another correlation process to reduce spurious alarms in the alarm processing modules. This type of correlation filters alarms through a user-defined threshold before reporting its presence. For example, an alarm (A) is not considered noteworthy unless it is repeated at least a specific number of times (seven) within a specific time period (20 seconds). So, alarm A is sent to the topology GUI only on its seventh occurrence within 20 seconds. If it does not occur seven times within 20 seconds, the alarm is not sent to the console. The concepts of fixed window and sliding window are applicable for these alarms.	HPOM (Correlation Composer has this functionality also)
Grade and quality of service alarm correlation If two alarms, A and B, are defined as critical by their definition, but A is from a source that has a defined higher grade of service in the customer Service Level Agreement (SLA), then A is graded higher than B for the operators (and ancillary Customer Care systems).	Correlation Composer
Impact analysis alarm correlation Impact analysis of alarms enables you to identify which services are or will be impacted by events. These events are generally referenced to a topology view of the End to End (E2E) environment to identify what is or will be impacted, and to what degree.	Correlation Composer
Thresholding Thresholding of the object status by using message key correlation.	HPOM

Out-of-the-Box Correlations

HPOM includes several Correlation Composer correlations as part of the OS SPI, as described in Table 4.

Table 4. Correlation Composer correlations

Name	Type	Description
osspi_good_ftp_bad_ftp	Multi-Source	When an FTP successful message is received within three minutes of FTP unsuccessful messages, all of the FTP unsuccessful messages are discarded.
osspi_good_log_bad_log	Multi-Source	When a logon successful message is received within three minutes of logon unsuccessful messages, all of the logon unsuccessful messages are discarded.
osspi_good_su_bad_su	Multi-Source	When a switch user successful message is received within three minutes of switch user unsuccessful messages, all of the switch user unsuccessful messages are discarded.
osspi_if_down	Multi-Source	Correlates the "Interface down" SNMP trap generated by HP NNM with the message that is generated by OS SPI instrumentation.
osspi_node_up	Multi-Source	Correlates messages from the SNMP trap template and the OS SPI log file encapsulators when a node comes up.
osspi_node_down	Multi-Source	Correlates messages from the SNMP trap template and the OS SPI log file encapsulators when a node goes down.
osspi_if_down_nfs_problems	Multi-Source	Correlates messages when "Interface down" messages are received for an interface from SNMP trap and OS SPI instrumentation, along with NFS problems. The interfaces are checked, and one combined message is sent.
osspi_high_cpu_swap_util	Multi-Source	Correlates messages from the OS SPI CPU monitor template and the SWAP monitor template, to give an indication. If thresholds set for both messages are critical, a thrashing kind of condition is indicated.

Additional example correlations, which show how you might configure your own environment-specific correlations, are described in Table 5.

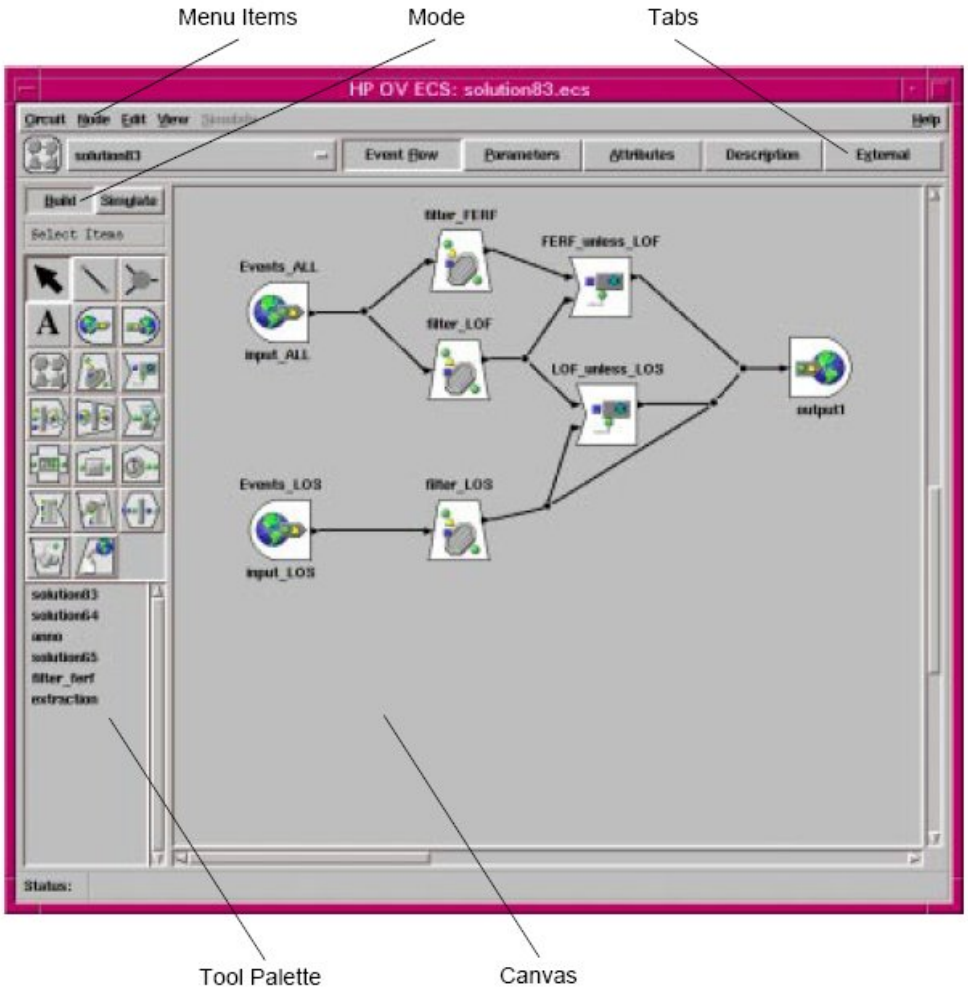
Table 5. Environment-specific correlations

Name	Type	Description
ora_disk	Multi-Source	Combine Oracle write failure with file system full message.
Add_host_info	Enhance	Add the info from `/etc/hosts` comment to message text.
Add_team_info	Enhance	Add the team name from `/opt/OV/contrib/ecs/external/perl/teams.txt` as a CMA. Team is selected by the HPOM message group.

ECS Designer

The ECS Designer GUI development environment is used to create and test correlation circuits. The correlation circuits are compiled and then deployed as HPOM policies to the run-time ECS engine on the central management server and the local managed node. A correlation circuit is created by selecting correlation nodes from a tool palette, placing them on the designer canvas, interconnecting the node ports, and configuring the node parameters, as shown in Figure 10.

Figure 10: ECS Designer GUI (Build Mode)



The correlation circuit may be saved, verified, and compiled. After they are verified, events may be processed through the correlation circuit using the ECS Designer in simulate mode, providing a visual indication of the event processing. Simulation uses a real correlation engine attached to the ECS Designer, but with the engine's notion of time controlled by the ECS Designer. The circuit designer has control over the flow of the simulation, including controlling the engine's time, stepping events from node to node, processing an event as far as currently possible, and setting break points on correlation nodes or on events. At any point, processing may be halted to allow the state of the correlation to be analyzed.

ECS correlation rules may use any data from the input HPOM messages to make correlation decisions. Data external to the ECS engine may be requested at any point within the correlation

decision process and used to modify or enhance the correlation. Data internal to the ECS engine may be output to HPOM at any point in the correlation process.

A correlation circuit is represented visually as a set of interconnected correlation nodes. Events enter the correlation circuit through Source nodes, flow through a series of correlation nodes arranged in any number of paths, and depart through Sink nodes. The correlation nodes control the disposition of events. Events may be discarded, combined, created, and so on, based on the correlation node types, the configuration of the nodes, and the dynamic conditions within the correlation circuit.

The correlation circuit paradigm results in a very intuitive implementation, using a visual design and debugging interface that allows the designer of the circuit to see which correlation operations depend on each other.

Nodes

The nodes you use to develop correlation circuits appear on the ECS Designer tool palette. These nodes can be grouped under five types, as described in Table 6.

Table 6. Nodes used to develop correlation circuits

Type	Description	Nodes	Description
External Connection Nodes	Connects the circuit to the outside world	Source	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.
		Sink	Forwards an event from the circuit or Compound node to the appropriate external event stream. You can have more than one Sink node in a circuit.
Event Traffic Cop Nodes	Controls the flow and structure of primitive events	Filter	Passes events that satisfy a user-defined condition. When the Filter node receives an event, it tests the event against the condition.
		Unless	Passes an event unless an inhibiting event occurs within a specified time frame.
		Combine	Groups events, and treats them as a single entity. It is typically used to group events that have been created almost simultaneously into a single composite event.
		Rearrange	Receives a composite event, and transmits a new event constructed from the events contained in the original composite event.
Time Control Nodes	Provides time control facilities	Clock	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	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	Measures the number of events per second passing through the node over a defined period 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.

Type	Description	Nodes	Description
		Delay	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 because of network delays.
Event Manipulation Nodes	Manipulates the information contained in each event	Table	Stores a history of extracted attributes of events or incoming events in event creation time order, and provides an interface that enables other nodes to query it.
		Extract	Creates and outputs composite events containing the received event and one or more other events copied from one or more Table nodes.
		Modify	Allows event attributes to be added, deleted, or changed.
		Create	Creates a new primitive event for each event that arrives at its input port.
		Annotate	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. This is how external data can be queried and incorporated.
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.		

Summary of Message and Event Correlation

Table 7 summarizes message and event correlation.

Table 7. Message and event correlation

Duplicate Message Suppression	Agent-based: ▪ By timer, counter, or a combination of both ▪ Built into HPOM policies Server-based: ▪ Message counter in browser
Smart Message Correlation	Built-in pair-wise correlation Defined through a message key and relations Persistence message handling Server-based
Composer	Supports six standard correlation scenarios and user-defined scenarios: ▪ *Enhance* Modify and enrich messages. ▪ *Multi-Source* Define a relationship between messages. Discard or modify a subset of messages. ▪ *Rate* Measure the number of messages occurring within a specified time period. ▪ *Repeated* Discard duplicate messages received within a timeframe. ▪ *Suppress* Used when a specific category of messages needs to be discarded. ▪ *Transient* Detect transient failures. On detection, discard both messages. ▪ *User-Defined* Extensible option. Much easier to use that ECS Designer Integrates external data into the correlation Extensible with Perl or C Agent and server correlations
ECS Designer	Very sophisticated Additional product Supports complex scenarios Agent and server correlations

For More Information

Information about the HPOM for UNIX Developer's Toolkit is available at the following location:

http://devresource.hp.com

August 2008