HP Business Service Management

For the Windows® and Linux operating systems

Software Version: 9.00

HP BSM Integration Adapter User Guide

Document Release Date: June 2010

Software Release Date: June 2010

Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notices

Trademark Notices

Adobe® and Acrobat® are trademarks of Adobe Systems Incorporated.

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

UNIX® is a registered trademark of The Open Group.

Acknowledgements

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com).

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/).

This product includes software written by Tim Hudson (tjh@cryptsoft.com).

This product includes an interface of the 'zlib' general purpose compression library, which is Copyright © 1995-2002 Jean-loup Gailly and Mark Adler.

Support

Visit the HP Software Support Online web site at:

http://www.hp.com/go/hpsoftwaresupport

This web site provides contact information and details about the products, services, and support that HP Software offers.

HP Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support web site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract. To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

To find more information about access levels, go to:
http://h20230.www2.hp.com/new_access_levels.jsp

Table of Contents

HP BSM Integration Adapter

BSM Integration Adapter enables you to monitor event sources, and, if certain conditions apply, to forward the detected events as HP Business Service Management (BSM) events directly to the BSM Operations Management event browser. For BSM Integration Adapter to be able to convert the source events to BSM events, the event sources must make their data available as SNMP traps or in XML-formatted files.

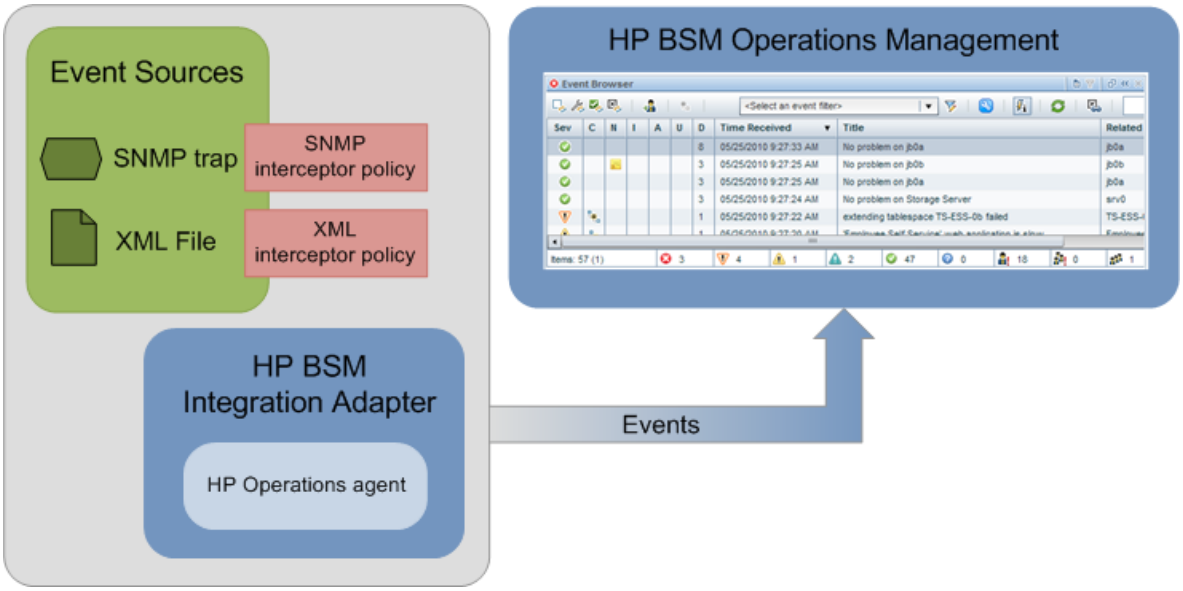
SNMP event sources can range from simple hardware devices that send SNMP traps to sophisticated network management solutions such as HP Network Node Manager i (NNMi). (NNMi provides an out-of-the-box integration with BSM Integration Adapter, which converts NNMi incidents to SNMPv2c traps and forwards these SNMPv2c traps as BSM events to the BSM Operations Management event browser. For more information about this integration, see the *NNMi Deployment Guide*.)

BSM Integration Adapter can also monitor XML-formatted files. Because BSM Integration Adapter is agnostic to the content and syntax of the XML files, you can monitor any XML file. (If the application that you want to monitor does not log its events in XML format, consider writing a program or script that diverts and converts the application's output to XML-formatted files.)

Note: BSM provides EMS (Enterprise Management Systems software) to facilitate the integration of data from other applications, both HP and third party. While BSM Integration Adapter is the recommended solution for integrating events into BSM, HP recommends that you use EMS to integrate other types of data, for example, metric data. See the *BSM Solutions and Integrations* guide for more information.

In the context of HP BSM Operations Manager i (OMi), the BSM Operations Management event browser can receive events from HP Operations Manager (HPOM), or directly from BSM Integration Adapter. BSM Integration Adapter is the preferred method for sending events to OMi for new event integrations, if there is no need to use the HPOM event processing capabilities or console, or if HPOM is not part of the BSM deployment. For more information about deploying BSM, see the *BSM Deployment Guide*.

BSM Integration Adapter includes the HP Operations agent. The HP Operations agent collects and monitors the data on the event source, enriches these events with information that is meaningful to BSM users, and sends the events to BSM where they display in the Operations Management event browser. BSM Integration Adapter uses policies to configure the agent. For each policy, you decide what kinds of events to monitor, how often to monitor, what to look for in the events, and what to do if certain events are detected.



HP Operations agents and BSM communicate using HTTPS, which is secure, reliable, and simplifies firewall configuration.

- Secure communication based on the HTTPS protocol. All communications between BSM and BSM Integration Adapter are strongly encrypted.
- Policies and events all contain signatures, which BSM servers and BSM Integration Adapter servers create and check using certificates. If a malicious user attempts to tamper with a signed policy or event, the signature becomes invalid.
- Simplified firewall configuration. BSM and BSM Integration Adapter accept all inbound communications to a single port, so it is simpler to configure firewalls and proxies.

You typically install BSM Integration Adapter on the computer that provides the input events. For example, to integrate NNMi events into BSM, install BSM Integration Adapter on the NNMi management server. The BSM Integration Adapter user interface is web based; you can therefore access it from anywhere using a supported web browser.

Managing policies

Policies are collections of configuration information used to configure the HP Operations agent on the BSM Integration Adapter server to perform monitoring tasks. When you develop monitor policies, you decide what kinds of events to monitor, how often to monitor, what to look for in the events, and what to do if certain events are detected.

Rules define what a policy should do in response to a specific type of event. Each rule consists of a condition and of settings for the event generated by the policy. The condition is the part of a policy that describes the type of event in the source. The settings enable you to configure the event that BSM Integration Adapter sends to the Operations Management event browser.

A policy must contain at least one rule. If the policy contains multiple rules, it is important to remember that the rules are evaluated in a specific order, and that when one condition is matched, no additional rules will be evaluated.

The rule types are:

- **Event on matched condition**

  If matched, BSM Integration Adapter sends an event to the Operations Management event browser. The event uses the settings defined for the rule. If you do not configure these settings, the default settings are used.

- **Suppress on matched condition**

  If matched, BSM Integration Adapter stops processing and does not send an event to the Operations Management event browser.

- **Suppress on unmatched condition**

  If not matched, BSM Integration Adapter stops processing and does not send an event to the Operations Management event browser.

Note: BSM Integration Adapter enables you to create and edit policies of the type SNMP interceptor and XML interceptor only. You can import policies developed on other servers, for example, HP Operations Manager (HPOM) management servers, HP Network Node Manager i (NNMi) management servers, or other BSM Integration Adapter servers.

When you create a new policy or import a policy, the policy exists in the BSM Integration Adapter policy repository but does not function yet. You must first activate the policy for it to start monitoring the corresponding event source.

Note: BSM tries to associate the events that it receives with a configuration item (CI) using CI resolution. This is only possible if the computer on which the event occurred is set up as a CI in BSM. It is therefore recommended that you set up all computers that may generate events as CIs in BSM.

Activate and deactivate policies

When you create a new policy or import a policy, the policy exists in the BSM Integration Adapter policy repository but does not function yet. You must first activate the policy for it to start monitoring the corresponding event source.

When you edit an existing, active policy, the previous version of the policy remains active on the BSM Integration Adapter server and you must reactivate the policy for your changes to take effect.

When you deactivate a policy, the policy remains installed on the system but does not function until it is activated.

To activate policies

1. In the list of policies in the BSM Integration Adapter user interface, select one or more policies. The activation state of the policies must be `deactivated` or `activated (reactivate for new version)`.
2. Click  in the toolbar. The activation state changes to `activated`.

To deactivate policies

1. In the list of policies in the BSM Integration Adapter user interface, select one or more policies.
2. Click  in the toolbar. The activation state changes to `deactivated`.

Import policies

BSM Integration Adapter enables you to create and edit policies of the type SNMP interceptor and XML interceptor only.

You can import policies developed on other servers, for example, HP Operations Manager (HPOM) management servers, HP Network Node Manager i (NNMi) management servers, or other BSM Integration Adapter servers.

When you download policies on an HPOM management server, make sure the resulting policy files support the XML-based policy exchange format:

- HP Operations Manager for Windows: use the `ovpmutil` command line utility.
- HP Operations Manager for UNIX or Linux: use the `opccfgdwn` command line tool.

NNMi provides the nnmopcexport.ovpl command line tool, which exports an SNMP interceptor policy for use with BSM Integration Adapter. For more information about running this tool and the NNMi integration in general, see the *NNMi Deployment Reference*.

You can also import policies developed on other BSM Integration Adapter systems, for example, to ensure that the same set of policies is available on multiple BSM Integration Adapter systems. This is necessary, for example, when running BSM Integration Adapter in a cluster environment.

BSM Integration Adapter stores policies in the following folders:

- Windows: `%OvDataDir%\datafiles\policymanagement\store`
- Linux: `/var/opt/OV/datafiles/policymanagement/store`

To import policies

1. In the BSM Integration Adapter user interface, click  in the toolbar. A file selection dialog box opens.
2. Navigate to the policy files and, for each policy, select both the header (`*_header.xml`) and the data (`*_data`) files. You can import up to 100 policies at once (that is, 200 policy files).
3. Click **Open** to start the import process.

If the same policies already exist in BSM Integration Adapter, you are asked whether you would like to replace them with the newly imported policies.

The imported policies appear in the list of policies in the BSM Integration Adapter user interface. They are by default deactivated.

Delete policies

You can only delete deactivated policies from BSM Integration Adapter. When you delete a policy the policy files are deleted from the file system. To temporarily stop a policy from functioning, deactivate the policy instead.

To delete policies

1. In the list of policies in the BSM Integration Adapter user interface, select the policy that you want to delete.
2. Make sure that the policy is deactivated. If necessary, click  in the toolbar to deactivate the policy.
3. Click  in the toolbar. A message box opens.
4. Confirm that you want to delete the policy.

Developing SNMP interceptor policies

This policy type monitors SNMP events, and responds when a character pattern that you choose is found in an SNMP trap. Choose this policy type if you want to monitor network components that send SNMP traps.

Tip: When you install BSM Integration Adapter on an HP Network Node Manager i (NNMi) management server, you must enable the HP Operations agent integration with NNMi so that the HP Operations agent can receive SNMP traps from the NNMi northbound interface. To enable the NNMi integration, follow the procedures in the *NNMi Deployment Reference* instead of the ones below..

To configure the HP Operations agent to receive SNMP traps

BSM Integration Adapter on Windows

- SNMPv1 traps only

  Start the Microsoft SNMP Trap service.

- SNMPv1 and SNMPv2 traps

  a. Stop the Microsoft SNMP Trap service. Set service start mode to Manual to prevent it from getting started at next system boot.

  b. Configure the agent to use the embedded NNMi SNMP libraries. Open a command prompt, and then type:

  ```
  ovconfchg -ns eaagt -set SNMP_SESSION_MODE NNM_LIBS
  ```

  c. Restart the trap interceptor. Open a command prompt, and then type:

  ```
  ovc -restart opctrapi
  ```

BSM Integration Adapter on Linux

- SNMPv1 and SNMPv2 traps

  a. Configure the agent not to use the embedded NNMi SNMP libraries but to access the default SNMP port directly (162 by default). Open a command prompt and type:

  ```
  ovconfchg -ns eaagt -set SNMP_SESSION_MODE NO_TRAPD
  ```

  b. Restart the trap interceptor. Open a command prompt, and then type:

  ```
  ovc -restart opctrapi
  ```

To configure an SNMP interceptor policy

1. In the BSM Integration Adapter user interface, click * in the toolbar, then click **SNMP**. The SNMP interceptor policy editor opens.

   Alternatively, double-click an existing SNMP interceptor policy to edit it.

2. Complete the information in the following tabs:

- **Properties** include information that is related to the policy itself (for example, the name and description of the policy).
- The policy **defaults** include default settings for all events generated by the policy (for example, default event attributes).
- **Rules** define what the policy should do in response to a specific type of event.
- **Options** configure several policy behaviors (for example, pattern matching options).

3. Click  in the toolbar to save the policy.
4. In the BSM Integration Adapter user interface, click  in the toolbar to update the list of policies.

Configure SNMP interceptor policy properties

Every policy has a set of properties that identify and describe the policy.

To configure properties of SNMP interceptor policies

1. In the BSM Integration Adapter user interface, click  in the toolbar, then click **SNMP**. The SNMP interceptor policy editor opens.

   Alternatively, double-click an existing SNMP interceptor policy to edit it.
2. Click **Properties**.
3. *Required.* In the **Name** box, type a name that will identify the policy. You can use spaces in policy names. The equal sign (=) is not allowed.
4. *Optional.* In the **Description** box, type a description of what the policy does. You might also add other notes, for example data sources that are used.
5. *Optional.* In the **Category** box, type one or more arbitrary categories. Policy categories may help you to better group your policies. Separate multiple categories with commas.
6. **Policy ID**: BSM Integration Adapter automatically assigns a GUID to the policy when it is first created.
7. **Last modification**: The date that the policy was saved.
8. **Last modified by**: The name of the user active when the policy was saved.
9. Click  in the toolbar to save the policy.
10. In the BSM Integration Adapter user interface, click  in the toolbar to update the list of policies.

Configure event defaults in SNMP interceptor policies

The Defaults tab enables you to indicate default settings for all events generated by the policy.

These defaults affect all new and existing rules. You can override the defaults in individual rules if needed. If a rule contains empty event attributes, the agent will use the defaults for the new event.

To configure attribute defaults in SNMP interceptor policies

1. In the BSM Integration Adapter user interface, click * in the toolbar, then click **SNMP**. The SNMP interceptor policy editor opens.

   Alternatively, double-click an existing SNMP interceptor policy to edit it.

2. Click **Defaults - Event**. The default settings include:

   - Configure event attributes

     The event attributes tab enables you to set the event attributes for a specific event (or for the event defaults). These attributes are visible in the Operations Management event browser and help the user to organize and evaluate the events.

     - **Severity**: Severity assigned to the event (Critical, Major, Minor, Warning, Normal, Unknown).
     - **Category**: Name of the logical group to which the event belongs (for example, Database, Security, or Network). The event category is similar in concept to the HP Operations Manager message group.

   - Configure event correlation

     Event correlation helps to prevent the Operations Management event browser from becoming cluttered by events that describe the same problem. When event correlation is enabled, you can set the type of duplicate event suppression and define the method used to suppress duplicate events.

     - **Event Key**: An identifier used to identify duplicates and for Close Events with Key.
     - **Close Events with Key**: If events with the event key that you type here exist in the Operations Management event browser when this event is received, these events are automatically closed. You can use pattern matching and variables to match multiple event keys. For example, consider the following pattern:

       ```
       <$MSG_SEV>:<$MSG_NODE_NAME>:<_><5*>
       ```

       This pattern is evaluated by first replacing the variables with the values that they resolve to, for example:

       ```
       critical:cabbage.example.com:<_><5*>
       ```

       This pattern is then compared using pattern matching rule against the event keys for all events in the Operations Management event browser. The pattern above would match the following event keys:

       ```
       critical:cabbage.example.com:  12345
       critical:cabbage.example.com:  TEST1
       ```

       When writing patterns for this box, note the following:

       - Although user-defined variables can be included in this box, these variables can only be expanded after the policy is deployed and therefore are not included in the syntax check that is performed when the policy is saved. Because of this, it is important to ensure that any user-defined variables in this box are correctly used.

     - **Deduplication on Server**: Clear to disable deduplication on the server. Stops automatic closing of new events that are duplicates of existing events.

     Suppress events which are:

- **Generated by same rule**: Select this option to suppress events that match the pattern specified for the selected rule. This is a more general setting for the suppression of duplicate events. For example, an XML log file entry policy might contain a rule with this match pattern: `Error Message<#>` The log file lines `Error Message10` and Error `Message20` are not identical, but would both match this rule.
- **Generated by the same input event**: Select this option to suppress events that were sent in response to two separate input events that are identical except for the date and time that the event was generated (for example, identical entries in an XML log file).
- **Identical relative to their attributes**: Select this option to suppress either events that have the same event key or (if no event key is present) events that have identical event attributes (except for the date and time that the event was generated).

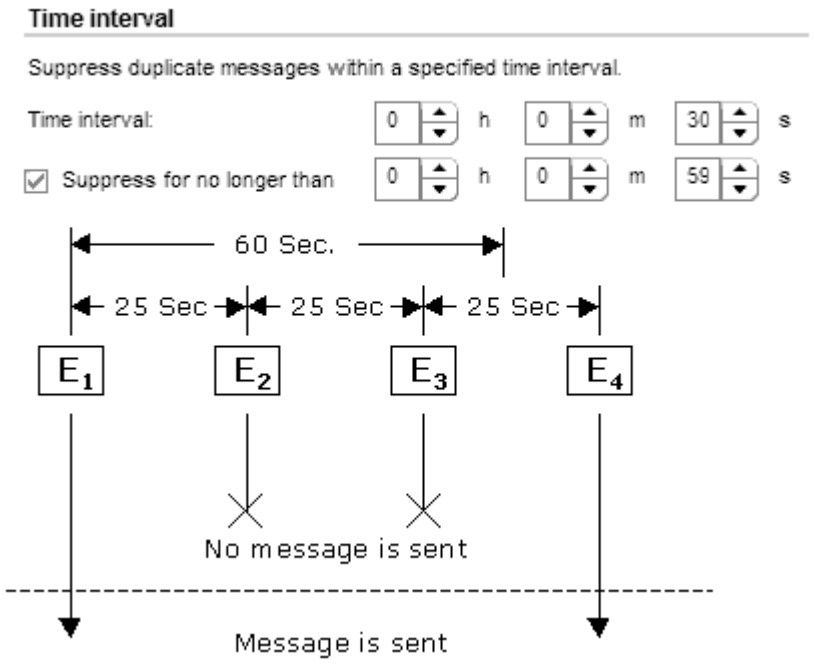
Suppression Method

For event correlation, you can define one of three correlation methods:

- **Time interval:** This correlation method lets you define an interval during which duplicate events will be ignored. For more information, read this detailed example.

Time interval correlation example

In the illustration below, the interval is set to 30 seconds, but the suppression is limited to 60 seconds.

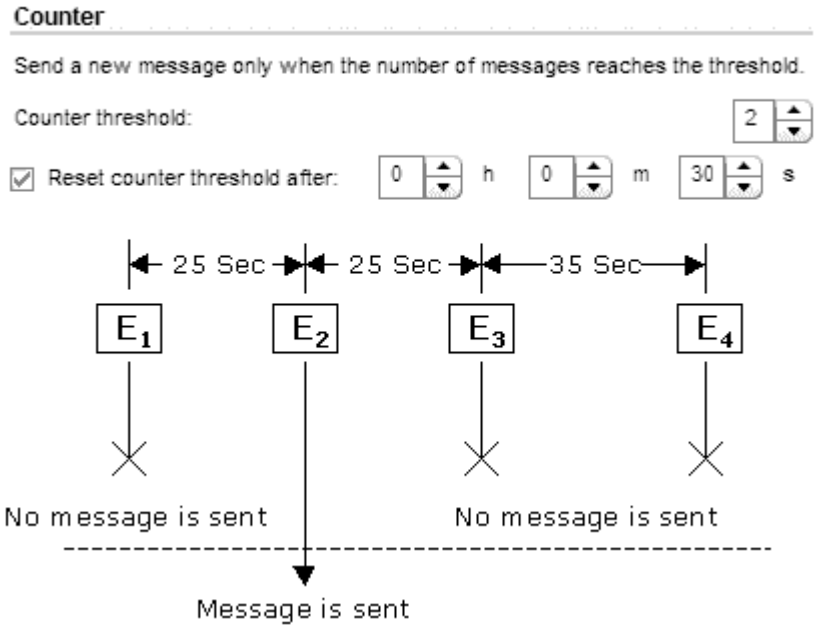


The E_x represents events that are identical.

- The first input event (E1) matches a rule in the policy. The policy sends an event and starts timing.
- A second matching event (E2) occurs 25 seconds later. This event occurred *less than 30* seconds after the first event, and is therefore suppressed.
- A third matching event (E3) occurs *less than 30 seconds after the second event*, and so is also suppressed.
- The next matching event (E4) occurs less than thirty seconds after the third event, but is also *more than 60 seconds after the first event*, and therefore the policy sends an event.

- **Counter:** This correlation method counts the number of matching input events and sends an event only after the number of matching input events equals the counter threshold. The counter can also be reset to zero after a time period that you specify. For more information, read this detailed example.

Counter correlation example



The E_x represent events that are identical.

- The first input event (E1) matches a rule in the policy, and the counter increments to one. No event is sent.
- A second matching event (E2) occurs, the counter increments to two, an event is sent, and the counter resets.
- A third matching event (E3), and the counter increments to one no event is sent.

- The next matching event (E4) occurs *more than thirty seconds* after the third event. Since at thirty seconds the counter was reset to zero, the counter now increments to one no event is sent.

- **Time interval/Counter** 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 an event to the Operations Management event browser.

Note: If you specify just time interval correlation or just counter-based correlation in an individual event, any eventdefaults for the other correlation method also apply. For example, if you specify time interval correlation for an event, and the event defaults specify counter-based correlation, the combined time interval and counter-based correlation applies to both new rules and existing rules.

You can change this default behavior, so that only the correlation method that you specify in the individual event applies. To change the default behavior, set the parameter `OPC_IGNORE_DEFAULT_MSG_CORRELATION=TRUE` in the `eaagt` namespace on the node. You can configure this parameter using `ovconfchg` or `ovconfpar` at a command prompt.

- Configure the message stream interface

  This tab lets you configure the interface between messages and external programs on the HP Operations agent.

  The message stream interface allows external applications to interact with the internal message flow of the HP Operations agent. The external application can be a read-write application, for example, a message processing program that can read HP Operations messages, modify attributes, and generate new messages for retransmission to the server. The application could also read messages, or send its own messages.

  When you enable the message stream interface, you can also allow external applications using the interface to set up automatic or operator-initiated commands.

  Select **Agent message stream interface** to allow messages to be directed to the **message stream interface** on the node. When switched on, you can choose between the following options:

  - Divert a message to the message stream interface instead of to the server when a message is requested by an external application.
  - Send the message to the server, and a copy of the message to the message stream interface.

- Configure OM attributes

  The OM attributes tab enables you to set additional attributes for a specific event (of for the event defaults). These attributes are visible in the Additional Information tab of the Operations Management event browser and help the user to organize and evaluate the events.

  - **Application**: Application that caused the event to occur. Unlike the Related CI attribute, which is a direct relationship to a CI in the ODB, the application attribute is a simple string-type attribute (for example, Oracle and OS).
  - **Object**: Device such as a computer, printer, or modem. Unlike the Related CI attribute, which is a direct relationship to a CI in the ODB, the application attribute is a simple string-type attribute (for example, C:, and /dev/spool).
  - **HPOM Service ID**: ID of the service associated the event. A service ID is a unique identifier for a service and can be used in BSM to identify the node and CI associated with the event.

- Add policy variables

You can use policy variables in event attributes. BSM Integration Adapter replaces the variables with the appropriate values in the generated event.

It is often useful to surround the variable with quotation marks, especially if it may return a value that contains spaces.

`<$#>`
Returns the number of variables in an enterprise-specific SNMP event (generic event 6 Enterprise specific ID). Sample output: `2`

`<$*>`
Returns all variables assigned to the event up to the possible fifteen. Sample output: `[1] .1.1 (OctetString): arg1 [2] .1.2 (OctetString): turnip.example.com`

`<$@>`
Returns the time the event was received as the number of seconds since Jan 1, 1970 using the *time_t* representation. Sample output: `859479898`

`<$1>`
Returns one or more of the fifteen possible event parameters that are part of an SNMP event. (<$1> returns the first variable, <$2> returns the second variable, and so on.)

`<$\>1>`
Returns all attributes greater than *n* as *value* strings, useful for printing a variable number of arguments. `<$\>0>` is equivalent to `$*` without sequence numbers, names, or types. Sample output: `bokchoy.example.com`

`<$\>+1>`
Returns all attributes greater than *n* as *name:value* string. Sample output: `.1.2: asparagus.example.com`

`<$+2>`
Returns the nth variable binding as *name:value* . Sample output: `.1.2: artichoke.example.com`

`<$\>-n >`
Returns all attributes greater than *n* as *[seq] name (type): value* strings. Sample output: `[2] .1.2 (OctetString): cauliflower.example.com`

`<$-2>`
Returns the nth variable binding as *[seq] name-type:value* . Sample output: `[2] .1.2 (OctetString): brusselsprouts.example.com`

`<$A>`
Returns the node that produced the event. Sample output: `eggplant.example.com`

`<$C>`
Returns the community of the event. Sample output: `public`

`<$c>`
Returns the event's category. Sample output: `SNMP`

`<$E>`
Returns the enterprise ID of the event. Sample output: `.1.3.6.1.4.1.11.2.17.1`

`<$e>`
Returns the enterprise object ID. Sample output: `.1.3.6.1.4.1.11.2.17.1`

`<$F>`
Returns the textual name of the remote postmaster daemon's computer if the event was forwarded. Sample output: `cress.example.com`

`<$G>`
Returns the generic event ID. Sample output: `6`

`<$MSG_NODE>`
Returns the IP address of the node on which the original event took place. Sample output: `192.168.1.123`

`<$MSG_NODE_NAME>`
Returns the name of the node on which the original event took place. This is the hostname that the agent resolves for the node. This variable is not fixed, however, and can be changed by a policy on a per-message basis. For example, if the policy is intercepting SNMP traps that originate from other devices, you might want to set this variable to the name of the device where the trap originated. If the policy is monitoring a log file on a network share where applications on several nodes write messages, you could extract the name of the node from the error message, save it in a user-defined variable, and assign it to MSG_NODE_NAME.

`<$MSG_OBJECT>`
Returns the name of the object associated with the event. This is set in the Message Defaults section of the policy editor.

`<$MSG_TEXT>`
Returns the full text of the event. In general, there are default texts for all editors derived from incoming event properties. Sample output: `SU 03/19 16:13 + ttyp7 bill-root`

`<$N>`
Returns the event name (textual alias) of the event format specification used to format the event, as defined in the Event Configurator. Sample output: `OV_Node_Down`

`<$O>`
Returns the name (object identifier) of the event. Sample output: `.1.3.6.1.4.1.11.2.17.1.0.58916865`

`<$o>`
Returns the numeric object identifier of the event. Sample output: `.1.3.6.1.4.1.11.2.17.1.0.58916865`

`<$R>`
Returns the true source of the event. This value is inferred through the transport mechanism which delivered the event. Sample output: `carrot.example.com`

`<$r>`
Returns the implied source of the event. This may not be the true source of the event if the true source is proxying for another source, such as when a monitoring application running locally is reporting information about a remote node. Sample output: `rutabaga.example.com`

`<$S>`
Returns the specific event ID. Sample output: `5891686`

`<$s>`
Returns the event's severity. Sample output: `Normal`

`<$T>`
Returns the event time stamp. Sample output: `0`

`<$V>`
Returns the event type, based on the transport from which the event was received. Currently supported types are SNMPv1, SNMPv2, CMIP, GENERIC, and SNMPv2INFORM. Sample output: `SNMPv1`

`<$X>`
Returns the time the event was received using the local time representation. Sample output: `17:24:58`

`<$x>`

Returns the date the event was received using the local date representation. Sample output: `03/27/10`

3. Click  in the toolbar to save the policy.
4. In the BSM Integration Adapter user interface, click  in the toolbar to update the list of policies.

Configure SNMP rules

Rules define what a policy should do in response to a specific type of event. Each rule consists of a condition and of settings for the event generated by the policy. The condition is the part of a policy that describes the type of event in the source. The settings enable you to configure the event that BSM Integration Adapter sends to the Operations Management event browser.

A policy must contain at least one rule. If the policy contains multiple rules, it is important to remember that the rules are evaluated in a specific order, and that when one condition is matched, no additional rules will be evaluated.

The rule types are:

- **Event on matched condition**

  If matched, BSM Integration Adapter sends an event to the Operations Management event browser. The event uses the settings defined for the rule. If you do not configure these settings, the default settings are used.

- **Suppress on matched condition**

  If matched, BSM Integration Adapter stops processing and does not send an event to the Operations Management event browser.

- **Suppress on unmatched condition**

  If not matched, BSM Integration Adapter stops processing and does not send an event to the Operations Management event browser.

Note: In all cases, if a rule evaluates as true, no more rules are processed. It is important to pay attention to the rule order.

The order in which rules are evaluated has a large effect on the type of messages you receive. It also affects the speed with which messages are sent and the amount of processor time that is required by the policy.

For example, you might have a policy that monitors CPU activity, containing these two rules:

1. `If usage is greater than 80%,`
   `send a warning message and stop processing rules.`
2. `If usage is greater than 95%,`
   `send a critical message and stop processing rules.`

If the rules were evaluated in the order shown, disk usage of 99% would only produce a warning message. If the order were reversed, however, a critical message would be sent. You could solve the problem by making the rules more specific, so that the order was not important:

1. `If usage is between 80% and 94%,`
   `send a warning message and stop processing rules.`

2. `If usage is greater than 95%,`
   `send a critical message and stop processing rules.`

In the example above, disk usage of 99% produces a critical message regardless of which rule is evaluated first. However, if the rules are evaluated in the order shown, disk usage of 99% is evaluated by two rules. If the order were reversed, it would be evaluated only by the first rule, thereby sending the message more quickly and reducing processing time on the managed node.

To configure rules in SNMP interceptor policies

1. In the BSM Integration Adapter user interface, click * in the toolbar, then click **SNMP**. The SNMP interceptor policy editor opens.

   Alternatively, double-click an existing SNMP interceptor policy to edit it.

2. Click **Rules**.

3. Click * in the toolbar and select the rule type. Then type the name of the rule. After a rule has been added, you can change the rule type by clicking the current rule type in the list of rules and selecting another rule type from the drop-down list.

   Alternatively, select an existing rule and click to copy the rule. You can then rename the copied rule and edit it.

4. Configure the following settings for the rule:

   - Configure the condition

     With this tab, you set the match conditions for an SNMP interceptor rule.

     - **Node**: If you only want to match SNMP events from a specific node, type the FQDN, the primary node name, or the IP address. Give multiple entries with the **OR** operator (for example, `celery.veg.com|broccoli.veg.com`), or leave blank for all nodes.
     - **Event Information**

       If you want to match a specific event, select **Event Object ID**. If you want to match a range of events or specify a specific event in **SNMPv1 Notation**, select the **SNMPv1 Notation** option.

       - **Event Object ID**

         Type the complete Event Object Identifier for the SNMP event that you want to match.

         For example: .1.3.6.1.4.1.11.2.17.1.0.40000001

       - **SNMPv1 notation**

         You can type the complete event object ID in SNMPv1 format or you can specify only part of the identifier. For example, by specifying only the Enterprise ID, you can match all events with a specific Enterprise ID.

       - **Enterprise ID**

         Type in the enterprise ID for incoming SNMP traps to be compared with this condition. The enterprise ID is a vendor-specific identifier for the trap. Standard BSM Integration Adapter pattern-matching syntax may not be used in this field; however, it is possible to match a range of objects by entering only a prefix. For instance, the pattern:

         .1.3.6.1.4.1.11.2.17

would match:

.1.3.6.1.4.1.11.2.17.1

.1.3.6.1.4.1.11.2.17.2

and so on.

- **Generic ID**

  From the list, select the appropriate Generic Trap ID. Possible values are:

  - (0) **ColdStart**
  - (1)**WarmStart**
  - (2) **LinkDown**
  - (3) **LinkUp**
  - (4) **Authentification**
  - (5) **EgpNeighborLoss**
  - (6) **EnterpriseSpecific**
  - (7) **don't care**

  If you select **(6) EnterpriseSpecific**, you can type in the specific trap ID. Select **don't care** to intercept any kind of trap.

- **Specific ID**

  Type in the specific trap ID if you have selected **(6)EnterpriseSpecific** in Generic Trap. Enterprise-specific SNMP traps can be implemented by vendors on their specific network devices. The specific trap ID is used to identify the source of the trap.

NOTE:
The SNMP syntax used by the editor requires that the trap string begins with a point.

- Configure the condition variable bindings

  Select the variable bindings you want the policy to monitor, and write one or more match patterns for each binding. You can use pattern-matching rules when matching variable bindings.

  To specify pattern matching options such as case sensitivity and field separators for the rule, click the  button. If you do not specify pattern matching options for the rule, either the defaults (case sensitive; a blank and the tab character as separators) or the default options set for the policy will be used.

  1 represents the first variable binding in the event, 2 the second variable, and so on. You do not need to prefix the variable with a dollar sign ($); BSM Integration Adapterdoes this automatically.

- Configure event attributes

  The event attributes tab enables you to set the event attributes for a specific event (or for the event defaults). These attributes are visible in the Operations Management event browser and help the user to organize and evaluate the events.

  - **Title**: Brief description of the nature of the event.
  - **Description**:Detailed description of the event.

- **Severity**: Severity assigned to the event. Accept the severity that is set in the event defaults or choose a specific event severity: Critical, Major, Minor, Warning, Normal.
- **Time Created**: Date and time when the event was created.

  If you leave the attribute empty, then the date and time when the agent created the event displays in the Operations Management event browser. This time always displays using the time zone of the agent at creation time (for example, 11:30 (CET/winter). This means that this time always displays in this fixed time zone.
- **Category**: Name of the logical group to which the event belongs (for example, Database, Security, or Network). The event category is similar in concept to the HP Operations Manager message group.
- **Subcategory**: Name of the logical subgroup (category) to which the event belongs (for example, Oracle (database), Accounts (security), or Routers (network))
- **ETI**: Display name of the event type indicator (ETI) used to calculate the status reported by the event and the current value (for example, Web application state:Slow).
- **Node**: Host system where the event occurred.
- **Related CI**: Name of the impaired configuration item (CI) where the event occurred.
- **Source Manager**: Name and instance of the event and performance monitoring solution that provides events to BSM Integration Adapter (for example, Sitescope:mgmt1.example.com or SCOM:mgmt2.example.com).
- **Send with closed status** Sets the event's lifecycle status to Closed before sending it to the Operations Management event browser.

- Configure event correlation

  Event correlation helps to prevent the Operations Management event browser from becoming cluttered by events that describe the same problem. When event correlation is enabled, you can set the type of duplicate event suppression and define the method used to suppress duplicate events.
  - **Event Key**: An identifier used to identify duplicates and for Close Events with Key.
  - **Close Events with Key**: If events with the event key that you type here exist in the Operations Management event browser when this event is received, these events are automatically closed. You can use pattern matching and variables to match multiple event keys. For example, consider the following pattern:

    ```
    <$MSG_SEV>:<$MSG_NODE_NAME>:<_><5*>
    ```

    This pattern is evaluated by first replacing the variables with the values that they resolve to, for example:

    ```
    critical:cabbage.example.com:<_><5*>
    ```

    This pattern is then compared using pattern matching rule against the event keys for all events in the Operations Management event browser. The pattern above would match the following event keys:

    ```
    critical:cabbage.example.com:  12345
    critical:cabbage.example.com:  TEST1
    ```

    When writing patterns for this box, note the following:

- Although user-defined variables can be included in this box, these variables can only be expanded after the policy is deployed and therefore are not included in the syntax check that is performed when the policy is saved. Because of this, it is important to ensure that any user-defined variables in this box are correctly used.

- **Deduplication on Server**: Clear to disable deduplication on the server. Stops automatic closing of new events that are duplicates of existing events.

Suppress events which are:

- **Generated by same rule**: Select this option to suppress events that match the pattern specified for the selected rule. This is a more general setting for the suppression of duplicate events. For example, an XML log file entry policy might contain a rule with this match pattern: `Error Message<#>` The log file lines `Error Message10` and Error `Message20` are not identical, but would both match this rule.
- **Generated by the same input event**: Select this option to suppress events that were sent in response to two separate input events that are identical except for the date and time that the event was generated (for example, identical entries in an XML log file).
- **Identical relative to their attributes**: Select this option to suppress either events that have the same event key or (if no event key is present) events that have identical event attributes (except for the date and time that the event was generated).

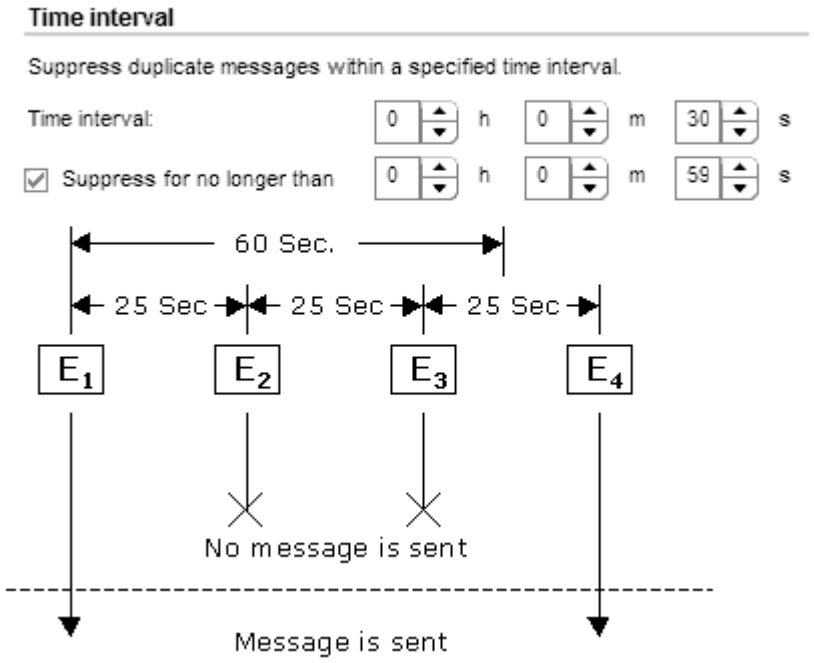
Suppression Method

For event correlation, you can define one of three correlation methods:

- **Time interval:** This correlation method lets you define an interval during which duplicate events will be ignored. For more information, read this detailed example.

Time interval correlation example

In the illustration below, the interval is set to 30 seconds, but the suppression is limited to 60 seconds.

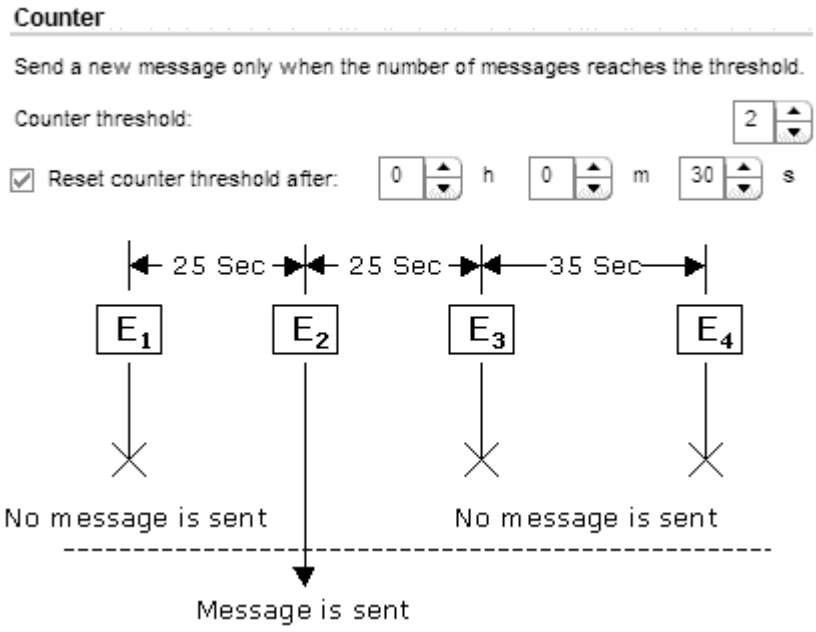


The E_x represents events that are identical.

- The first input event (E1) matches a rule in the policy. The policy sends an event and starts timing.
- A second matching event (E2) occurs 25 seconds later. This event occurred *less than 30* seconds after the first event, and is therefore suppressed.
- A third matching event (E3) occurs *less than 30 seconds after the second event*, and so is also suppressed.
- The next matching event (E4) occurs less than thirty seconds after the third event, but is also *more than 60 seconds after the first event*, and therefore the policy sends an event.

- **Counter:** This correlation method counts the number of matching input events and sends an event only after the number of matching input events equals the counter threshold. The counter can also be reset to zero after a time period that you specify. For more information, read this detailed example.

Counter correlation example



The E_x represent events that are identical.

- The first input event (E1) matches a rule in the policy, and the counter increments to one. No event is sent.
- A second matching event (E2) occurs, the counter increments to two, an event is sent, and the counter resets.
- A third matching event (E3), and the counter increments to one no event is sent.
- The next matching event (E4) occurs *more than thirty seconds* after the third event. Since at thirty seconds the counter was reset to zero, the counter now increments to one no event is sent.

- **Time interval/Counter** 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 an event to the Operations Management event browser.

Note: If you specify just time interval correlation or just counter-based correlation in an individual event, any eventdefaults for the other correlation method also apply. For example, if you specify time interval correlation for an event, and the event defaults specify counter-based correlation, the combined time interval and counter-based correlation applies to both new rules and existing rules.

You can change this default behavior, so that only the correlation method that you specify in the individual event applies. To change the default behavior, set the parameter `OPC_IGNORE_DEFAULT_MSG_CORRELATION=TRUE` in the `eaagt` namespace on the node. You can configure this parameter using `ovconfchg` or `ovconfpar` at a command prompt.

- Configure custom attributes

  Custom attributes are additional attributes that contain any information that is meaningful to you.

For example, you might add a company name, contact information, or a city location to an event. You can have more than one CA attached to a single event. This attribute information displays in the Operations Management event browser in a column you have previously created to contain it.

To create a new event custom attribute:

i. Click * in the toolbar.

ii. Type the name of the event custom attribute. The name is case-insensitive.

iii. Type a value for the event custom attribute.

- Configure the message stream interface

  This tab lets you configure the interface between messages and external programs on the HP Operations agent.

  The message stream interface allows external applications to interact with the internal message flow of the HP Operations agent. The external application can be a read-write application, for example, a message processing program that can read HP Operations messages, modify attributes, and generate new messages for retransmission to the server. The application could also read messages, or send its own messages.

  When you enable the message stream interface, you can also allow external applications using the interface to set up automatic or operator-initiated commands.

  Select **Agent message stream interface** to allow messages to be directed to the **message stream interface** on the node. When switched on, you can choose between the following options:

  - Divert a message to the message stream interface instead of to the server when a message is requested by an external application.
  - Send the message to the server, and a copy of the message to the message stream interface.

- Configure OM attributes

  The OM attributes tab enables you to set additional attributes for a specific event (of for the event defaults). These attributes are visible in the Additional Information tab of the Operations Management event browser and help the user to organize and evaluate the events.

  - **Application**: Application that caused the event to occur. Unlike the Related CI attribute, which is a direct relationship to a CI in the ODB, the application attribute is a simple string-type attribute (for example, Oracle and OS).
  - **Object**: Device such as a computer, printer, or modem. Unlike the Related CI attribute, which is a direct relationship to a CI in the ODB, the application attribute is a simple string-type attribute (for example, C:, and /dev/spool).
  - **Type**: String used to organize different types of events within an event category or subcategory (for example, users or applications, accounts and security).
  - **HPOM Service ID**: ID of the service associated the event. A service ID is a unique identifier for a service and can be used in BSM to identify the node and CI associated with the event.

- Add policy variables

  You can use policy variables in event attributes. BSM Integration Adapter replaces the variables with the appropriate values in the generated event.

It is often useful to surround the variable with quotation marks, especially if it may return a value that contains spaces.

`<$#>`
Returns the number of variables in an enterprise-specific SNMP event (generic event 6 Enterprise specific ID). Sample output: `2`

`<$*>`
Returns all variables assigned to the event up to the possible fifteen. Sample output: `[1] .1.1 (OctetString): arg1 [2] .1.2 (OctetString): turnip.example.com`

`<$@>`
Returns the time the event was received as the number of seconds since Jan 1, 1970 using the *time_t* representation. Sample output: `859479898`

`<$1>`
Returns one or more of the fifteen possible event parameters that are part of an SNMP event. (<$1> returns the first variable, <$2> returns the second variable, and so on.)

`<$\>1>`
Returns all attributes greater than *n* as *value* strings, useful for printing a variable number of arguments. `<$\>0>` is equivalent to `$*` without sequence numbers, names, or types. Sample output: `bokchoy.example.com`

`<$\>+1>`
Returns all attributes greater than *n* as *name:value* string. Sample output: `.1.2: asparagus.example.com`

`<$+2>`
Returns the nth variable binding as *name:value* . Sample output: `.1.2: artichoke.example.com`

`<$\>-n >`
Returns all attributes greater than *n* as *[seq] name (type): value* strings. Sample output: `[2] .1.2 (OctetString): cauliflower.example.com`

`<$-2>`
Returns the nth variable binding as *[seq] name-type:value* . Sample output: `[2] .1.2 (OctetString): brusselsprouts.example.com`

`<$A>`
Returns the node that produced the event. Sample output: `eggplant.example.com`

`<$C>`
Returns the community of the event. Sample output: `public`

`<$c>`
Returns the event's category. Sample output: `SNMP`

`<$E>`
Returns the enterprise ID of the event. Sample output: `.1.3.6.1.4.1.11.2.17.1`

`<$e>`
Returns the enterprise object ID. Sample output: `.1.3.6.1.4.1.11.2.17.1`

`<$F>`
Returns the textual name of the remote postmaster daemon's computer if the event was forwarded. Sample output: `cress.example.com`

`<$G>`
Returns the generic event ID. Sample output: `6`

`<$MSG_NODE>`

Returns the IP address of the node on which the original event took place. Sample output: `192.168.1.123`

`<$MSG_NODE_NAME>`
Returns the name of the node on which the original event took place. This is the hostname that the agent resolves for the node. This variable is not fixed, however, and can be changed by a policy on a per-message basis. For example, if the policy is intercepting SNMP traps that originate from other devices, you might want to set this variable to the name of the device where the trap originated. If the policy is monitoring a log file on a network share where applications on several nodes write messages, you could extract the name of the node from the error message, save it in a user-defined variable, and assign it to MSG_NODE_NAME.

`<$MSG_OBJECT>`
Returns the name of the object associated with the event. This is set in the Message Defaults section of the policy editor.

`<$MSG_TEXT>`
Returns the full text of the event. In general, there are default texts for all editors derived from incoming event properties. Sample output: `SU 03/19 16:13 + ttyp7 bill-root`

`<$N>`
Returns the event name (textual alias) of the event format specification used to format the event, as defined in the Event Configurator. Sample output: `OV_Node_Down`

`<$O>`
Returns the name (object identifier) of the event. Sample output: `.1.3.6.1.4.1.11.2.17.1.0.58916865`

`<$o>`
Returns the numeric object identifier of the event. Sample output: `.1.3.6.1.4.1.11.2.17.1.0.58916865`

`<$R>`
Returns the true source of the event. This value is inferred through the transport mechanism which delivered the event. Sample output: `carrot.example.com`

`<$r>`
Returns the implied source of the event. This may not be the true source of the event if the true source is proxying for another source, such as when a monitoring application running locally is reporting information about a remote node. Sample output: `rutabaga.example.com`

`<$S>`
Returns the specific event ID. Sample output: `5891686`

`<$s>`
Returns the event's severity. Sample output: `Normal`

`<$T>`
Returns the event time stamp. Sample output: `0`

`<$V>`
Returns the event type, based on the transport from which the event was received. Currently supported types are SNMPv1, SNMPv2, CMIP, GENERIC, and SNMPv2INFORM. Sample output: `SNMPv1`

`<$X>`
Returns the time the event was received using the local time representation. Sample output: `17:24:58`

`<$x>`
Returns the date the event was received using the local date representation. Sample output: `03/27/10`

5. Click  in the toolbar to save the policy.

6. In the BSM Integration Adapter user interface, click  in the toolbar to update the list of policies.

Configure SNMP interceptor policy options

The options tab enables you to configure several policy behaviors.

To configure options in SNMP interceptor policies

1. In the BSM Integration Adapter user interface, click  in the toolbar, then click **SNMP**. The SNMP interceptor policy editor opens.

   Alternatively, double-click an existing SNMP interceptor policy to edit it.

2. Click **Options**. Options include:

   - Log local events

     BSM Integration Adapter allows you to define which events, if any, are logged on the node from which they originated. These events are logged on the local node in the log file: `<data_dir>\log\OpC\opcmsglg`.

     Three logging options are available.

     - Log local events **that match a rule and trigger a message**. This selection logs any events in the event source that match the policy rules.
     - Log local events **that match a rule and are ignored**. This selection logs any events in the event source that are suppressed (that is, they do not cause an event to be sent to the Operations Management event browser).
     - Log local events **that don't match any rule**. This selection logs any events that do not match any of the rules in the policy.

   - Capture unmatched events

     You can configure a policy to send an event to the Operations Management event browser when an event does not match any rule in the policy because none of the conditions apply or because the policy does not contain any rules. This ensures that unexpected events that might be important do not go unreported. By default, unmatched events are ignored.

     Each policy that sends unmatched events to the Operations Management event browser creates an event with the default values of the policy.

     Tip: If you want a policy to send events only with the default values, omit all rules from the policy.

     The following options are available:

     - Unmatched events **are sent to the messages browser**
     - Unmatched events **are sent to the acknowledged messages browser**
     - Unmatched events **are ignored** (default)

     Nodes create an event about an unmatched event only if the input event is unmatched in all SNMP interceptor policies on the node. Nodes send only one event for each unmatched input event.

   - Pattern matching options

     The following pattern matching options are available:

- **Case sensitivity**

  You can choose whether the case (uppercase or lowercase) of a text string is considered when the pattern of a rule is compared with the event. When switched on, a match only occurs if the use of uppercase and lowercase letters is exactly the same in both the event and the pattern. This is the default setting.

- **Field separators**

  You can indicate which characters should be considered to be field separators. Field separators are used in the pattern as separator characters for the rule condition. You can define up to seven separators, including these special characters:

  - \n new line (NL)
  - \r carriage return (CR)
  - \t horizontal tab (HT)
  - \f form feed (FF)
  - \v vertical tab (VT)
  - \a alert (BEL)
  - \b backspace (BS)
  - \\ backslash (\)

  For example, if you wanted a backslash, an asterisk, and the letter A to define the fields in the event, you would type ***A** (with no spaces separating the characters).

  If you leave this box empty, the default separators (a blank and the tab character) are used by default.

If you change the pattern matching options, they apply to all new rules in a policy. Click **Apply to all** to apply them to all existing rules in a policy. This overwrites any modifications made to the pattern matching options in individual rules.

You can set case sensitivity and separator characters for individual rules in a policy by clicking the  button in the **Condition Condition Variable Bindings** tab of a rule. Pattern matching options can also be set in the **Event Correlation** tab for the **Close Events with Key** field.

3. Click  in the toolbar to save the policy.
4. In the BSM Integration Adapter user interface, click  in the toolbar to update the list of policies.

Developing XML interceptor policies

This policy type monitors values in an XML log file and responds when the value that you choose appears in the log file. Choose this policy type if you want to monitor entries in an XML log file. XML log file policies are especially suited for integrating events from other applications.

XML interceptor policies process XML files and send events to the Operations Management event browser when certain conditions apply. You can define the attributes of the BSM event based on information in the XML file. This enables you to process events generated by other applications and to convert them to BSM events.

XML interceptor policies process exactly the XML elements and attributes that you define. The XML syntax is not important to the policy, as long as the event information is embedded in XML elements and attributes.

If the application does not store its events in XML files, you may write a program or script that extracts the events from wherever they are stored, formats the data using XML syntax, and generates an XML file with the events. If you have control over the XML elements that are used in the XML file, choose XML elements and attributes that map to event attributes and values. This will simplify the policy.

Note: XML interceptor policies read each line of an XML log file individually. Therefore, you cannot match patterns that span multiple lines in the log file.

To configure an XML log file policy

1. In the BSM Integration Adapter user interface, click  in the toolbar, then click **XML**. The XML log file policy editor opens.

   Alternatively, double-click an existing XML log file policy to edit it.

2. Complete the information in the following tabs:
   - **Properties** include information that is related to the policy itself (for example, the name and description of the policy).
   - The policy **source** is the XML log file that the policy monitors (for example, the path and name of the XML log file).
   - The policy **defaults** include:
     - Default mappings of XML elements and attributes to custom variables
     - Default settings for all events generated by the policy (for example, default event attributes)
   - **Rules** define what the policy should do in response to a specific type of event.
   - **Options** configure several policy behaviors (for example, pattern matching options).
3. Click  in the toolbar to save the policy.
4. In the BSM Integration Adapter user interface, click  in the toolbar to update the list of policies.

Configure XML log file policy properties

Every policy has a set of properties that identify and describe the policy.

To configure properties of XML log file policies

1. In the BSM Integration Adapter user interface, click * in the toolbar, then click **XML**. The XML log file policy editor opens.

   Alternatively, double-click an existing XML log file policy to edit it.

2. Click **Properties**.
3. *Required.* In the **Name** box, type a name that will identify the policy. You can use spaces in policy names. The equal sign (=) is not allowed.
4. *Optional.* In the **Description** box, type a description of what the policy does. You might also add other notes, for example data sources that are used.
5. *Optional.* In the **Category** box, type one or more arbitrary categories. Policy categories may help you to better group your policies. Separate multiple categories with commas.
6. **Policy ID**: BSM Integration Adapter automatically assigns a GUID to the policy when it is first created.
7. **Last modification**: The date that the policy was saved.
8. **Last modified by**: The name of the user active when the policy was saved.
9. Click in the toolbar to save the policy.
10. In the BSM Integration Adapter user interface, click in the toolbar to update the list of policies.

Configure XML log file source properties

The source tab of the XML log file policy editor enables you to specify which log file the policy reads. You can also set options that configure how the policy reads the log file.

XML log files should meet the following criteria so that they can be processed correctly by XML interceptor policies:

- The root element is optional.
- If a root element exists, it must not be closed by an end tag.
- All other XML elements must be complete.

To configure the XML log file source

1. In the BSM Integration Adapter user interface, click * in the toolbar, then click **XML**. The XML log file policy editor opens.

   Alternatively, double-click an existing XML log file policy to edit it.

2. Click **Source**. Configure the XML log file and other options:
   - Select the log file to monitor

     Required. Specify the log file that the policy reads. Type the drive letter and the full path for the location of this log file on the BSM Integration Adapter system. You can use Windows environmental variables (for example **winnt** or **clusterlog**) to make your policies more flexible. The proper syntax for these variables is `<$variablename>`.

You can also call a script or command that returns the path and name of the log file you want to monitor. For example, type

<`command`>

where command is the name of a script that returns the path and name of the log file you want to monitor. The command can also return more than one log file path separated by spaces. The HP Operations agent monitors each of the files using the same options and conditions as configured for this policy. This is very useful when you want to dynamically determine the log file path or monitor multiple instances of a log file.

- Set the log file polling interval

  Optional. You can indicate how often the policy should read the log file. This period of time is the polling interval. The polling interval should be as large as possible, although this depends on the amount of new data written to the file and the read mode that you choose. Set the interval to no less than 30 seconds; usually 5 minutes is appropriate. Note, however, that a policy begins to evaluate data *after* the first polling interval passes. A shorter polling interval is better when you are testing a policy.

- Select the log file character set

  Optional. Indicate the name of the character set used by the log file that you are monitoring. It is important to choose the correct character set. If the character set that the policy is expecting does not match the character set in the log file, pattern matching may not work, and the event details can have incorrect characters or be truncated in the Operations Management event browser. If you are unsure of which character set is used by the log file that you want to monitor, consult the documentation of the program that writes the log file.

  NOTE:
  The character set of the log file must be convertible to the HP Operations agent node character set. For example, if agent character set is iso88591 (English) then, ACP 1252, ACSII, ISO 8859-1, OEMCP 850, OEMCP 437, ROMAN 8, or EBCDIC may be used. If the agent character set is sjis (Japanese), then ACP 932, ACSII, or EUC may be used. If the agent character set is iso88595 (Cyrillic), then iso88595, ASCII, ACP 1251, or OEMCP 866 may be used.

  The character sets supported by Windows and Linux nodes are:

Character set	Description
ACP 1250	Central European
ACP 1251	Cyrillic
ACP 1252	Western European
ACP 932	Includes all characters defined in the shift-JIS code. This character set is supported by the Japanese versions of Microsoft Windows NT and Microsoft Windows 95/98.
ACSII	English (American Standard Code for Information Interchange)
BIG-5 Taiwanese	Taiwanese
EBCDIC	(Extended Binary-Coded Decimal Interchange Code) Generally used only on large IBM computers.

EUC Japanese	(Extended UNIX Code) Japanese
EUC Korean	(Extended UNIX Code) Korean
EUC Taiwanese	(Extended UNIX Code) Taiwanese
GB-2312-80 Chinese	Chinese
ISO 8859-1	Most West European languages, including French, Spanish, Catalan, Basque, Portuguese, Italian, Albanian, Rhaeto-Romanic, Dutch, German, Danish, Swedish, Norwegian, Finnish, Faeroese, Icelandic, Irish, Scottish, and English. Also Afrikaans, Swahili.
ISO 8859-15	Latin alphabet
ISO 8859-2	Central and Eastern European languages, including Czech, Hungarian, Polish, Romanian, Croatian, Slovak, Slovenian, Sorbian.
ISO 8859-5	Languages that use Cyrillic characters, including Bulgarian, Belorussian, Macedonian, Russian, Serbian and Ukrainian.
ISO 8859-6	Arabic
ISO 8859-7	Greek
ISO 8859-8	Hebrew and Yiddish
ISO 8859-9	Same as ISO 8859-1, but with Turkish, instead of Icelandic.
OEMCP 437	U.S. English
OEMCP 737	Greek (formerly 437G)
OEMCP 775	Baltic
OEMCP 850	All the characters used by most European, North American, and South American languages
OEMCP 852	Slavic (Latin II)
OEMCP 857	IBM Turkish

OEMCP 860	Portuguese
OEMCP 861	Icelandic
OEMCP 862	Hebrew
OEMCP 863	Canadian-French
OEMCP 864	Arabic
OEMCP 865	Nordic
OEMCP 866	Russian
OEMCP 869	IBM Modern Greek
ROMAN 8	European characters
SHIFT-JIS	Microsoft's standard encoding for Japanese.
UCS-2	This codeset is intended to express all characters in the world in a united character set.
UTF-8	(Unicode Transformation Format-8) This codeset is intended to express all characters in the world in a united character set.

- Send event if log file does not exist

  Optional. Select if you want BSM Integration Adapter to send an event if the specified XML file does not exist

- Close after read

  Optional. Select if you want the policy to close the XML file (and release its file handle) after reading it. If you do not select this option and the name of the XML file changes, the policy continues to read the open, renamed XML file instead of processing the new XML file. A short polling interval is recommended when using this option.

- Set the read mode

  Required. The read mode of an XML log file policy indicates whether the policy should process the entire log file or should only process new log file entries. The available read modes are described in the table below. Note that every policy reads log files independently from any other policies. This means, for example, that if "Policy 1" with read mode **Read from beginning (first time)** is enabled on a node where "Policy 2" with the same read mode already exists, "Policy 1" will still read the entire log file after it has been enabled.

  Log file read modes

Mode: Description	Advantage / Disadvantage

Read from last position: The policy reads only new—appended—entries written in the log file while the policy is activated on the node. If the log file decreases in size between readings, then the entire log file is read. Log file entries that are added to the log file when the policy is deactivated are not processed by the policy. Choose this option if you are concerned only with log file entries that occur when the policy is enabled.	**Advantage:** No chance of reading the same entry twice. (Unless the log file decreases in size because some entries were deleted.) **Disadvantage:** Entries written to the log file while the policy is deactivated or the agent is not running will not be processed by the policy.
Read from beginning (first time): The policy reads the complete log file each time the policy is activated or the agent restarts on the node. This ensures that all entries in the log file are compared with the rules in the policy. Each successive time that the policy reads the log file, only new (appended) entries in the log file are processed. Choose this option if you want to ensure that every existing and future entry in the log file will be processed by the policy while it is enabled.	**Advantage:** Every existing and future entry in the log file will be processed by the policy. **Disadvantage:** Duplicate entries can occur if an activated policy is deactivated and reactivated, or if the agent stops and restarts.
Read from beginning (always): The policy reads the complete log file every time it detects that the log file has changed. The policy scans the log file at the specified polling interval. If no change is detected, the log file is not processed. Any log file entries overwritten while the agent is not running or the policy is deactivated will not be evaluated by the policy. Choose this option if you are monitoring a log file that is overwritten, rather than appended.	**Advantage:** Ensures that log files that are overwritten are correctly processed. **Disadvantage:** Only valid for log files that are overwritten, rather than appended.

- Specify the XML event tag

  Required. The XML event tag creates a shortcut to the XML element that you want to monitor. An event tag typically identifies an event record in an XML log file. You can define more than one event tag (for example, `PerformanceAlert` and `AvailabilityAlert`).

  To specify an XML event tag, click * and type the XML element. Alternatively, if you are working with sample data, click **Select** and double-click the XML element in the list.

  Caution: Deleting an event tag that is referenced in a policy corrupts the policy and renders it unusable.

- Load XML sample data

  Optional. Load a sample of the XML log file that you want to monitor with this policy. BSM Integration Adapter makes the XML elements and values of the sample file available to you in the defaults and rules tabs so that you can insert them by dragging and dropping.

To load a sample XML file, click **Load** and then select the file. You can also display the selected XML file by clicking **View**. When you load sample data, BSM Integration Adapter replaces already loaded data with the new data. This does not affect any mappings that are defined based on previously available sample data.

3. Click in the toolbar to save the policy.
4. In the BSM Integration Adapter user interface, click in the toolbar to update the list of policies.

Configure XML log file mapping defaults

The mapping defaults tab enables you to map XML elements and attributes to custom variables.

A custom variable consists of a map name, an optional XML property (XML elements or attributes), and one or more source and target value pairs. For example, you can assign the XML element `Severity` to the map name `map@Severity`, and add a source value of `Warning`. You can then assign the target value `Major` to the variable so that BSM Integration Adapter inserts the value `Major` into the event in all places where the variable is used and the source value is `Warning` in the XML log file.

Map Name	XML Property	Source Value	Target Value
mapSeverity	<$DATA:/SCOM_Alert/Severity>	Warning	Major
mapName	<$DATA:/SCOM_Alert/Name>	Error	Critical

XML properties use the following syntax: <$DATA:/*<XML_property>*

<XML_property> is the XML path, separated by slash marks (/), from the XML event tag to the XML element or attribute.

For example, the custom variable `map@Severity` has the following XML property: `<$DATA:/SCOM_Alert/Severity>` where `Severity` is a child element of `SCOM_Alert`.

XML properties are optional. If you do not assign an XML property to a variable, you must add the source value directly to the variable when you insert the variable in an event attribute.

Note: The Sample Data tab is empty if no sample data has been loaded into the policy or if the sample data does not match any specified XML event tags.

If sample data is available, then the XML Properties section of the Sample Data tab shows all XML elements and attributes that match an XML event tag. (You can identify attributes based on the preceding at sign (@).) The XML Properties section by default shows the short path to the XML property or value. To view the full path, click . The full path begins with the XML event tag specified in the Source tab.

The Values section displays the values of an XML property selected in the XML Properties section. If a value appears more than once, click to show or hide duplicate values. To find values that belong to more than one XML property, select the value and click . The XML Sample Data window opens and shows all XML properties that have the selected value.

When you drag an XML element or attribute from the XML properties list and drop it on the Default Value Mapping List, BSM Integration Adapter automatically adds the default prefix `map` to the map name and inserts the correct path to the XML property. You can then drag one or more XML source values from the XML values list and drop them on the Source Value list. You then finally only have to type the target values.

To configure mapping defaults in XML log file policies

1. In the BSM Integration Adapter user interface, click * in the toolbar, then click **XML**. The XML log file policy editor opens.

   Alternatively, double-click an existing XML log file policy to edit it.
2. Click **Defaults - Mappings**.
3. Create one or more custom variables by defining map names and XML properties, either manually or by dragging them from the XML properties list.
4. Add one or more source and target value pairs to each custom variable, either manually or by dragging them from the XML values list.
5. Click in the toolbar to save the policy.
6. In the BSM Integration Adapter user interface, click in the toolbar to update the list of policies.

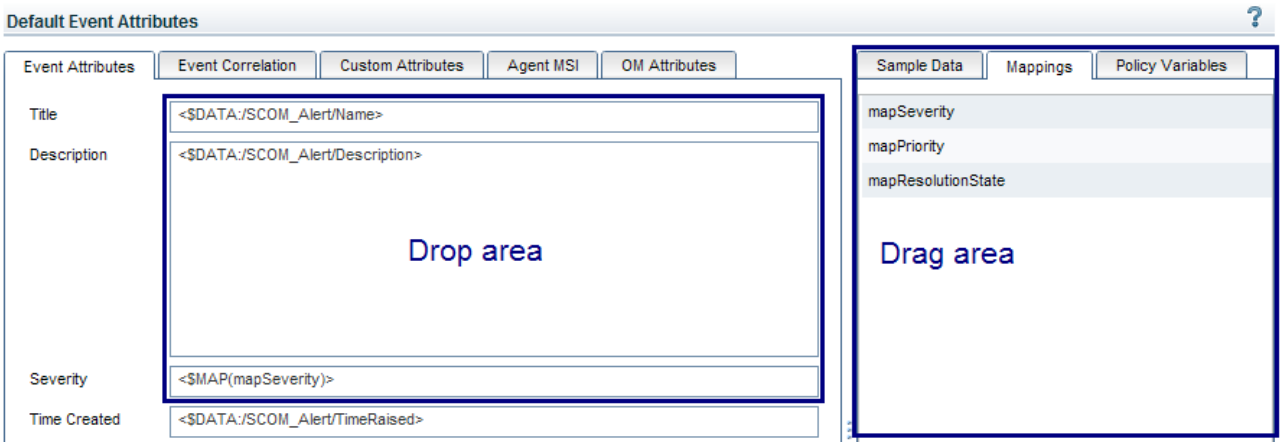
Configure XML log file event defaults

The attribute defaults tab enables you to indicate default settings for all events generated by the policy.

These defaults affect all new and existing rules. You can override the defaults in individual rules if needed. If a rule contains empty event attributes, the agent will use the defaults for the new event.

Tip: The BSM Integration Adapter user interface enables you to drag data from the tabs on the right onto boxes on the left.

The following illustration shows the default event attributes page of XML interceptor policies. You can select data in the Sample Data, Mappings, and Policy Variables tabs on the right and drag it to the Event Attributes, Event Correlation, Custom Attributes, and OM Attributes tabs on the left. BSM Integration Adapter inserts the data at the current cursor position.



To configure attribute defaults in XML log file policies

1. In the BSM Integration Adapter user interface, click * in the toolbar, then click **XML**. The XML log file policy editor opens.

   Alternatively, double-click an existing XML log file policy to edit it.

2. Click **Defaults - Event**. The default settings include:

   - Configure event attributes

     The event attributes tab enables you to set the event attributes for a specific event (or for the event defaults). These attributes are visible in the Operations Management event browser and help the user to organize and evaluate the events.

     - **Title**: Brief description of the nature of the event.
     - **Description**:Detailed description of the event.
     - **Severity**: Severity assigned to the event.
     - **Time Created**: Date and time when the event was created.

       Use the following conventions when specifying the date and time attribute:

       - **Integers.** XML interceptor policies interpret integers in XML files as seconds since 00:00:00 UTC on 1 January 1970 (Unix time). For example, 1276600333 is 15 June 2010, at 11:12:13.
       - **Default time formats.** XML interceptor policies by default interpret the following time formats:

         yyyy-mm-ddTHH:MM:SS (for example, 2010-06-15T11:12:13)

         mm/dd/yyyy HH:MM:SS (for example, 06/15/2010 11:12:13)
       - **Pattern matching.** You can use the function `<$DATETIME(FORMAT,VALUE)>` to specify a pattern (`FORMAT`) that matches the time string in a given XML property (`VALUE`). You can use standard BSM Integration Adapter pattern-matching rules when matching values. By default, pattern matching for the time format is case sensitive. The default field separators are the space and the tab characters.

         `FORMAT` must be enclosed in quotation marks (`"FORMAT"`) and accepts the following variables:

         `H` (hours), `M` (minutes), `S` (seconds). If H, M, or S is not set, the hour, minute, or second displays as zero.

         `d` (day), `m` (month), `y` (year). If d or m is not set, the day or month display as one. If y is not set, the current year is assumed. If y is less than 100, the current millennium is assumed; for example, if y matches 10, the year displays as 2010. It is not possible to match a year earlier than 1970.

         `p` (P.M.) If p is set, XML interceptor policies add 12 hours to the hours that precede the variable.

         `VALUE` is the XML property or value to match.

         XML properties use the following syntax: <$DATA:/<*XML_property*>

<*XML_property*> is the XML path, separated by slash marks (/), from the XML event tag to the XML element or attribute.

Examples:

To match the time format `06/15/2010 11:12:13` in the XML property `<$DATA:/SCOM_Alert/TimeRaised>`, type
`<$DATETIME("^<#.m>/<#.d>/<#.y> <#.H>:<#.M>:<#.S>$",<$DATA:/SCOM_Alert/TimeRaised>)>`

To match the time format `11:12 15.06.2010` in the XML property `<$DATA:/SCOM_Alert/TimeRaised>`, type
`<$DATETIME("^<#.H>:<#.M> <#.d>.<#.m>.<#.y>$",<$DATA:/SCOM_Alert/TimeRaised>)>`

To match the time format `06/15/2010 1:35 PM` in the XML property `<$DATA:/SCOM_Alert/TimeRaised>`, type
`<$DATETIME("^<#.m>/<#.d>/<#.y> <#.H>:<#.M> <2*.p>$",<$DATA:/SCOM_Alert/TimeRaised>)>`

If you leave the attribute empty or if none of the time formats above can be matched, then the date and time when the agent created the event displays in the Operations Management event browser. This time always displays using the time zone of the agent at creation time (for example, 11:30 (CET/winter). This means that this time always displays in this fixed time zone.

- **Category**: Name of the logical group to which the event belongs (for example, Database, Security, or Network). The event category is similar in concept to the HP Operations Manager message group.
- **Subcategory**: Name of the logical subgroup (category) to which the event belongs (for example, Oracle (database), Accounts (security), or Routers (network))
- **ETI**: Display name of the event type indicator (ETI) used to calculate the status reported by the event and the current value (for example, Web application state:Slow).
- **Node**: Host system where the event occurred.
- **Related CI**: Name of the impaired configuration item (CI) where the event occurred.
- **Source Manager**: Name and instance of the event and performance monitoring solution that provides events to BSM Integration Adapter (for example, Sitescope:mgmt1.example.com or SCOM:mgmt2.example.com).

- Configure event correlation

  Event correlation helps to prevent the Operations Management event browser from becoming cluttered by events that describe the same problem. When event correlation is enabled, you can set the type of duplicate event suppression and define the method used to suppress duplicate events.

  - **Event Key**: An identifier used to identify duplicates and for Close Events with Key.
  - **Close Events with Key**: If events with the event key that you type here exist in the Operations Management event browser when this event is received, these events are automatically closed. You can use pattern matching and variables to match multiple event keys. For example, consider the following pattern:

    `<$MSG_SEV>:<$MSG_NODE_NAME>:<_><5*>`

This pattern is evaluated by first replacing the variables with the values that they resolve to, for example:

```
critical:cabbage.example.com:<_><5*>
```

This pattern is then compared using pattern matching rule against the event keys for all events in the Operations Management event browser. The pattern above would match the following event keys:

```
critical:cabbage.example.com:  12345
critical:cabbage.example.com:  TEST1
```

When writing patterns for this box, note the following:

- Although user-defined variables can be included in this box, these variables can only be expanded after the policy is deployed and therefore are not included in the syntax check that is performed when the policy is saved. Because of this, it is important to ensure that any user-defined variables in this box are correctly used.

- **Deduplication on Server**: Clear to disable deduplication on the server. Stops automatic closing of new events that are duplicates of existing events.

Suppress events which are:

- **Generated by same rule**: Select this option to suppress events that match the pattern specified for the selected rule. This is a more general setting for the suppression of duplicate events. For example, an XML log file entry policy might contain a rule with this match pattern: `Error Message<#>` The log file lines `Error Message10` and Error `Message20` are not identical, but would both match this rule.

- **Generated by the same input event**: Select this option to suppress events that were sent in response to two separate input events that are identical except for the date and time that the event was generated (for example, identical entries in an XML log file).

- **Identical relative to their attributes**: Select this option to suppress either events that have the same event key or (if no event key is present) events that have identical event attributes (except for the date and time that the event was generated).

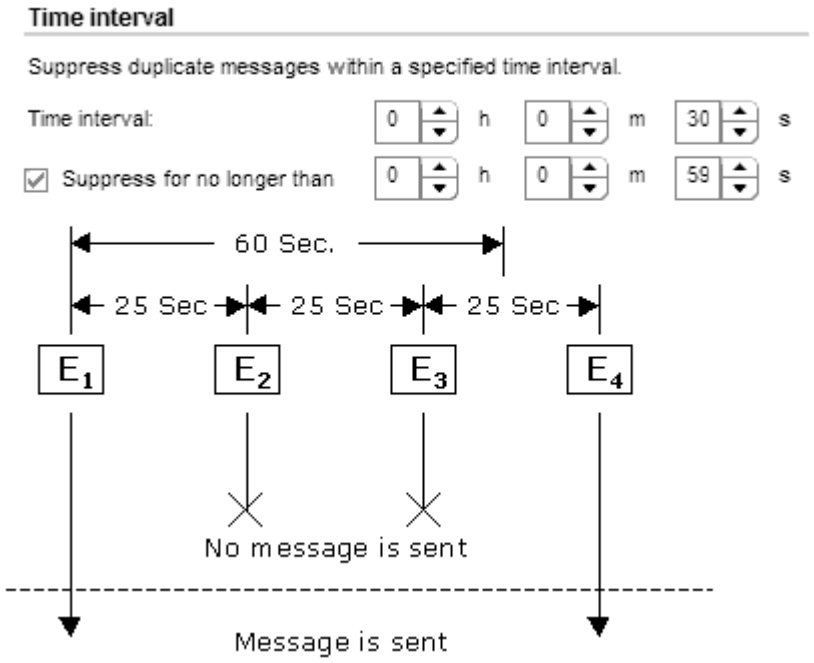
Suppression Method

For event correlation, you can define one of three correlation methods:

- **Time interval:** This correlation method lets you define an interval during which duplicate events will be ignored. For more information, read this detailed example.

Time interval correlation example

In the illustration below, the interval is set to 30 seconds, but the suppression is limited to 60 seconds.

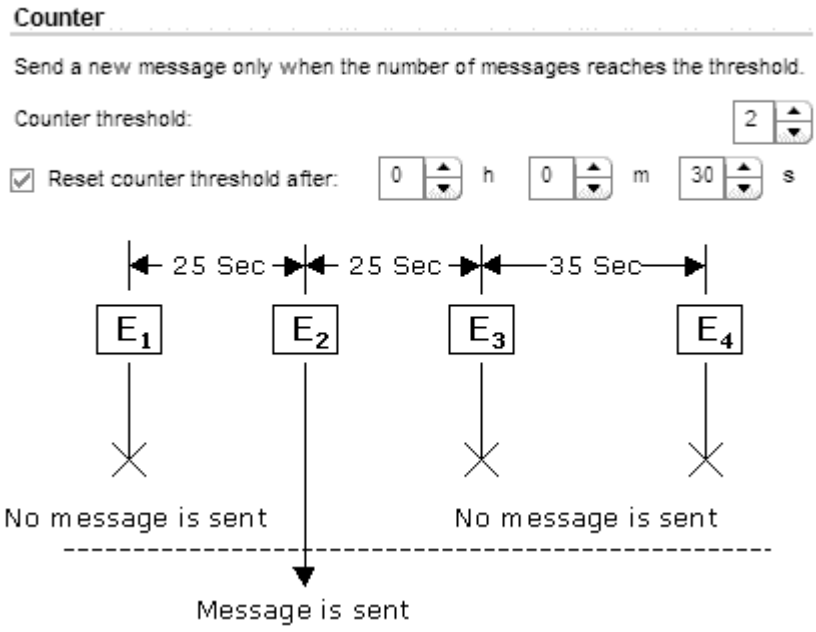


The E_x represents events that are identical.

- The first input event (E1) matches a rule in the policy. The policy sends an event and starts timing.
- A second matching event (E2) occurs 25 seconds later. This event occurred *less than 30* seconds after the first event, and is therefore suppressed.
- A third matching event (E3) occurs *less than 30 seconds after the second event*, and so is also suppressed.
- The next matching event (E4) occurs less than thirty seconds after the third event, but is also *more than 60 seconds after the first event*, and therefore the policy sends an event.

- **Counter:** This correlation method counts the number of matching input events and sends an event only after the number of matching input events equals the counter threshold. The counter can also be reset to zero after a time period that you specify. For more information, read this detailed example.

Counter correlation example



The E_x represent events that are identical.

- The first input event (E1) matches a rule in the policy, and the counter increments to one. No event is sent.
- A second matching event (E2) occurs, the counter increments to two, an event is sent, and the counter resets.
- A third matching event (E3), and the counter increments to one no event is sent.
- The next matching event (E4) occurs *more than thirty seconds* after the third event. Since at thirty seconds the counter was reset to zero, the counter now increments to one no event is sent.

- **Time interval/Counter** 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 an event to the Operations Management event browser.

Note: If you specify just time interval correlation or just counter-based correlation in an individual event, any eventdefaults for the other correlation method also apply. For example, if you specify time interval correlation for an event, and the event defaults specify counter-based correlation, the combined time interval and counter-based correlation applies to both new rules and existing rules.

You can change this default behavior, so that only the correlation method that you specify in the individual event applies. To change the default behavior, set the parameter `OPC_IGNORE_DEFAULT_MSG_CORRELATION=TRUE` in the `eaagt` namespace on the node. You can configure this parameter using `ovconfchg` or `ovconfpar` at a command prompt.

- Configure custom attributes

  Custom attributes are additional attributes that contain any information that is meaningful to you.

For example, you might add a company name, contact information, or a city location to an event. You can have more than one CA attached to a single event. This attribute information displays in the Operations Management event browser in a column you have previously created to contain it.

To create a new event custom attribute:

i. Click * in the toolbar.

ii. Type the name of the event custom attribute. The name is case-insensitive.

iii. Type a value for the event custom attribute.

- Configure the message stream interface

  This tab lets you configure the interface between messages and external programs on the HP Operations agent.

  The message stream interface allows external applications to interact with the internal message flow of the HP Operations agent. The external application can be a read-write application, for example, a message processing program that can read HP Operations messages, modify attributes, and generate new messages for retransmission to the server. The application could also read messages, or send its own messages.

  When you enable the message stream interface, you can also allow external applications using the interface to set up automatic or operator-initiated commands.

  Select **Agent message stream interface** to allow messages to be directed to the **message stream interface** on the node. When switched on, you can choose between the following options:

  - Divert a message to the message stream interface instead of to the server when a message is requested by an external application.
  - Send the message to the server, and a copy of the message to the message stream interface.

- Configure OM attributes

  The OM attributes tab enables you to set additional attributes for a specific event (of for the event defaults). These attributes are visible in the Additional Information tab of the Operations Management event browser and help the user to organize and evaluate the events.

  - **Application**: Application that caused the event to occur. Unlike the Related CI attribute, which is a direct relationship to a CI in the ODB, the application attribute is a simple string-type attribute (for example, Oracle and OS).
  - **Object**: Device such as a computer, printer, or modem. Unlike the Related CI attribute, which is a direct relationship to a CI in the ODB, the application attribute is a simple string-type attribute (for example, C:, and /dev/spool).
  - **Type**: String used to organize different types of events within an event category or subcategory (for example, users or applications, accounts and security).
  - **HPOM Service ID**: ID of the service associated the event. A service ID is a unique identifier for a service and can be used in BSM to identify the node and CI associated with the event.

- Add XML sample data

  If you are working with sample data, you can drag XML properties (XML elements and attributes) and values from the Sample Data tab and drop them onto the attribute boxes. Alternatively, you can type the path to the XML property or value directly into the attribute box.

  XML properties use the following syntax: <$DATA:/<*XML_property*>

<*XML_property*> is the XML path, separated by slash marks (/), from the XML event tag to the XML element or attribute.

BSM Integration Adapter replaces the XML property at runtime with the value of the specified XML element or attribute. If you insert an XML value, the value will be used.

Note: The Sample Data tab is empty if no sample data has been loaded into the policy or if the sample data does not match any specified XML event tags.

If sample data is available, then the XML Properties section of the Sample Data tab shows all XML elements and attributes that match an XML event tag. (You can identify attributes based on the preceding at sign (@).) The XML Properties section by default shows the short path to the XML property or value. To view the full path, click  . The full path begins with the XML event tag specified in the Source tab.

The Values section displays the values of an XML property selected in the XML Properties section. If a value appears more than once, click  to show or hide duplicate values. To find values that belong to more than one XML property, select the value and click  . The XML Sample Data window opens and shows all XML properties that have the selected value.

- Add mappings

  Mappings are custom variables that you define in the mappings tab. To insert a custom variable, you can drag it from the Mappings list and drop it on the event attribute.

  Alternatively, type the custom variable into the attribute box using the following syntax:

  <$MAP(<*custom_variable*>)> where <*custom_variable*> is the map name of the variable (for example, `<$MAP(map@Severity)>`).

  If the custom variable does not have an XML property assigned, use the following syntax:

  <$MAP(<*custom_variable*>,<<*source_value*>>) where <*source_value*> can be one of the following:

  - XML path to the source value, for example `<$MAP(map@Severity, <$DATA/SCOM_Alert/Severity>)>`
  - The source value itself, for example `<$MAP(map@Severity, Warning)>`

- Add policy variables

  You can use policy variables in event attributes. BSM Integration Adapter replaces the variables with the appropriate values in the generated event.

  It is often useful to surround the variable with quotation marks, especially if it may return a value that contains spaces.

  `<$LOGFILE>`
  Returns the name of the log file that contains the input event. Sample output:`program_log.txt`

  `<$LOGPATH>`
  Returns the name and path of the log file that contains the input event. Sample output:`C:\temp\mylogfile\program_log.txt`

  `<$MSG_NODE>`
  Returns the IP address of the node on which the original event took place. Sample output: `192.168.1.123`

  `<$MSG_NODE_NAME>`

Returns the name of the node on which the original event took place. This is the hostname that the agent resolves for the node. This variable is not fixed, however, and can be changed by a policy on a per-message basis. For example, if the policy is intercepting SNMP traps that originate from other devices, you might want to set this variable to the name of the device where the trap originated. If the policy is monitoring a log file on a network share where applications on several nodes write messages, you could extract the name of the node from the error message, save it in a user-defined variable, and assign it to MSG_NODE_NAME.

`<$MSG_TEXT>`
Returns the full text of the event. In general, there are default texts for all editors derived from incoming event properties. Sample output: `SU 03/19 16:13 + ttyp7 bill-root`

3. Click  in the toolbar to save the policy.
4. In the BSM Integration Adapter user interface, click  in the toolbar to update the list of policies.

Configure XML log file rules

Rules define what a policy should do in response to a specific type of event. Each rule consists of a condition and of settings for the event generated by the policy. The condition is the part of a policy that describes the type of event in the source. The settings enable you to configure the event that BSM Integration Adapter sends to the Operations Management event browser.

A policy must contain at least one rule. If the policy contains multiple rules, it is important to remember that the rules are evaluated in a specific order, and that when one condition is matched, no additional rules will be evaluated.

The rule types are:

- **Event on matched condition**

  If matched, BSM Integration Adapter sends an event to the Operations Management event browser. The event uses the settings defined for the rule. If you do not configure these settings, the default settings are used.

- **Suppress on matched condition**

  If matched, BSM Integration Adapter stops processing and does not send an event to the Operations Management event browser.

- **Suppress on unmatched condition**

  If not matched, BSM Integration Adapter stops processing and does not send an event to the Operations Management event browser.

Note: In all cases, if a rule evaluates as true, no more rules are processed. It is important to pay attention to the rule order.

The order in which rules are evaluated has a large effect on the type of messages you receive. It also affects the speed with which messages are sent and the amount of processor time that is required by the policy.

For example, you might have a policy that monitors CPU activity, containing these two rules:

1. `If usage is greater than 80%,`
   `send a warning message and stop processing rules.`

2. `If usage is greater than 95%,`
   `send a critical message and stop processing rules.`

If the rules were evaluated in the order shown, disk usage of 99% would only produce a warning message. If the order were reversed, however, a critical message would be sent. You could solve the problem by making the rules more specific, so that the order was not important:

1. `If usage is between 80% and 94%,`
   `send a warning message and stop processing rules.`
2. `If usage is greater than 95%,`
   `send a critical message and stop processing rules.`

In the example above, disk usage of 99% produces a critical message regardless of which rule is evaluated first. However, if the rules are evaluated in the order shown, disk usage of 99% is evaluated by two rules. If the order were reversed, it would be evaluated only by the first rule, thereby sending the message more quickly and reducing processing time on the managed node.

Tip: The BSM Integration Adapter user interface enables you to drag data from the tabs on the right onto boxes on the left.

The following illustration shows a section of the rules page of XML interceptor policies. You can select data in the Sample Data, Pattern Matching Variables, Mappings, and Policy Variables tabs on the right and drag it to the Condition, Event Attributes, Event Correlation, Custom Attributes, and OM Attributes tabs on the left. BSM Integration Adapter inserts the data at the current cursor position.

To configure rules in XML log file policies

1. In the BSM Integration Adapter user interface, click * in the toolbar, then click **XML**. The XML log file policy editor opens.

   Alternatively, double-click an existing XML log file policy to edit it.

2. Click **Rules**.

3. Click * in the toolbar and select the rule type. Then type the name of the rule. After a rule has been added, you can change the rule type by clicking the current rule type in the list of rules and selecting another rule type from the drop-down list.

   Alternatively, select an existing rule and click to copy the rule. You can then rename the copied rule and edit it.

4. Configure the following settings for the rule:

   - Configure conditions

     The condition tab enables you to specify the XML properties and values that the policy searches

for in the XML log file that the policy monitors. If the policy finds a match, it may or may not generate an event, depending on the rule type.

i. Specify the XML element or attribute that the policy searches for.

If you are working with sample data, you can drag and drop the XML element or attribute from the XML Properties list to the Properties field. Alternatively, click * and type the XML property into the box. You must specify the XML path from the XML event tag to the property, separated by slash marks (/).

ii. Select the operator. BSM Integration Adapter provides the following operators:

Operator	Description
==	Equal to
!=	Not equal to
<	Less than
<=	Less than or equal to
>	More than
>=	More than or equal to
~=	Pattern operator

iii. In the Operand field, type the value or pattern that you want the policy to compare with the XML log file line. If you are working with sample data, you can drag the value from the XML Values list and drop it on the operand box.

You can use standard BSM Integration Adapter pattern-matching rules when matching values. Select the ~= pattern operator and click  to open the pattern matching expression toolbox. The toolbox also enables you to specify pattern matching options such as case sensitivity and field separators for the rule. If you do not specify pattern matching options for the rule, either the defaults (case sensitive; a blank and the tab character as separators) or the default options set for the policy will be used.

- Configure event attributes

The event attributes tab enables you to set the event attributes for a specific event (or for the event defaults). These attributes are visible in the Operations Management event browser and help the user to organize and evaluate the events.

- **Title**: Brief description of the nature of the event.
- **Description**:Detailed description of the event.
- **Severity**: Severity assigned to the event.
- **Time Created**: Date and time when the event was created.

  Use the following conventions when specifying the date and time attribute:

  - **Integers.** XML interceptor policies interpret integers in XML files as seconds since 00:00:00 UTC on 1 January 1970 (Unix time). For example, 1276600333 is 15 June 2010,

at 11:12:13.

- **Default time formats.** XML interceptor policies by default interpret the following time formats:

  yyyy-mm-ddTHH:MM:SS (for example, 2010-06-15T11:12:13)

  mm/dd/yyyy HH:MM:SS (for example, 06/15/2010 11:12:13)

- **Pattern matching.** You can use the function `<$DATETIME(FORMAT,VALUE)>` to specify a pattern (`FORMAT`) that matches the time string in a given XML property (`VALUE`). You can use standard BSM Integration Adapter pattern-matching rules when matching values. By default, pattern matching for the time format is case sensitive. The default field separators are the space and the tab characters.

  `FORMAT` must be enclosed in quotation marks (`"FORMAT"`) and accepts the following variables:

  `H` (hours), `M` (minutes), `S` (seconds). If H, M, or S is not set, the hour, minute, or second displays as zero.

  `d` (day), `m` (month), `y` (year). If d or m is not set, the day or month display as one. If y is not set, the current year is assumed. If y is less than 100, the current millennium is assumed; for example, if y matches 10, the year displays as 2010. It is not possible to match a year earlier than 1970.

  `p` (P.M.) If p is set, XML interceptor policies add 12 hours to the hours that precede the variable.

  `VALUE` is the XML property or value to match.

  XML properties use the following syntax: <$DATA:/*<XML_property>*

  <XML_property> is the XML path, separated by slash marks (/), from the XML event tag to the XML element or attribute.

  Examples:

  To match the time format `06/15/2010 11:12:13` in the XML property `<$DATA:/SCOM_Alert/TimeRaised>`, type
  `<$DATETIME("^<#.m>/<#.d>/<#.y> <#.H>:<#.M>:<#.S>$",<$DATA:/SCOM_Alert/TimeRaised>)>`

  To match the time format `11:12 15.06.2010` in the XML property `<$DATA:/SCOM_Alert/TimeRaised>`, type
  `<$DATETIME("^<#.H>:<#.M> <#.d>.<#.m>.<#.y>$",<$DATA:/SCOM_Alert/TimeRaised>)>`

  To match the time format `06/15/2010 1:35 PM` in the XML property `<$DATA:/SCOM_Alert/TimeRaised>`, type
  `<$DATETIME("^<#.m>/<#.d>/<#.y> <#.H>:<#.M> <2*.p>$",<$DATA:/SCOM_Alert/TimeRaised>)>`

If you leave the attribute empty or if none of the time formats above can be matched, then the date and time when the agent created the event displays in the Operations Management event browser. This time always displays using the time zone of the agent at creation time (for example, 11:30 (CET/winter). This means that this time always displays in this fixed time zone.

- **Category**: Name of the logical group to which the event belongs (for example, Database,

Security, or Network). The event category is similar in concept to the HP Operations Manager message group.

- **Subcategory**: Name of the logical subgroup (category) to which the event belongs (for example, Oracle (database), Accounts (security), or Routers (network))
- **ETI**: Display name of the event type indicator (ETI) used to calculate the status reported by the event and the current value (for example, Web application state:Slow).
- **Node**: Host system where the event occurred.
- **Related CI**: Name of the impaired configuration item (CI) where the event occurred.
- **Source Manager**: Name and instance of the event and performance monitoring solution that provides events to BSM Integration Adapter (for example, Sitescope:mgmt1.example.com or SCOM:mgmt2.example.com).
- **Send with closed status** Sets the event's lifecycle status to Closed before sending it to the Operations Management event browser.

- Configure event correlation

  Event correlation helps to prevent the Operations Management event browser from becoming cluttered by events that describe the same problem. When event correlation is enabled, you can set the type of duplicate event suppression and define the method used to suppress duplicate events.

  - **Event Key**: An identifier used to identify duplicates and for Close Events with Key.
  - **Close Events with Key**: If events with the event key that you type here exist in the Operations Management event browser when this event is received, these events are automatically closed. You can use pattern matching and variables to match multiple event keys. For example, consider the following pattern:

    ```
    <$MSG_SEV>:<$MSG_NODE_NAME>:<_><5*>
    ```

    This pattern is evaluated by first replacing the variables with the values that they resolve to, for example:

    ```
    critical:cabbage.example.com:<_><5*>
    ```

    This pattern is then compared using pattern matching rule against the event keys for all events in the Operations Management event browser. The pattern above would match the following event keys:

    ```
    critical:cabbage.example.com:  12345
    critical:cabbage.example.com:  TEST1
    ```

    When writing patterns for this box, note the following:

    - Although user-defined variables can be included in this box, these variables can only be expanded after the policy is deployed and therefore are not included in the syntax check that is performed when the policy is saved. Because of this, it is important to ensure that any user-defined variables in this box are correctly used.

  - **Deduplication on Server**: Clear to disable deduplication on the server. Stops automatic closing of new events that are duplicates of existing events.

  Suppress events which are:

- **Generated by same rule**: Select this option to suppress events that match the pattern specified for the selected rule. This is a more general setting for the suppression of duplicate events. For example, an XML log file entry policy might contain a rule with this match pattern: `Error Message<#>` The log file lines `Error Message10` and Error `Message20` are not identical, but would both match this rule.
- **Generated by the same input event**: Select this option to suppress events that were sent in response to two separate input events that are identical except for the date and time that the event was generated (for example, identical entries in an XML log file).
- **Identical relative to their attributes**: Select this option to suppress either events that have the same event key or (if no event key is present) events that have identical event attributes (except for the date and time that the event was generated).

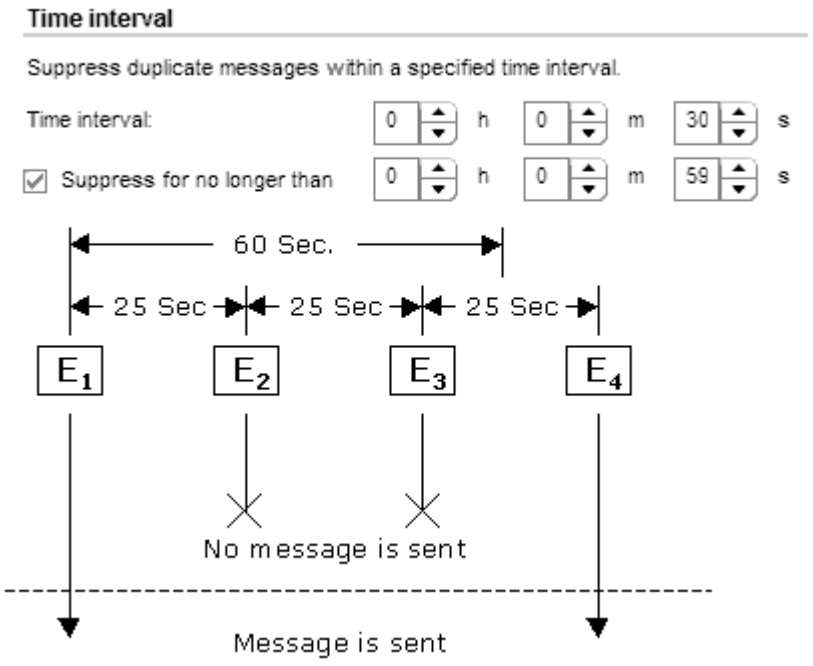
Suppression Method

For event correlation, you can define one of three correlation methods:

- **Time interval:** This correlation method lets you define an interval during which duplicate events will be ignored. For more information, read this detailed example.

Time interval correlation example

In the illustration below, the interval is set to 30 seconds, but the suppression is limited to 60 seconds.

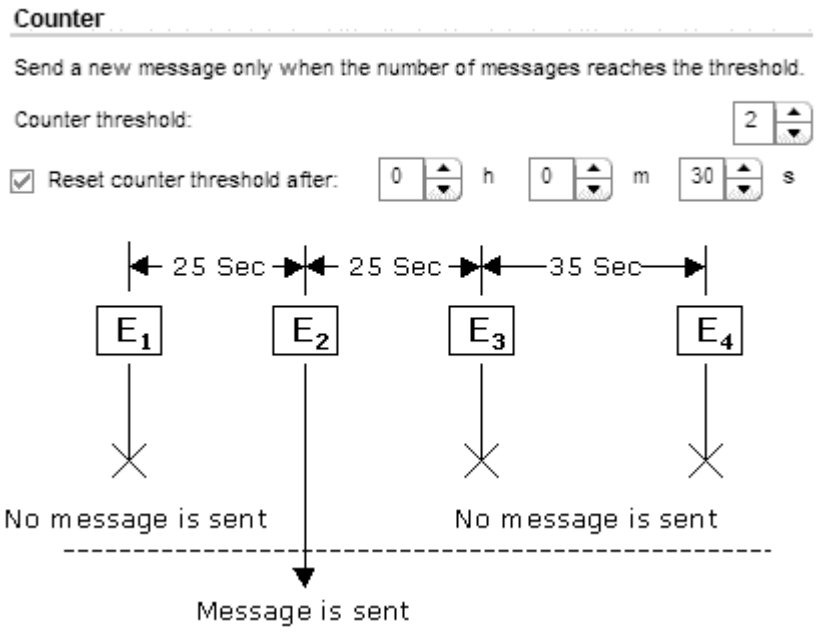


The E_x represents events that are identical.

- The first input event (E1) matches a rule in the policy. The policy sends an event and starts timing.
- A second matching event (E2) occurs 25 seconds later. This event occurred *less than 30* seconds after the first event, and is therefore suppressed.
- A third matching event (E3) occurs *less than 30 seconds after the second event*, and so is also suppressed.
- The next matching event (E4) occurs less than thirty seconds after the third event, but is also *more than 60 seconds after the first event*, and therefore the policy sends an event.

- **Counter:** This correlation method counts the number of matching input events and sends an event only after the number of matching input events equals the counter threshold. The counter can also be reset to zero after a time period that you specify. For more information, read this detailed example.

Counter correlation example



The E_x represent events that are identical.

- The first input event (E1) matches a rule in the policy, and the counter increments to one. No event is sent.
- A second matching event (E2) occurs, the counter increments to two, an event is sent, and the counter resets.
- A third matching event (E3), and the counter increments to one no event is sent.

- The next matching event (E4) occurs *more than thirty seconds* after the third event. Since at thirty seconds the counter was reset to zero, the counter now increments to one no event is sent.

- **Time interval/Counter** 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 an event to the Operations Management event browser.

Note: If you specify just time interval correlation or just counter-based correlation in an individual event, any eventdefaults for the other correlation method also apply. For example, if you specify time interval correlation for an event, and the event defaults specify counter-based correlation, the combined time interval and counter-based correlation applies to both new rules and existing rules.

You can change this default behavior, so that only the correlation method that you specify in the individual event applies. To change the default behavior, set the parameter `OPC_IGNORE_DEFAULT_MSG_CORRELATION=TRUE` in the `eaagt` namespace on the node. You can configure this parameter using `ovconfchg` or `ovconfpar` at a command prompt.

- Configure custom attributes

  Custom attributes are additional attributes that contain any information that is meaningful to you. For example, you might add a company name, contact information, or a city location to an event. You can have more than one CA attached to a single event. This attribute information displays in the Operations Management event browser in a column you have previously created to contain it.

  To create a new event custom attribute:

  i. Click * in the toolbar.

  ii. Type the name of the event custom attribute. The name is case-insensitive.

  iii. Type a value for the event custom attribute.

- Configure the message stream interface

  This tab lets you configure the interface between messages and external programs on the HP Operations agent.

  The message stream interface allows external applications to interact with the internal message flow of the HP Operations agent. The external application can be a read-write application, for example, a message processing program that can read HP Operations messages, modify attributes, and generate new messages for retransmission to the server. The application could also read messages, or send its own messages.

  When you enable the message stream interface, you can also allow external applications using the interface to set up automatic or operator-initiated commands.

  Select **Agent message stream interface** to allow messages to be directed to the **message stream interface** on the node. When switched on, you can choose between the following options:

  - Divert a message to the message stream interface instead of to the server when a message is requested by an external application.
  - Send the message to the server, and a copy of the message to the message stream interface.

- Configure OM attributes

  The OM attributes tab enables you to set additional attributes for a specific event (of for the event

defaults). These attributes are visible in the Additional Information tab of the Operations Management event browser and help the user to organize and evaluate the events.

- **Application**: Application that caused the event to occur. Unlike the Related CI attribute, which is a direct relationship to a CI in the ODB, the application attribute is a simple string-type attribute (for example, Oracle and OS).
- **Object**: Device such as a computer, printer, or modem. Unlike the Related CI attribute, which is a direct relationship to a CI in the ODB, the application attribute is a simple string-type attribute (for example, C:, and /dev/spool).
- **Type**: String used to organize different types of events within an event category or subcategory (for example, users or applications, accounts and security).
- **HPOM Service ID**: ID of the service associated the event. A service ID is a unique identifier for a service and can be used in BSM to identify the node and CI associated with the event.

- Add XML sample data

  If you are working with sample data, you can drag XML properties (XML elements and attributes) and values from the Sample Data tab and drop them onto the attribute boxes. Alternatively, you can type the path to the XML property or value directly into the attribute box.

  XML properties use the following syntax: <$DATA:/*<XML_property>*

  <XML_property> is the XML path, separated by slash marks (/), from the XML event tag to the XML element or attribute.

  BSM Integration Adapter replaces the XML property at runtime with the value of the specified XML element or attribute. If you insert an XML value, the value will be used.

  Note: The Sample Data tab is empty if no sample data has been loaded into the policy or if the sample data does not match any specified XML event tags.

  If sample data is available, then the XML Properties section of the Sample Data tab shows all XML elements and attributes that match an XML event tag. (You can identify attributes based on the preceding at sign (@).) The XML Properties section by default shows the short path to the XML property or value. To view the full path, click  . The full path begins with the XML event tag specified in the Source tab.

  The Values section displays the values of an XML property selected in the XML Properties section. If a value appears more than once, click  to show or hide duplicate values. To find values that belong to more than one XML property, select the value and click  . The XML Sample Data window opens and shows all XML properties that have the selected value.

- Add pattern matching variables

  Pattern matching variables insert matched strings that have previously been assigned to variables. To insert a pattern matching variable, type the variable name enclosed in angle brackets (for example, `<variablename>`) or drag and drop it from the Pattern Matching Variables list to the event attribute.

- Add mappings

  Mappings are custom variables that you define in the mappings tab. To insert a custom variable, you can drag it from the Mappings list and drop it on the event attribute.

  Alternatively, type the custom variable into the attribute box using the following syntax:

<$MAP(<*custom_variable*>)> where <*custom_variable*> is the map name of the variable (for example, `<$MAP(map@Severity)>`).

If the custom variable does not have an XML property assigned, use the following syntax:

<$MAP(<*custom_variable*>,<<*source_value*>>) where <*source_value*> can be one of the following:

- XML path to the source value, for example `<$MAP(map@Severity, <$DATA/SCOM_Alert/Severity>)>`
- The source value itself, for example `<$MAP(map@Severity, Warning)>`

- Add policy variables

  You can use policy variables in event attributes. BSM Integration Adapter replaces the variables with the appropriate values in the generated event.

  It is often useful to surround the variable with quotation marks, especially if it may return a value that contains spaces.

  `<$LOGFILE>`
  Returns the name of the log file that contains the input event. Sample output:`program_log.txt`

  `<$LOGPATH>`
  Returns the name and path of the log file that contains the input event. Sample output:`C:\temp\mylogfile\program_log.txt`

  `<$MSG_NODE>`
  Returns the IP address of the node on which the original event took place. Sample output: `192.168.1.123`

  `<$MSG_NODE_NAME>`
  Returns the name of the node on which the original event took place. This is the hostname that the agent resolves for the node. This variable is not fixed, however, and can be changed by a policy on a per-message basis. For example, if the policy is intercepting SNMP traps that originate from other devices, you might want to set this variable to the name of the device where the trap originated. If the policy is monitoring a log file on a network share where applications on several nodes write messages, you could extract the name of the node from the error message, save it in a user-defined variable, and assign it to MSG_NODE_NAME.

  `<$MSG_TEXT>`
  Returns the full text of the event. In general, there are default texts for all editors derived from incoming event properties. Sample output: `SU 03/19 16:13 + ttyp7 bill-root`

5. Click in the toolbar to save the policy.
6. In the BSM Integration Adapter user interface, click in the toolbar to update the list of policies.

Configure XML log file policy options

The options tab enables you to configure several policy behaviors.

To configure options in XML log file policies

1. In the BSM Integration Adapter user interface, click * in the toolbar, then click **XML**. The XML log file policy editor opens.

   Alternatively, double-click an existing XML log file policy to edit it.

2. Click **Options**. Options include:

   - Log local events

     BSM Integration Adapter allows you to define which events, if any, are logged on the node from which they originated. These events are logged on the local node in the log file: `<data_dir>\log\OpC\opcmsglg`.

     Three logging options are available.

     - Log local events **that match a rule and trigger a message**. This selection logs any events in the event source that match the policy rules.
     - Log local events **that match a rule and are ignored**. This selection logs any events in the event source that are suppressed (that is, they do not cause an event to be sent to the Operations Management event browser).
     - Log local events **that don't match any rule**. This selection logs any events that do not match any of the rules in the policy.

   - Capture unmatched events

     You can configure a policy to send an event to the Operations Management event browser when an event does not match any rule in the policy because none of the conditions apply or because the policy does not contain any rules. This ensures that unexpected events that might be important do not go unreported. By default, unmatched events are ignored.

     Each policy that sends unmatched events to the Operations Management event browser creates an event with the default values of the policy.

     Tip: If you want a policy to send events only with the default values, omit all rules from the policy.

     The following options are available:

     - Unmatched events **are sent to the messages browser**
     - Unmatched events **are sent to the acknowledged messages browser**
     - Unmatched events **are ignored** (default)

     If several policies forward unmatched events to the Operations Management event browser you could receive multiple events about a single input event.

   - Pattern matching options

     The following pattern matching options are available:

     - **Case sensitivity**

       You can choose whether the case (uppercase or lowercase) of a text string is considered when the pattern of a rule is compared with the event. When switched on, a match only occurs if the use of uppercase and lowercase letters is exactly the same in both the event and the pattern. This is the default setting.

- **Field separators**

  You can indicate which characters should be considered to be field separators. Field separators are used in the pattern as separator characters for the rule condition. You can define up to seven separators, including these special characters:

  - \n new line (NL)
  - \r carriage return (CR)
  - \t horizontal tab (HT)
  - \f form feed (FF)
  - \v vertical tab (VT)
  - \a alert (BEL)
  - \b backspace (BS)
  - \\ backslash (\)

  For example, if you wanted a backslash, an asterisk, and the letter A to define the fields in the event, you would type ***A** (with no spaces separating the characters).

  If you leave this box empty, the default separators (a blank and the tab character) are used by default.

If you change the pattern matching options, they apply to all new rules in a policy. Click **Apply to all** to apply them to all existing rules in a policy. This overwrites any modifications made to the pattern matching options in individual rules.

You can set case sensitivity and separator characters for individual rules in a policy by clicking the  button in the **Condition** tab of a rule. Pattern matching options can also be set in the **Event Correlation** tab for the **Close Events with Key** field.

3. Click  in the toolbar to save the policy.
4. In the BSM Integration Adapter user interface, click  in the toolbar to update the list of policies.

Pattern matching

To make your policies as flexible as possible, you can use pattern-matching syntax. The pattern-matching syntax makes it possible to write rule conditions that match strings very specifically.

Pattern-matching details

BSM Integration Adapter provides a powerful pattern-matching language that reduces the number of conditions you must use. Selected, dynamic parts of text-based events can be extracted, assigned to variables, and used as parameters to build the event description or to set other attributes.

The pattern-matching language enables you to very accurately specify the character string that you want a rule to match.

Matching special characters

Ordinary characters are expressions which represent themselves. Any character of the supported character set may be used. However, if any of the following special characters are used they must be prefaced with a backslash (\) that masks their usual function.

```
\ [ ] < > | ^ $
```

If ^ and $ are not used as anchoring characters, that is, not as first or last characters, they are considered ordinary characters and do not need to be masked.

Matching characters at the beginning or end of a line

If the caret (^) is used as the first character of the pattern, only expressions discovered at the beginning of lines are matched. For example, "^ab" matches the string "ab" in the line "abcde", but not in the line "xabcde".

If the dollar sign is used as the last character of a pattern, only expressions at the end of lines are matched. For example, "de$" matches "de" in the line "abcde", but not in the line "abcdex".

Matching multiple characters

Patterns used to match strings consisting of an arbitrary number of characters require one or more of the following expressions:

- <*> matches any string of zero or more arbitrary characters (including separators)
- <*n**> matches a string of *n* arbitrary characters (including separators)
- <#> matches a sequence of one or more digits
- <*n*#> matches a number composed of *n* digits
- <_> matches a sequence of one or more field separators
- <*n*_> matches a string of *n* separators
- <@> matches any string that contains no separator characters, in other words, a sequence of one or more non-separators; this can be used for matching words
- </> matches one or more line breaks
- <*n*/> matches exactly *n* line breaks

Separator characters are configurable for each pattern. By default, separators are the space and the tab characters.

Matching two or more different expressions

Two expressions separated by the special character vertical bar (|) matches a string that is matched by either expression. For example, the pattern:

```
[ab|c]d
```

matches the string "abd" and the string "cd".

Matching text that does not contain an expression

The NOT **operator** (!) must be used with delimiting square brackets, for example:

```
<![WARNING]>
```

The pattern above matches all text which does not contain the string "WARNING".

The **NOT operator** may also be used with complex subpatterns:

```
SU <*> + <@.tty> <![root|[user[1|2]]].from>-<*.ot>
```

The above pattern makes it possible to generate a "switch user" event for anyone who is not user1, user2 or root. Therefore the following would be matched:

```
SU 03/25 08:14 + ttyp2 user11-root
```

However, this line would not be matched, because it contains an entry concerning "user2":

```
SU 03/25 08:14 + ttyp2 user2-root
```

Notice that if the subpattern including the **not operator** does not find a match, the **not operator**behaves like a <*>: it matches zero or more arbitrary characters. For this reason, the pattern-matching expression: <![1|2|3]> matches any character or any number of characters, except 1, 2, or 3.

Mask (\ ) Operator

The backslash (\ ) is used to mask the special meaning of the characters:

```
[ ] < > | ^ $
```

A special character preceded by \ results in an expression that matches the special character itself.

Notice that because ^ and $ only have special meaning when placed at the beginning and end of a pattern respectively, you do not need to mask them when they are used within the pattern (in other words, not at beginning or end).

The only exception to this rule is the tab character, which is specified by entering "\t" into the pattern string.

Bracket ([and]) Expressions

The brackets ([and]) are used as delimiters to group expressions. To increase performance, brackets should be avoided wherever they are unnecessary. In the pattern:

```
ab[cd[ef]gh]
```

all brackets are unnecessary--"abcdefgh" is equivalent.

Bracketed expressions are used frequently with the **OR operator**, the **NOT operator** and when using **subpatterns** to assign strings to variables.

Numeric range operators

BSM Integration Adapter provides six numeric range operators that can be used in pattern matching. The operators are used in this way:

Operator name	Syntax	Example/Explanation
Less than	`<[pattern] -lt n>`	`<[<#>] -lt 5>` matches every number less than 5
Less than or equal to	**<[***pattern***] -le** *n* **>**	`<[<#>] -le 5>` matches 5 and every number less than 5
Greater than	**<[***pattern***] -gt** *n* **>**	`<[<#>] -gt 5>` matches every number greater than 5
Greater than or equal to	**<[***pattern***] -ge** *n* **>**	`<[<#>] -ge 5>` matches 5 and every number greater than 5
Equal to	**<[***pattern***] -eq** *n* **>**	`<[<#>] -eq 5>` matches 5 or 5.0
Not equal to	**<[***pattern***] -ne** *n* **>**	`<[<#>] -ne 5>` matches every number but 5 and 5.0
The operators can also be combined to produce matches according to ranges of numbers:		
Matches numbers that belong to the interval, excluding the limits	**<** *n* **-lt [***pattern***] -lt** *n* **>**	`<5 -lt [<#>] -lt 10>` matches every number between 5 and 10 (but not 5 or 10)
Matches numbers that belong to the interval, including the limits	**<** *n* **-le [***pattern***] -le** *n* **>**	`<5 -le [<#>] -le 10>` matches every number between 5 and 10 (including 5 and 10)
Matches numbers that do not belong to the interval, excluding the limits	**<** *n* **-gt [***pattern***] -gt** *n* **>**	`<10 -gt [<#>] -gt 5>` matches every number between 5 and 10 (but not 5 or 10)
Matches numbers that do not belong to the interval, including the limits	**<** *n* **-ge [***pattern***] -ge** *n* **>**	`<10 -ge [<#>] -ge 5>` matches every number between 5

		and 10 (including 5 and 10)

User-defined variables in patterns

Any matched string can be assigned to a variable, which can be used to compose events. To define a parameter, add ". `parametername` " before the closing bracket. The pattern:

```
^errno: <#.number> - <*.error_text>
```

matches an event such as:

```
errno: 125 - device does not exist
```

and assigns "125" to **number** and "device does not exist" to **error_text**.

When using these variables, the syntax is `<variable_name>` (for example, `<number>`).

Rules by which BSM Integration Adapter assigns strings to variables

In matching the pattern <***.var1**><***.var2**> against the string "abcdef", it is not immediately clear which substring of the input string will be assigned to each variable. For example, it is possible to assign an empty string to **var1** and the whole input string to **var2**, as well as assigning "a" to **var1**and "bcdef" to **var2**, and so forth.

The pattern matching algorithm always scans both the input line and the pattern definition (including alternative expressions) from left to right. <*> expressions are assigned as few characters as possible. <#>, <@>, `<s>` expressions are assigned as many characters as possible. Therefore, **var1** will be assigned an empty string in the example above.

To match an input string such as:

```
this is error 100: big bug
```

use a pattern such as:

```
error<#.errnumber>:<*.errtext>
```

In which:

- "100" is assigned to **errnumber**
- "big bug" is assigned to **errtext**

For performance and pattern readability purposes, you can specify a delimiting substring between two expressions. In the above example, ":" is used to delimit <#> and <*>.

Matching <@.**word**><#.**num**> against "abc123" assigns "abc12" to **word** and "3" to **num**, as digits are permitted for both <#> and <@>, and the left expression takes as many characters as possible.

Patterns without expression anchoring can match any substring within the input line. Therefore, patterns such as:

```
this is number<#.num>
```

are treated in the same way as:

```
<*>this is number<#.num><*>
```

Using subpatterns to assign strings to variables

In addition to being able to use a single operator, such as ***** or **#**, to assign a string to a variable, you can also build up a complex subpattern composed of a number of operators, according to the following pattern:
<[*subpattern*].**var>**

For instance: <[<@>file.tmp].**fname>**

In the example above, the period (.) between "file" and "tmp" matches a similar dot character, while the dot between "]" and "**fname**" is necessary syntax. This pattern would match a string such as "Logfile.tmp" and assigns the complete string to **fname**.

Other examples of subpatterns are:

- <[Error|Warning].**sev**>
- <[Error[<#.n><*.msg>]].**complete**>

In the first example above, any line with either the word "Error" or the word "Warning" is assigned to the variable, **sev**. In the second example, any line containing the word "Error" has the error number assigned to the variable, **n**, and any further text assigned to **msg**. Finally, both number and text are assigned to **complete**.

Pattern matching for variables

BSM Integration Adapter enables you to test a string or variable against a pattern, and define an output string that is conditional on the result. You can do this using **`$MATCH`**, which has the following syntax:

```
$MATCH(string, pattern, true, [false])
```

Specify the parameters as follows:

`string`
Specify a literal string (for example, `TEST STRING`) or an BSM Integration Adapter variable (for example `<$LOGPATH>`).

`pattern`
Specify a pattern, using BSM Integration Adapter pattern matching syntax. You can create user-defined variables in the pattern to use in the parameters `true` and `false`. The pattern is case sensitive.

`true`
Specify a string to return if the string and pattern match. You can specify a literal string, or a user-defined variable, or an BSM Integration Adapter variable.

`false`
Optional. Specify a string to return if the string and pattern do not match. You can specify a literal string, or a user-defined variable, or an BSM Integration Adapter variable.

Separate each parameter with a comma (,). To specify a comma within a parameter, you must precede it with two backslashes (\\).

You can use `$MATCH` within your policies in the following event attributes:

- Service ID
- Message type
- Message group
- Application
- Object

- Message text
- Automatic command
- Custom message attribute

Note: You can use $MATCH only once in each message attribute. You cannot use $MATCH recursively.

Example

An XML logfile entry policy can monitor a number of log files. The name of path of the log file is available in the BSM Integration Adaptervariable `<$LOGPATH>`. If part of the log file path corresponds to an application name, you can use $MATCH to set the application event attribute as follows:

```
$MATCH(<$LOGPATH>,<@.application>.log, <application>, Unknown)
```

Examples of pattern matching in rule conditions

The following examples show some of the many ways in which the pattern-matching language can be used.

- `Error`

  Recognizes any event containing the keyword `Error` at any place in the event. (It is case sensitive by default.)

- `panic`

  Matches all events containing `panic`, `Panic`, `PANIC` anywhere in the text of the event, when case sensitive mode is switched off.

- `logon|logoff`

  Uses the **OR operator** to recognize any event containing the keyword `logon`or `logoff`.

- `^getty:<*.msg> errno<*><#.errnum>$`

  Recognizes any event such as: `getty: cannot open ttyxx errno : 6`or `getty: can't open ttyop3; errno 16`

  In the example `getty: cannot open ttyxx errno : 6`, the string "cannot open ttyxx" is assigned to the variable **msg**. The digit 6 is assigned to the variable **errnum**. Note that the dollar sign ($) is used as an anchoring symbol to specify that the digit 6 will only be matched if it is at the end of the line.

- `^errno[ |=]<#.errnum> <*.errtext>`

  Matches events such as: `errno 6 - no such device or address`or `errno=12 not enough core`.

  Note the space before the **OR operator**. The expression in square brackets matches either this blank space, or the "equals" sign. The space between **<#.errnum>** and **<*.errtext>** is used as a delimiter. Although not strictly required for assignments to the variables shown here, this space serves to increase performance.

- `^hugo:<*>:<*.uid>:`

  Matches any **/etc/passwd** entry for user `hugo` and returns the user ID to variable **uid**. Notice that ":" in the middle of the pattern is used to delimit the string passed to **uid** from the preceding string. The colon ":" at the end of the pattern is used to delimit the string passed to **uid** from the succeeding group ID in

the input pattern. Here, the colon is necessary not only as a speed enhancement, but also as a means of logical separation between strings.

- `^Warning:<*.text>on node<@.node>$`

  Matches any event such as: `Warning: too many users on node hpbbx` and assigns `too many users` to **text**, and `hpbbx`to **node**.

- `^<*.line1><1/><*.line2><1/><*.line3><1/><*.line4>$`

  Matches four lines of text, for example:

  ```
  Security ID:     S-1-5-21-3358208617-1210941181-189752109-500
  Account Name:    Administrator
  Account Domain:  EXAMPLE
  Logon ID:        0x228a2
  ```

  There is one line break between each line. The pattern assigns each line of text to a variable.

- `<<#> -le 45>`

  This pattern matches all strings containing a number which is less than or equal to 45. For example, the event: *ATTENTION: Error 40 has occurred* would be matched.

  Note that the number 45 in the pattern is a true numeric value and not a string. Numbers higher than 45, for instance, "4545" will not be matched even if they contain the combination, "45".

- `<15 -lt <2#> -le 87>`

  This pattern matches any event in which the first two digits of a number are within the range 16-87. For instance, the event: *Error Message 3299* would be matched. The string: *Error Message 9932* would not be matched.

- `^ERROR_<[<#.err>] -le 57>`

  This pattern matches any text starting with the string "ERROR_" immediately followed by a number less than, or equal to, 57.

  For example, the event: *ERROR_34: processing stopped* would be matched and the string 34 would be assigned to the variable, *err*.

- `<120 -gt [<#>1] -gt 20>`

  Matches all numbers between 21 and 119 which have 1 as their last digit. For instance, events containing the following numbers would be matched: 21, 31, 41... 101... 111 and so on.

- `Temperature <*> <@.plant>: <<#> -gt 100> F$`

  This pattern matches strings such as: "Actual Temperature in Building A: 128 F". The letter "A" would be assigned to the variable, *plant*.

- `Error <<#> -eq 1004>`

  This pattern matches any event containing the string "Error" followed by a space and the sequence of digits, "1004".

  For example, *Warning: Error 1004 has occurred*would be matched by this pattern. However, *Error 10041* would not be matched by this pattern.

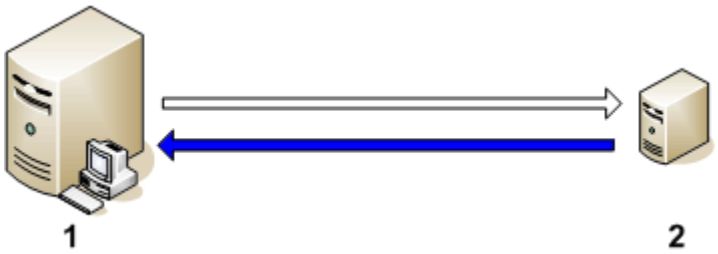
- `WARNING <<#> -ne 107>`

  This pattern matches any event containing the string "WARNING" followed by a space and any sequence of one or more digits, except "107". For example, the event: *Application Enterprise (94/12/45 14:03): WARNING 3877* would be matched.

Configuring HTTPS communication through firewalls

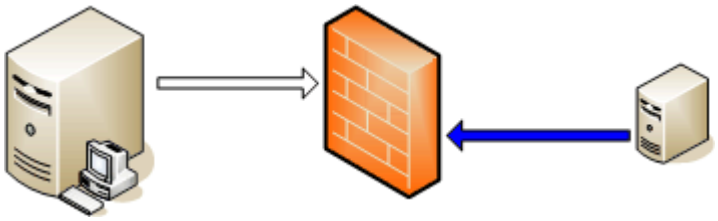
HP Business Service Management (BSM) and HP BSM Integration Adapter (BSM Integration Adapter) communicate with each other over the network. This communication uses the HTTPS protocol. The figure below shows the network connections between BSM and BSM Integration Adapter as follows:

- BSM **(1)** opens connections to BSM Integration Adapter **(2)**, for example to grant certificate requests.
- BSM Integration Adapter **(2)** opens connections to BSM **(1)**, for example to send events.



When BSM or BSM Integration Adapter opens a new connection, the operating system allocates the local port for the connection. On the other side of the connection, BSM and BSM Integration Adapter both have communication brokers, which listen on port 383 for incoming connections. So by default, all connections have a local port assigned by the operating system and the destination port is 383.

If BSM and BSM Integration Adapter are on different networks that are separated by a firewall, the firewall may block connections between them, as the figure below shows.This prevents you from receiving events in the Operations Management event browser, because, for example, BSM Integration Adapter cannot send them.



If a firewall blocks HTTPS connections, you can reconfigure communication between BSM and BSM Integration Adapter in several ways. The configuration you choose to implement depends mainly on the configuration of your network.

- If your network allows HTTPS connections through the firewall in both directions, but with certain restrictions, the following configuration options are possible to accommodate these restrictions:
  - If your network allows only certain proxy systems to open connections through the firewall, you can redirect communication through these proxies.
  - If your network allows inbound connections to only certain destination ports, but not to port 383, you can configure alternate communication broker ports.
  - If your network allows outbound connections from only certain local ports, you can configure BSM and BSM Integration Adapter to use specific local ports.
- If your network allows only outbound HTTPS connections from BSM across the firewall, and blocks inbound connections from BSM Integration Adapter, you can configure a reverse channel proxy.

Configuring two-way communication

If your network allows HTTPS connections through the firewall in both directions, but with certain restrictions, the following configuration options are possible to accommodate these restrictions:

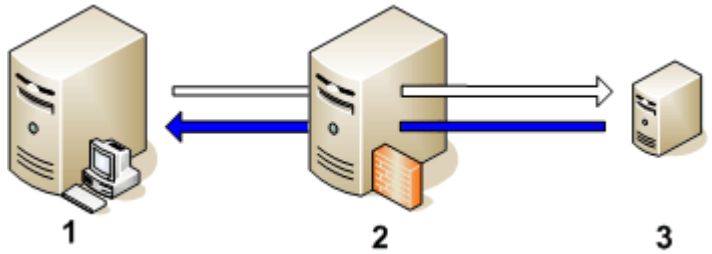
- If your network allows only certain proxy systems to open connections through the firewall, you can redirect communication through these proxies.
- If your network allows inbound connections to only certain destination ports, but not to port 383, you can configure alternate communication broker ports.
- If your network allows outbound connections from only certain local ports, you can configure BSM and BSM Integration Adapter to use specific local ports.

Redirect HTTPS communication through proxies

Overview

You can redirect connections from BSM and BSM Integration Adapter that are on different networks through a proxy. The figure below shows connections between BSM and BSM Integration Adapter through a proxy as follows:

- BSM **(1)** opens connections to the proxy **(2)**, for example to grant certificate requests. The proxy opens connections to BSM Integration Adapter **(3)** on behalf of BSM, and forwards communication between them.
- BSM Integration Adapter **(3)** opens connections to the proxy **(2)**, for example to send events. The proxy opens connections to BSM **(1)** on behalf of BSM Integration Adapter.



You can also redirect communication through proxies in more complex environments as follows:

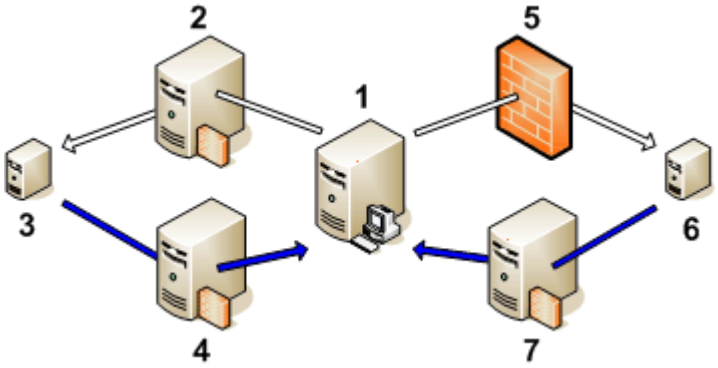
- Each BSM and BSM Integration Adapter can use a different proxy server to communicate with each other.
- You can configure BSM and BSM Integration Adapter to select the correct proxy according to the host they need to connect to.

The figure below shows connections between BSM and BSM Integration Adapter through multiple proxies as follows:

- BSM **(1)** opens connections to a proxy **(2)**. The proxy opens connections to BSM Integration Adapter **(3)** on behalf of BSM.
- BSM Integration Adapter **(3)** opens connections to a different proxy **(4)**. The proxy opens connections to BSM **(1)** on behalf of BSM Integration Adapter.
- The network allows BSM **(1)** to make outbound HTTP connections directly through the firewall **(5)** to

another BSM Integration Adapter **(6)**. (The BSM Integration Adapter servers **(3, 6)** are on different networks.)

- The firewall **(5)** does not allow inbound HTTP connections. Therefore, BSM Integration Adapter **(6)** opens connections to BSM through a proxy **(7)**.



PROXY parameter syntax

You redirect outbound HTTPS communication through proxies by setting the **PROXY** parameter in the **bbc.http** name space on BSM and BSM Integration Adapter. You can configure this parameter using `ovconfchg`.

The value of the **PROXY** parameter can contain one or more proxy definitions. Specify each proxy in the following format:

<proxy_hostname>:<proxy_port>+(<included hosts>)-(<excluded hosts>)

Replace *<included_hosts>* with a comma-separated list of hostnames or IP addresses to which the proxy enables communication. Replace *<excluded hosts>* with a comma-separated list of hostnames or IP addresses to which the proxy cannot connect. Asterisks (*) are wild cards in hostnames and IP addresses. Both *<included_hosts>* and *<excluded hosts>* are optional.

To specify multiple proxies, separate each proxy with a semicolon (;). The first suitable proxy in the list takes precedence.

Example PROXY parameter values

To configure BSM Integration Adapter to use `proxy1.example.com` port `8080` for all outbound connections, you would use the following value:

```
proxy1.example.com:8080
```

To configure BSM to use `proxy2.example.com:8080` to connect to any host with a hostname that matches `*.example.com` or `*example.org` except hosts with an IP address in the range 192.168.0.0 to 192.168.255.255, you would use the following value:

```
proxy2.example.com:8080+(*.example.com,*.example.org)-(192.168.*.*)
```

To extend the above example to use `proxy3.example.com` to connect to `backup.example.com` only, you would use the following value:

```
proxy3.example.com:8080+(backup.example.com);
proxy2.example.com:8080+(*.example.com,*.example.org)-(192.168.*.*)
```

In the above example, `proxy3.example.com:8080+(backup.example.com)` must be first, because the include list for `proxy2.example.com` contains `*.example.com`.

To redirect HTTPS communication through proxies using `ovconfchg`

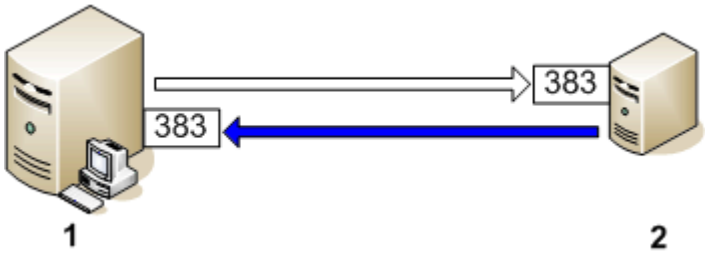
1. Log in to BSM or BSM Integration Adapter server as a user with administrative rights and open a command prompt or shell.
2. On systems that run a Linux operating system, ensure that the PATH variable contains the path to the agent commands.
   - Type **`export PATH=/opt/OV/bin:$PATH`** and then press **Enter**.
3. Specify the proxies that BSM or BSM Integration Adapter should use. You can specify different proxies to use depending on the host to connect to. Type the following command:

   `ovconfchg -ns bbc.http -set PROXY <proxy>`

Configure communication broker ports

Overview

BSM and BSM Integration Adapter both include communication brokers that listen for inbound connections on port 383, as the figure below shows. The communication broker on BSM **(1)** handles all inbound connections from BSM Integration Adapter servers. The communication broker on BSM Integration Adapter **(2)** handles all inbound connections from BSM.



You can configure any communication broker to listen on a port other than 383. If you do this, you must also configure the other BSM and BSM Integration Adapter servers in the environment, so that their outbound connections are destined for the correct port. For example, if you configure the BSM Integration Adaptercommunication broker to listen on port 5000, you must also configure the BSM so that it connects to port 5000 when it communicates with BSM Integration Adapter.

PORTS parameter syntax

You configure communication broker ports by setting the **`PORTS`** parameter in the **`bbc.cb.ports`** name space on all BSM and BSM Integration Adapter servers that communicate with each other. You can configure this parameter using `ovconfchg`.

The values must contain one or more host names or IP addresses and have the following format:

`<host>:<port>[,<host>:<port>] ...`

The `<host>` can be either a domain name or IP address. For example, to configure the communication broker port to **`5000`** on a BSM server with the host name `manager1.emea.example.com`, use the

itself, and also any other BSM and BSM Integration Adapter servers that open connections to it:

```
ovconfchg -ns bbc.cb.ports -set PORTS manager1.emea.example.com:5000
```

If you need to configure communication broker ports on multiple systems, you can use wildcards and ranges, as follows:

- You use a wildcard at the start of a domain name by adding an asterisk (*). For example:
  - `*.emea.example.com:5000`
  - `*.test.com:5001`
  - `*:5002`
- You can use wildcards at the end of an IP address by adding up to three asterisks (*). For example:
  - `192.168.1.*:5003`
  - `192.168.*.*:5004`
  - `10.*.*.*:5005`
- You can replace one octet in an IP address with a range. The range must be before any wildcards. For example:
  - `192.168.1.0-127:5006`
  - `172.16-31.*.*:5007`

If you specify multiple values for the **PORTS** parameter, separate each with a comma (,). For example:

```
ovconfchg -ns bbc.cb.ports -set PORTS *.emea.example.com:5000,10.*.*.*:5005
```

When you specify multiple values using wildcards and ranges that overlap, BSM orBSM Integration Adapter selects the port to use in the following order:

- Fully qualified domain names.
- Domain names with wildcards.
- Complete IP addresses.
- IP addresses with ranges.
- IP addresses with wildcards.

For example, if you configure communication broker ports on all BSM and BSM Integration Adapter servers with the following command:

```
ovconfchg -ns bbc.cb.ports -set PORTS
*.emea.example.com:6000,10.*.*.*:6001,manager1.emea.example.com:6002,10.0-
127.*.*:6003
```

the following ports are used:

- Host name: **`node1.asia.example.com`**
  IP address: **`10.127.1.1`**
  Communication broker port: **`6003`**.
- Host name: **`manager1.emea.example.com`**
  IP address: **`10.1.1.1`**
  Communication broker port: **`6002`**.

- Host name: **`node1.test.com`**
  IP address: **`192.168.1.1`**
  Communication broker port: **`383`**.

To find out which port is currently configured, type the following command:

`bbcutil -getcbport <host>`

Tip: To organize settings for many communication broker ports, you can add parameters of any name in the **`bbc.cb.ports`** name space. The value of any parameter in the name space is evaluated. The **`PORTS`** parameter is optional.

To configure communication broker ports using `ovconfchg`

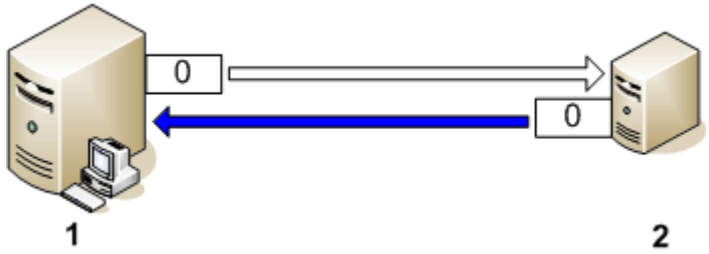
1. Log in to BSM or BSM Integration Adapter server as a user with administrative rights and open a command prompt or shell.
2. On systems that run a Linux operating system, ensure that the PATH variable contains the path to the agent commands.
   - Type **`export PATH=/opt/OV/bin:$PATH`** and then press **Enter**.
3. Specify the communication broker ports by typing the following command:

   `ovconfchg -ns bbc.cb.ports -set PORTS <host>:<port>[,<host>:<port>] ...`

Configure local communication ports

Overview

BSM servers open outbound connections to BSM Integration Adapter servers, and BSM Integration Adapter servers open outbound connections to BSM servers, as the figure below shows. BSM **(1)** opens these connections, for example, to grant certificates. BSM Integration Adapter **(2)** opens these connections, for example, to send events.



By default, BSM and BSM Integration Adapteruse local port 0 for outbound connections, which means that the operating system allocates the local port for each connection. Typically, the operating system will allocate local ports sequentially. For example, if the operating system allocated local port 5055 to an Internet browser, and then BSM Integration Adapter opens a connection, BSM Integration Adapter receives local port 5056.

However, if a firewall restricts the ports that you can use, you can configure BSM and BSM Integration Adapter to use a specific range of local ports instead.

CLIENT_PORT parameter syntax

You configure local communication ports by setting the **CLIENT_PORT** parameter in the **bbc.http** name space on the BSM and BSM Integration Adapter server. You can configure this parameter using ovconfchg.

The value must be a range of ports in the following format:

<lower port number>-<higher port number>

For example, if the firewall only allows outbound connections that originate from ports 5000 to 6000 you would use the following value:

```
5000-6000
```

To configure local communication ports using `ovconfchg`

1. Log in to BSM or BSM Integration Adapter server as a user with administrative rights and open a command prompt or shell.
2. On systems that run a Linux operating system, ensure that the PATH variable contains the path to the agent commands.
   - Type **`export PATH=/opt/OV/bin:$PATH`** and then press **Enter**.
3. Specify the range of local ports that BSM or BSM Integration Adapter can use for outbound connections by typing the following command:

   `ovconfchg -ns bbc.http -set CLIENT_PORT <lower port number>-<higher port number>`

Configure multihomed systems

By default, when an BSM Integration Adapter server has several IP addresses, the agent uses them as follows:

- The communication broker accepts incoming connections on all IP addresses.
- The agent opens connections to BSM using the first network interface that it finds.
- The embedded performance component accepts incoming connections on all IP addresses.

You can configure the agent to always use a specific IP address. You do this by configuring the parameters in the following table using `ovconfchg` at a command prompt:

Namespace	Parameter	Value
bbc.cb	SERVER_ BIND_ ADDR	The IP address that you want the communication broker to listen on for incoming connections.
bbc.http	CLIENT_ BIND_ ADDR	The IP address that you want the agent use on the BSM Integration Adapter end of connections to BSM.
coda.comm	SERVER_ BIND_ ADDR	The IP address that you want the embedded performance component to listen on for incoming connections. This can either be the same IP address as the SERVER_BIND_ADDR parameter in the bbc.cb namespace, or you

		can set this value to localhost. If you set the value to localhost, the embedded performance component accepts incoming connections only through the communication broker.

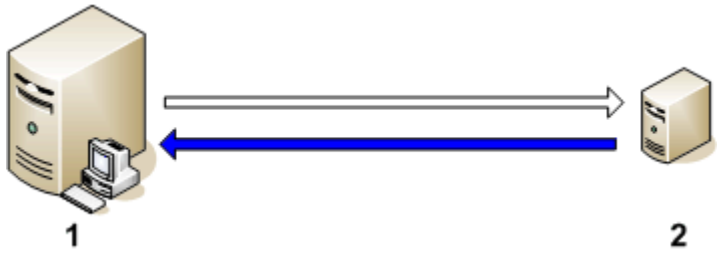
Configuring outbound-only communication

Overview

BSM and BSM Integration Adapter communicate with each other over the network. Normally, BSM servers open outbound network connections to BSM Integration Adapter servers and BSM Integration Adapter servers open inbound network connections to BSM servers.

The figure below shows the network connections where there is no firewall that blocks inbound HTTPS connections to BSM as follows:

- BSM **(1)** opens outbound connections to BSM Integration Adapter **(2)**, for example to grant certificate requests.
- BSM Integration Adapter **(2)** opens inbound connections to BSM **(1)**, for example to send events.



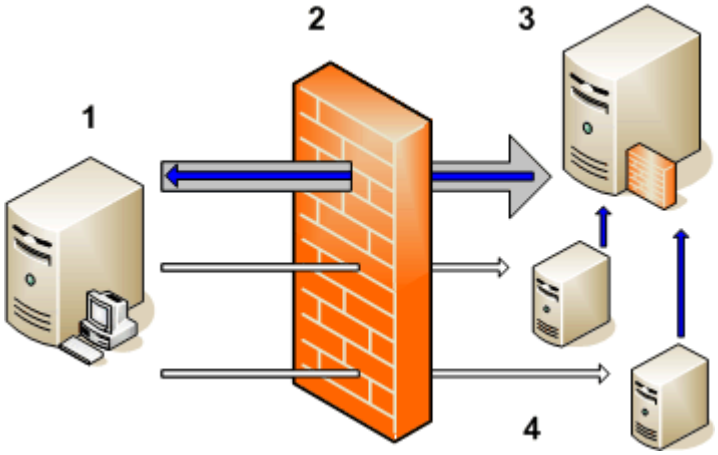
If a firewall blocks inbound HTTPS connections from BSM Integration Adapter to BSM, BSM Integration Adapter cannot communicate with BSM properly. To enable proper communication, you configure an BSM Integration Adapter to act as a reverse channel proxy (RCP).

An RCP handles communication between BSM and BSM Integration Adapter, so that they do not need to communicate with each other directly. An RCP can run on the BSM Integration Adapter that it serves, or on a separate system that serves multiple BSM Integration Adapter servers. The RCP is on the same side of the firewall as the BSM Integration Adapter servers that it serves.

Outbound-only communication through one firewall

The figure below shows the network connections where there is a firewall that blocks inbound HTTPS connections to BSM as follows:

- BSM **(1)** makes an outbound connection through the firewall **(2)** to an RCP **(3)**. This connection is called a reverse administration channel. BSM maintains the reverse administration channel, so that the RCP never needs to make an inbound connection to BSM.
- BSM Integration Adapter servers **(4)** open connections to the RCP, instead of BSM. The RCP **(3)** forwards BSM Integration Adapter communications to BSM using the reverse administration channel.
- BSM **(1)** also makes outbound connections directly to BSM Integration Adapter servers **(4)**.



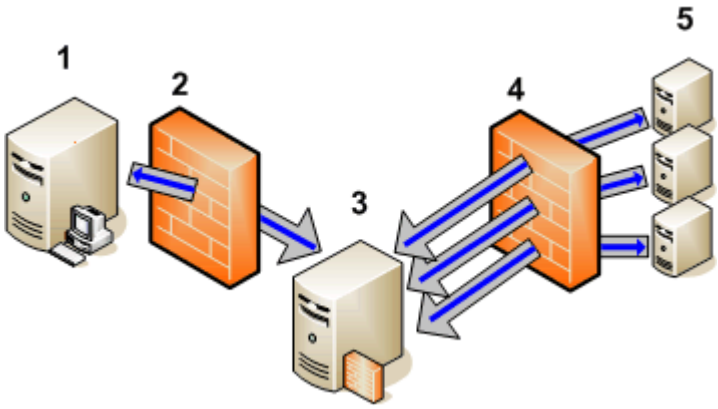
To configure outbound-only communication in this scenario, you must:

1. Configure the RCP, so that it listens for incoming connections.
2. Configure BSM, so that it opens the reverse administration channel to the RCP.
3. Configure the BSM Integration Adapter servers, so that they use the RCP for their outbound connections to the BSM.

Outbound-only communication through two firewalls

The figure below shows the network connections where there are two firewalls. One firewall blocks inbound connections to the BSM. The other firewall blocks inbound connections to the BSM Integration Adapter servers.

- BSM **(1)** opens a reverse administration channel through the firewall **(2)** to the RCP **(3)**. BSM maintains the reverse administration channel, so that the RCP never needs to make an inbound connection to BSM.
- Each BSM Integration Adapter**(5)** opens a reverse administration channel through the firewall **(4)** to the RCP **(3)**. BSM Integration Adapter serversmaintain these connections, so that the RCP never needs to make inbound connections to the BSM Integration Adapter servers.
- BSM **(1)** and BSM Integration Adapter servers **(5)** open outbound connections to the RCP, instead of directly to each other. The RCP **(3)** forwards the these communications using the reverse administration channel.



To configure outbound-only communication in this scenario, you must:

1. Configure the RCP, so that it listens for incoming connections.
2. Configure BSM, so that it opens a reverse administration channel to the RCP.
3. Configure BSM, so that it uses the RCP as a proxy for its outbound connections to BSM Integration Adapter servers.
4. Configure the BSM Integration Adapter servers, so that they each open a reverse administration channel to the RCP.
5. Configure the BSM Integration Adapter servers, so that they use the RCP for their outbound connections to BSM.

Configure a reverse channel proxy

Before you can configure a system as a reverse channel proxy (RCP), you must install BSM Integration Adapter. You must also configure certificates.

To configure a reverse channel proxy

1. Log in to the BSM Integration Adapter server as a user with administrative rights and open a command prompt or shell.
2. On systems that run a Linux operating system, ensure that the PATH variable contains the path to the agent commands.
   - Type **`export PATH=/opt/OV/bin:$PATH`** and then press **Enter**.
3. Set the port that BSM Integration Adapter servers and BSM servers can connect to. Type following command:

   `ovconfchg -ns bbc.rcp -set SERVER_PORT <port_number>`

   Note: Make sure that the port number you specify is not already in use by any other software on the system.
4. Register the RCP component so that `ovc` starts, stops and monitors it. Type the following commands:
   a. **`ovcreg -add <install_dir>/newconfig/DataDir/conf/bbc/ovbbcrcp.xml`**
   b. **`ovc -kill`**
   c. **`ovc -start`**

Note: After you configure the BSM server to establish a connection with this RCP you can check that the connection exists, by typing the following command:

`ovbbcrcp -status`
The command shows details of the reverse channel connection.

Configure reverse administration channels

To configure a reverse administration channel

1. Log in to BSM or BSM Integration Adapter server as a user with administrative rights and open a command prompt or shell.
2. Enable outbound-only communication. By default, this is disabled. To change this, type the following command:

   ovconfchg [-ovrg server] -ns bbc.cb -set ENABLE_REVERSE_ADMIN_CHANNELS true
3. Specify the reverse channel proxies (RCPs) that to which you want to open reverse administration channels. You must specify RCPs in the following format:

   `<host>:<port>[,<OvCoreID>]`

   For example, if BSM must connect to port 50000 on `rcp1.example.com` you specify the RCP with:

   `rcp1.example.com:50000,9fcc7062-0472-751c-1236-84372bec342d`

   If you specify the optional OvCoreID, BSM checks that the RCP has that OvCoreID. You can specify the RCPs at the command prompt or in a file:

   - To specify the RCPs at the command prompt, type the following command:

     ovconfchg [-ovrg server] -ns bbc.cb -set RC_CHANNELS <*rcp*>[;<*rcp*>]

     Separate each RCP with a semicolon (;).
   - To specify the RCPs in a file:
     i. Create a text file that specifies each RCP on a separate line.
     ii. *Optional.* Add comments on lines that begin with the number sign (#).
     iii. Save the file in the folder `<data_dir>\conf\bbc`.
     iv. Type the following command:

     ovconfchg [-ovrg server] -ns bbc.cb -set RC_CHANNELS_CFG_FILES <*file name*>

     You must also type the same command if you later change the contents of the file. BSM reads the file only after you use `ovconfchg`.
4. *Optional.* Configure whether BSM should automatically retry failed reverse administration channel connections. By default, the server does not retry failed connections. To change the default, type the following command:

   ovconfchg [-ovrg server] -ns bbc.cb -set RETRY_RC_FAILED_CONNECTION TRUE
5. *Optional.* Set the maximum number of attempts that BSM should make to reconnect to a failed reverse administration channel connection. By default, this is set to -1 (infinite). To change the default, type the following command:

   ovconfchg [-ovrg server] -ns bbc.cb -set MAX_RECONNECT_TRIES <*number of tries*>
6. *Optional.* Configure BSM to generate a warning message about failed reverse administration

channel connections. By default, BSM does not generate this message. To change the default, type the following command:

ovconfchg [-ovrg server] -ns bbc.cb -set GENERATE_OVEVENT_FOR_FAILED_RC_NODES TRUE

However, if you set `RETRY_RC_FAILED_CONNECTION` to `TRUE`, BSM attempts to reconnect the failed connection without generating the message.

7. *Optional.* Configure the number of minimum and maximum number of worker threads for connections to RCPs. The communication broker can use multiple worker threads to enhance the performance of connections to RCPs.

   By default, the maximum number of worker threads is 1 and the minimum number of worker threads is 0. If the system has sufficient resources, you can increase the number of worker threads. To change the defaults, type the following commands:

   ovconfchg [-ovrg server] -ns bbc.cb -set RC_MAX_WORKER_THREADS <*number*>

   ovconfchg [-ovrg server] -ns bbc.cb -set RC_MIN_WORKER_THREADS <*number*>

8. *Optional.* To check that the reverse administration channel is open, type the following command:

   ovbbccb -status

   The output lists all open reverse administration channels. If any of the reverse administration channel connections has the state `FAILED`, you can attempt to restore the connections by typing the following command:

   ovbbccb -retryfailedrcp [-ovrg server]

Forward outbound connections through a reverse channel proxy

To forward outbound connections through a reverse channel proxy

1. Log in to BSM or BSM Integration Adapter server as a user with administrative rights and open a command prompt or shell.
2. On systems that run a Linux operating system, ensure that the PATH variable contains the path to the agent commands.
   - Type **export PATH=/opt/OV/bin:$PATH** and then press **Enter**.
3. Specify the RCP to use for outbound connections. You can specify different RCPs to use depending on the destination host. Type the following command:

   ovconfchg -ns bbc.http -set PROXY <*rcp*>[;<*rcp*>]

   Separate each RCP with a semicolon. Specify each <*rcp*> in the following format:

   <*rcp_hostname*>:<*rcp_port*>+(<*included hosts*>)-(<*excluded hosts*>)

   Replace <*included_hosts*> with a comma-separated list of valid destination hostnames or IP addresses for the RCP. Replace <*excluded hosts*> with a comma-separated list of hostnames or IP addresses that system should not use the RCP to connect to. Asterisks (*) are wild cards in hostnames and IP addresses.

   Note: <*excluded_hosts*> must always contain the hostname and the fully qualified domain name of the RCP.

For example, to configure BSM Integration Adapter to use `rcp1.example.com:50000` to connect to any host with a hostname that matches `*.example.com` or `*example.org` except hosts with an IP address in the range 192.168.0.0 to 192.168.255.255, you would type the following command:

```
 ovconfchg -ns bbc.http -set PROXY
rcp1.example.com:50000+(*.example.com,*.example.org)-
(192.168.*.*,rcp1.example.com,rcp1)
```

4. *Optional.* For BSM Integration Adapter, specify the OvCoreID of the BSM server that the RCP should connect this BSM Integration Adapter to. This is useful if the RCP cannot resolve the hostnames of BSM because of firewalls. When BSM Integration Adapter attempts to open a connection to a BSM server, the RCP can use the OvCoreID instead of the hostname to select the correct reverse administration channel. You can either specify the BSM server's OvCoreID directly, or specify a command that returns the OvCoreID.
   - To specify the BSM server's OvCoreID directly, type the following command:

     ovconfchg -ns bbc.http -set TARGET_FOR_RC <*server OvCoreID*>
   - To specify a command that returns the BSM server's OvCoreID, type the following command:

     ovconfchg -ns bbc.http -set TARGET_FOR_RC_CMD <*command*>
5. To restart the message agent, type **`ovc -restart opcmsga`** and then press **Enter**.

Agent Command-Line Utilities

These utilities can be executed on the HP Operations agent version 8.60 and lower.

Note: The information in this *HPOM Agent Application Integration Guide* is only for version 8.60 and lower of HP Operations agent.

bbc.ini

NAME

bbc.ini

- Configuration file for HTTPS communication.

DESCRIPTION

`bbc.ini` is the configuration file of a node using HTTPS communication and is located at:

`/<OvInstallDir>/misc\xpl\config\defaults`

It consists of sections headed by namespaces which contain the settings for each namespace. The `bbc.ini` file contains the namespaces listed below. Possible and default settings are described for each namespace.

bbc.cb

The Communication-Broker Namespace. You can use the following parameters:

`string CHROOT_PATH = <path>`
On UNIX systems only, the `chroot` path is used by the `ovbbccb` process. If this parameter is set, the `ovbbccb`process uses this path as the effective root thus restricting access to a limited part of the file system. Default is `<OvDataDir>`. This parameter is ignored on MS Windows and Sun Solaris 7 systems. See the `chroot` man page for details on `chroot`.

`bool SSL_REQUIRED = false`
If this parameter is set to `true`, the communication broker requires SSL authentication for all administration connections to the communication broker. If this parameter is set to `false`, non-SSL connections are allowed to the communication broker.

`bool LOCAL_CONTROL_ONLY = false`
If this parameter is set to `true`, the communication broker only allows local connections to execute administrative commands such as `start` and `stop`.

`bool LOG_SERVER_ACCESS = false`
If this parameter is set to `true`, every access to the server is logged providing information about the sender's IP address, requested HTTP address, requested HTTP method, and response status.

`int SERVER_PORT = 383`
By default this port is set to `383`. This is the port used by the communication broker to listen for requests. If a port is set in the namespace `[bbc.cb.ports]`, it takes precedence over this parameter.

`string SERVER_BIND_ADDR = <address>`
Bind address for the server port. Default is `INADDR_ANY`.

bbc.cb.ports

The Communication-Broker-Port Namespace. This parameter defines the list of ports for all Communications Brokers in the network that may be contacted by applications on this host. The default port number for all BBC CBs is 383. You can use the following parameters:

`string PORTS`

This configuration parameter must be the same on all nodes. To change the port number of a BBC CB on a particular host, the hostname must be added to this parameter, for example, `name.hp.com:8000`. You can use an asterisk "`*`" as a wild card to denote an entire network, for example; `*.hp.com:8001`. Note too, that either a comma "," or a semicolon ";" should be used to separate entries in a list of hostnames, for example;

`name.hp.com:8000,*.hp.com:8001.`

In these examples, all hostnames ending in "hp.com" will configure their BBC Communication Broker to use port 8001 except host "name" which will use port 8000. All other hosts use the default port 383. You can also use IP addresses and the asterisk wild card (*) to specify hosts. For example; `15.0.0.1:8002,15.*.*.*:8003`

bbc.http

The HTTP Namespace for node-specific configuration. For application-specific settings, see the section bbc.http.ext.*. Note that application-specific settings in `bbc.http.ext.*` override node-specific settings in `bbc.http`. You can use the following parameters:

`int SERVER_PORT = 0`
By default this port is set to `0`. If set to `0`, the operating system assigns the first available port number. This is the port used by the application *<app_Name>* to listen for requests. Note that it only really makes sense to explicitly set this parameter in the `bbc.http.ext.<app_Name>` namespace, as the parameter is application specific with any other value than the default value.

`string SERVER_BIND_ADDR = <address>`
Bind address for the server port. Default is `localhost`.

`string CLIENT_PORT` = 0
Bind port for client requests. This may also be a range of ports, for example 10000-10020. This is the bind port on the originating side of a request. Default is port `0`. The operating system will assign the first available port.
Note that MS Windows systems do not immediately release ports for reuse. Therefore on MS Windows systems, this parameter should be a large range.

`string CLIENT_BIND_ADDR = <address>`
Bind address for the client port. Default is `INADDR_ANY`.

`bool LOG_SERVER_ACCESS = false`
If this parameter is set to `true`, every access to the server is logged providing information about the sender's IP address, requested HTTP address, requested HTTP method, and response status.

`string PROXY`
Defines which proxy and port to use for a specified hostname.

Format:
`proxy:port +(a)-(b);proxy2:port2+(a)-(b); ...;`

a: list of hostnames separated by a comma or a semicolon, for which this proxy shall be used.

b: list of hostnames separated by a comma or a semicolon, for which the proxy shall *not* be used.

The first matching proxy is chosen.

It is also possible to use IP addresses instead of hostnames so `15.*.*.*` or `15:*:*:*:*:*:*:*` would be valid as well, but the correct number of dots or colons MUST be specified. IP version 6 support is not currently available but will be available in the future.

`string DOMAIN`
This defines the default DNS domain to use if no domain is specified for a target host. This domain name will be appended to hostnames not containing a DNS domain name, if a match for the hostname alone cannot be found. This will be done for PROXY lookups and lookups in the [cb.ports] table, for example if the hostname "merlin" is specified and the DOMAIN = "bbn.hp.com", then the [cb.ports] entries will first be searched for the match of "merlin". If there is no match found for the hostname

"merlin", then a search will be made for "merlin.bbn.hp.com", "*.bbn.hp.com", "*.hp.com", "*.com" and "*", in that order.

bbc.fx

BBC File-Transfer Namespace for node-specific configuration. For application-specific settings, see the section bbc.fx.ext.*. Note that application-specific settings in `bbc.fx.ext.*` override node-specific settings in `bbc.fx`. You can use the following parameters:

`int FX_MAX_RETRIES = 3`
Maximum number of retries to be attempted for the successful transfer of the object.

`string FX_BASE_DIRECTORY` = *<directory path>*
Base directory for which files may be uploaded or downloaded. Default directory is `<OvDataDir>`.

`string FX_TEMP_DIRECTORY = <directory path>`
Temporary directory where uploaded files are placed while upload is in progress. At completion of upload, the file will be moved to *<directory path>*. Default directory is `<OvDataDir>/tmp/bbc/fx`.

`string FX_UPLOAD_DIRECTORY = <directory path>`
Target directory for uploaded files. By default this is the base directory. The upload target directory may be overridden with this configuration parameter. Default directory is `FX_BASE_DIRECTORY`.

bbc.snf

BBC Store-and-Forward Namespace for node-specific configuration. For application-specific settings, see the section bbc.snf.ext.*. Note that application-specific settings in `bbc.snf.ext.*` override node-specific settings in `bbc.snf`. You can use the following parameters:

`string BUFFER_PATH = <path>`
Specifies the SNF path were the buffered requests are stored. Default is:

`<OvDataDir>/datafiles/bbc/snf/<app_Name>`

`int MAX_FILE_BUFFER_SIZE = 0`
Specifies the maximum amount of disk space that the buffer is allowed to consume on the hard disk.

`0` = No limit

bbc.http.ext.*

HTTP External-Communication Namespaces: `bbc.http.ext.<compID>.<app_Name>` and `bbc.http.ext.<app_Name>`.

This is the Dynamic External-Communication Namespace for application-specific settings. Note that application-specific settings in `bbc.http.ext.*`override node-specific settings in `bbc.http`.

See the section `bbc.http` for a list of the parameters you can use in the `bbc.http.ext.*` namespace.

bbc.fx.ext.*

The Dynamic File-Transfer (fx) Namespace for external-component and application-specific settings. Note that application-specific settings in `bbc.fx.ext.*` override node-specific settings in `bbc.fx`.

File Transfer External Namespaces: `bbc.fx.ext.<compID>.<app_Name>` and `bbc.fx.ext.<app_Name>`.

See the section `bbc.fx` for a list of the parameters you can use in the `bbc.fx.ext.*` namespace.

bbc.snf.ext.*

The Dynamic Store-and-Forward (snf) Namespace for external-component and application-specific settings. Note that application-specific settings in `bbc.snf.ext.*` override node-specific settings in `bbc.snf`.

Store and Forward External Namespace: `bbc.snf.ext.<compID>.<app_Name>` and `bbc.snf.ext.<app_Name>`.

See the section `bbc.snf` for a list of the parameters you can use in the `bbc.snf.ext.*` namespace.

AUTHOR

`bbc.ini` was developed by Hewlett-Packard Company.

EXAMPLES

```
PROXY=web-proxy:8088-(*.hp.com)+(*.a.hp.com;*)
```

The proxy `web-proxy` is used with port `8088` for every server (`*`) except hosts that match `*.hp.com`, for example `www.hp.com`. If the hostname matches `*.a.hp.com`, for example, `merlin.a.hp.com` the proxy server will be used.

SEE ALSO

bbcutil(1)

bbcutil

NAME

bbcutil

- a tool for debugging a BBC-based server.

SYNOPSIS

```
bbcutil -h|-help

bbcutil -version

bbcutil -ovrg [<ovrg>]

bbcutil -reg|-registrations [<hostname>|<ip>] [-v|-verbose]

bbcutil -deregister {<path>|*} [-force] [-v|-verbose]

bbcutil -ping {[<hostname>|<ip>[:<port>]] | [<uri>]} [count] [-v|-verbose]

bbcutil -status {[<hostname>|<ip>[:<port>]] | [<uri>} [-v|-verbose]]

bbcutil -migrate {[<namespace>] [<appname>] [<filename>]} [-v|-verbose]

bbcutil -count|-size|-list [-p|-path <path>] [-t|-target <target>] [-v|-verbose]

bbcutil -getcbport [<hostname>|<ip>]

bbcutil -gettarget [<hostname>|<ip>]
```

DESCRIPTION

The `bbcutil` command helps you to debug a BBC-based server. The `bbcutil` command can be used to list all applications registered to a Communication Broker, to check whether specified communication services are alive, and to display details about the current state of the server.

Parameters

The `bbcutil` command incorporates the options in the following list. The syntax for the `[<hostname>|<ip>][:<port>]]` string, for example; in the options `-registrations` or `-ping`, can be

a hostname and a port separated by a colon (`:`) but can also be a full URL path (including protocol), such as:

`https://merlin.guilford.mycom.com:383/com.hp.ov.coda`

bbcutil recognizes the following options:

`-h|-help`
Displays and describes the available options for the `bbcutil` command.

`-version`
Displays the version of the HP Software communication in use.

-ovrg <*`ovrg`*>
Executes `a bbcutil` command option in the context of the OV resource group specified by <*`ovrg`*>. This is an optional command. It can be used with other `bbcutil` commands. For example, `bbcutil -ovrg testsrv -getcbport` command returns the Communications Broker port number of the OV resource group, testsrv.

`-reg|-registrations [<`*`hostname`*`>|<`*`ip`*`>]`
Queries a Communications Broker on the node specified by *`<hostname>`* or *`<ip>`* and displays a list of all registered applications. If the hostname or IP address is not specified, localhost is assumed.

`-deregister {<`*`path`*`>|*} [-force]`
Deregisters the specified path from the Communications Broker on the localhost. You can use the asterisk character '*' to denote *all* paths. The specified path will not be deregistered if the application servicing the specified path is currently running. Use the `-force` option to override this behavior and force the path to be deregistered.

`-ping {[<`*`hostname`*`>|<`*`ip`*`>][:<`*`port`*`>]] | [<`*`uri`*`>]} [count]`
Pings the specified HP Software server process. A hostname or IP address with an optional port number or a URL may be given to locate the server process to ping. If a URL is given with the path of a valid process registered with the Communications Broker, the Communications Broker will automatically forward the ping to the registered process. Count specifies the number of times to execute the ping. The node may be specified with a hostname or IP address. Default for the node is "localhost". Default for the port is the Communications Broker port on the specified node. Default count is 1.

`-status {[<`*`hostname`*`>|<`*`ip`*`>[:<`*`port`*`>]] | [<`*`uri`*`>]}`
Displays the status of the specified HP Software server process. A hostname or IP address with an optional port number or a URI may be given to locate the server process. The node may be specified with a hostname or IP address. Default for the node is localhost. Default for the port is the Communications Broker on the specified node.

`-migrate {[<`*`namespace`*`>] [<`*`appname`*`>] [<`*`filename`*`>]} [-v|-verbose]`
Migrates the specified BBC configuration parameters. If no command parameters are specified the BBC 2 LLB and the BBC 4 CB parameters will be migrated to the namespace `bbc.cb` in the configuration database. The BBC 2/3 DEFAULT parameters will be migrated to the namespaces `bbc.http`, `bbc.fx`, and `bbc.snf`. BBC 4 CB parameters will override BBC 2 LLB parameters. The namespace specifies the BBC 2/3/4 namespace to migrate the parameters from. The *`<appname>`* specifies the application name to use in determining the BBC 5 target namespace. Parameters are migrated to the `bbc.http.ext.`*`<appname>`*, `bbc.fx.ext.`*`<appname>`*, and `bbc.snf.ext.`*`<appname>`* namespaces. The file name parameter specifies the file to read the parameters from. Default file name is the BBC 2 standard `default.txt` file and the standard BBC 4 Communications Broker settings.ini file. The BBC 4 settings.ini parameters override the BBC 2 default.txt parameters.

`-count`
Displays the number of requests in a store-and-forward buffer for the specified target, or the entire buffer if no target is specified.

`-size`

The `-size` option displays the size of a store-and-forward buffer. If `-verbose` is specified as well, the size of each individual request is displayed. If a target is specified, only the size of the requests to this target are displayed.

`-list`
The `-list` option displays all requests in a store-and-forward buffer for the specified target or the entire buffer if no target is specified.

`-p|-path <path>`
The `-path` option defines the path to the store-and-forward buffer. This parameter is used to set the BUFFER_PATH parameter.

`-t|-target <target>`
The `-target` option specifies the target URI, whose information you want to display. If no target is specified, information for all targets in the buffer is displayed.

`-verbose`
Shows more detailed output.

`-getcbport` [<*hostname*>|<*ip*>]
Displays the configured Communications Broker port number of the node specified by *<hostname>* or *<ip>*. If the hostname or IP address is not specified, localhost is assumed. If no Communication Broker port number is configured for the node, the default value 383 is displayed.

`-gettarget` [<*hostname*>|<*ip*>]
Displays the IP address of the target node and the Communications Broker port number, or the HTTP Proxy and port number, if a proxy is configured for the specified *<hostname>* or *<ip>*.

AUTHOR

`bbcutil` was developed by Hewlett-Packard Company.

EXIT STATUS

The following exit values are returned:

`0`
bbcutil exited normally with no error.

`1`
Command syntax error encountered. See command syntax for more details on possible values.

`2`
Command partially succeeded.

`3`
Command failed. See command output for more detailed information.

`4`
bbcutil could not complete the requested command due to an authorization error.

`100`
An exception was encountered causing the Communications Broker to exit.

Corresponding error messages are written to stderror.

EXAMPLES

The following examples show you how to use the `bbcutil` command:

- To show the status of Communication Broker on the local node:

  ```
  bbcutil -status
  ```

- To query the communication server located at `https://merlin.guilford.mycom.com:383/com.hp.ov.coda` for details about the current state

of the server:

```
bbcutil -ping https://merlin.guilford.mycom.com:383/com.hp.ov.coda
```

- To get the IP address and Communications Broker port number of a target node `node1`

```
bbcutil -gettarget node1
```

SEE ALSO

ovbbccb(1), bbc.ini(4)

ovbbccb

NAME

ovbbccb

- Controls HTTPS communication using Communication Broker proxies on local nodes.

SYNOPSIS

```
ovbbccb -h|-help

ovbbccb -version

ovbbccb -install|-remove [-v|-verbose]

ovbbccb -daemon|-nodaemon [-debug] [-v|-verbose]

ovbbccb -start|-stop <ovrg> [<hostname>|<ip>] [-v|-verbose]

ovbbccb -kill|-reinit [<hostname>|<ip>] [-v|-verbose]

ovbbccb -listovrg [<hostname>|<ip>] [-v|-verbose]

ovbbccb -ping {[<hostname>|<ip>[:<port>]] | [<uri>} [-v|-verbose]]

ovbbccb -status {[<hostname>|<ip>[:<port>]] | [<uri>} [-v|-verbose]]

ovbbccb -retryfailedrcp -ovrg [<resource_group>]
```

DESCRIPTION

`ovbbccb` command is used to control HTTPS communication using Communication Broker proxies on local nodes. It controls starting of the Communication Broker as a background daemon process or in normal mode, stopping, and re-initializing of the Communication Broker. `ovbbccb` is also used to start and stop OV resource groups in the Communication Broker.

`ovbbccb` can also be used to list all active OV resource groups and all applications registered to a Communication Broker, to check whether specified communication services are alive and to display details about the current state of the server.

Parameters

The `ovbbccb` command incorporates the options in the following list. The syntax for the `[<hostname>|<ip>][:<port>]]` string, for example; in the options `-registrations` or `-ping`, can be a hostname and a port separated by a colon (`:`) but can also be a full URL path including protocol. for example:

```
https://merlin.guilford.mycom.com:383/com.hp.ov.coda
```

ovbbccb recognizes the following options:

`-h|-help`

Displays and describes the available options for the `ovbbccb` command.

`-version`
Displays the version of the OV communication in use.

`-install`
Installs the Communications Broker program as a service on a Microsoft Windows machine.

`-remove`
Removes the Communications Broker program from the services on a Microsoft Windows machine.

`-daemon`
Starts the Communication Broker either as a background daemon process on a UNIX machine or a service on a Microsoft Windows machine.

`-nodaemon`
Starts the Communication Broker as a foreground process (*default*).

`-debug`
Disable Control-C signal handler for debugging.

`-verbose`
Shows more detailed output.

`-start <ovrg> [<hostname>|<ip>]`
Starts the OV resource group specified by *<ovrg>* in the Communication Broker on the host specified by *<hostname>* or *<ip>*. If the hostname or IP is not specified, ovbbccb uses the local host as the host. You must configure the resource group on a cluster node to use this option.

`-stop <ovrg> [<hostname>|<ip>]`
Stops the OV resource group specified by *<ovrg>* in the Communication Broker on the host specified by *<hostname>* or *<ip>*. If the hostname or IP is not specified, ovbbccb uses the local host as the host. You must configure the resource group on a cluster node to use this option.

`-kill [<hostname>|<ip>]`
Stops the Communication Broker on the host specified by *<hostname>* or *<ip>*. If the hostname or IP is not specified, ovbbccb used the local host as the host. You must set the LOCAL_CONTROL_ONLY parameter to false to make this option work on a remote node.

`-reinit [<hostname>|<ip>]`
The Communication Broker specified in *<hostname>* or *<ip>* reloads the configuration data and is re-initialized. If the hostname or IP is not specified, ovbbccb uses the local host as the host.

The `SIGHUP` signal may also be used on UNIX systems to re-initialize the Communication Broker process.

You must set the LOCAL_CONTROL_ONLY parameter to false to make this option work on a remote node.

`-listovrg [<hostname>|<ip>]`
Displays a list of all active OV resource groups for the Communication Broker on the node specified by *<hostname>* or *<ip>*. If the hostname or IP is not specified, ovbbccb uses the local host as the host. You must set the LOCAL_CONTROL_ONLY parameter to false to make this option work on a remote node.

`-ping {[<hostname>|<ip>[:<port>]] | [<uri>]}`
Pings the specified HP Software server process. A hostname or IP address with an optional port number or a URI may be given to locate the server process to ping. If a URI is given with the path of a valid process registered with the Communication Broker, the Communication Broker will automatically forward the ping to the registered process. The node may be specified with a hostname or IP address. Default for the node is "localhost". Default for the port is the HP Software Communication Broker port on the specified node.

`-status {[<hostname>|<ip>[:<port>]] | [<uri>]} [-v|-verbose]`
Displays the status of the specified HP Software server process. A hostname or IP address with an optional port number may be given to locate the server process. Default for the node is "localhost".

Default for the port is the HP Software Communication Broker port on the specified node.

The status message presents the details of all the active and attempted reverse channel connections. For every connection, the following details are listed:

Source machine
The details of the machine that tries to establish the reverse channel connection.

Time and date
The time and date when the node started trying to connect to the Communication Broker through a reverse channel.

Time duration
The time interval for which a node attempted to establish a connection to the Communication Broker through the reverse channel (in milliseconds).

The verbose option displays the following details of every failed connection:

Type of failure
A connection failure can be a time-out, rejection, or a reset. This information helps you identify the true nature of the failure.

Cause of failure
The cause of failure helps you diagnose the underlying problem that triggered the connection failure.

Attempts
The number of attempts made by the node to reinstate the communication is presented within parenthesis.

`-retryfailedrcp [-ovrg <resource_group>]`
This option starts to restore all failed reverse channel connections to the specified resource group. If you do not specify a resource group name, the command tries to restore all failed reverse channel connections to the default resource group.

AUTHOR

`ovbbccb` was developed by Hewlett-Packard Company.

EXIT STATUS

The following exit values are returned:

`0`
ovbbccb exited normally with no error.

`1`
Command syntax error encountered. See command syntax for more details on possible values.

`2`
Command partially succeeded.

`3`
Command failed. See command output for more detailed information.

`4`
The Communications Broker start command failed because a Communications Broker process is already running

`5`
The Communications Broker failed to start because a Local Location Broker process is already running. The HP Software Communications Broker is not supported on systems running the LLB. Stop the LLB before attempting to start the Communications Broker.

`6`
The Communications Broker failed to stop because the Communications Broker process is already stopped.

7
The Communications Broker failed to start due to a bind exception on the Communications Broker port to be opened.

8
The Communications Broker could not complete the command due to an authorization error.

100
An exception was encountered causing the Communications Broker to exit.

Corresponding error messages are written to stderror.

EXAMPLES

The following examples show you how to use the `ovbbccb` command:

- To start the Communication Broker as a daemon process on the local system:

  ```
  ovbbccb -daemon
  ```

- To start the OV resource group `WebCluster1` in the Communication Broker on host `merlin`:

  ```
  ovbbccb -start WebCluster1 merlin
  ```

- To display the status of the specified HP Software server process:

  ```
  ovbbccb -status
  ```

  The following output appears:

  Status: OK

  (Namespace, Port, Bind Address, Open Sockets)

  <default> 383 ANY 2HP OpenView HTTP Communication Incoming Connections

  To machine1.example.hp.com:

  localhost:17282 76bb6662-2cd3-7531-1221-b67340fb721f BBC 06.10.209; ovbbccb 06.10.209

  HP OpenView HTTP Communication Reverse Channel Connections

  Opened from machine1.example.hp.com:

  machine31.example.hp.com:8188 BBC 06.10.143; ovbbcrcp 06.10.143 (1) 30 Jan 2009 15:38:13 GMT 317 ms

  machine32.example.hp.com:8196 BBC 06.10.143; ovbbcrcp 06.10.143 (1) 30 Jan 2009 15:38:13 GMT 241 ms

  Failed from:

  machine21.example.hp.com:8188 BBC 06.10.143; ovbbcrcp 06.10.143 (1) 30 Jan 2009 15:38:13 GMT 307 ms

  machine22.example.hp.com:8196 BBC 06.10.143; ovbbcrcp 06.10.143 (1) 30 Jan 2009 15:38:13 GMT 291 ms

  Pending from :

  machine11.example.hp.com:6244 Connection Refused / remote RCProxy not listening (1) 30 Jan 2009 15:37:58 GMT 3 ms

  machine12.example.hp.com:6252 Connection Refused / remote RCProxy not listening (1) 30 Jan 2009 15:37:58 GMT 2 ms

SEE ALSO

bbcutil(1) bbc.ini(4)

ovbbcrcp

NAME

ovbbcrcp

- a tool to manage Reverse Channel Proxy (RCP) and monitor RCP connections.

SYNOPSIS

```
ovbbcrcp -h|-help

ovbbcrcp -v|-version

ovbbcrcp -kill

ovbbcrcp -status
```

DESCRIPTION

You can use the `ovbbcrcp` tool to manage RCPs and monitor RCP connections. Many HP BTO Software products that follow a client-server architecture use the Black Box Communication (BBC) component for communication. You can use a Reverse Channel Proxy (RCP) to satisfy the advanced security requirements for communication across trust zones separated by firewalls. An RCP allows you to establish a two-way communication (outbound and inbound) channel across a firewall configured to allow only outbound communication.

The RCP functions as a channel between the BBC server and the requests to the BBC server. An established RCP channel is referred to as a reverse channel. A reverse channel through which RCPs request the BBC server to initiate more reverse channels is referred to as a reverse administration channel.

You can deploy an RCP on one of the following:

Any client systems

A dedicated RCP server

To establish a reverse channel, you must configure the BBC server, the BBC client, and the RCP.

Configuring a BBC Server to Enable RCP Communication

To enable communication from clients to the BBC server through an RCP, you must configure each BBC server. The BBC server loads the configuration from the `bbc.<server>` namespace and establishes reverse administration channels during startup. Use the following options to configure a BBC server:

`ENABLE_REVERSE_ADMIN_CHANNELS`- You can set this option to `true` to establish a permanent reverse administration channel with the RCPs specified in the `RC_CHANNELS` option. By default, this option is set to `false` for all BBC servers, except for the BBC Communication Broker (BBC CB). Refer to the following example for more information about this option.

[bbc.cb]

ENABLE_REVERSE_ADMIN_CHANNELS=true

RC_CHANNELS=pnode:9090

The options specified in the example instructs BBC CB on the management server to contact the RCP on the `pnode` node and port 9090 when starting up.

RC_CHANNELS- Use this option to specify the list of RCPs with which you can establish reverse channels. If the OvCoreID is specified, BBC validates this ID against the core ID of the RCP. You can specify multiple RCPs by separating the RCPs using the semicolon (;). You can specify the list of RCPs in the following format.

<RCP_hostname>:<RCP_port>[,<RCP_OvCoreID>][;<RCP2>.....], where *<RCP_hostname>* specifies the RCP host name, *<RCP_port>* specifies the RCP port number, and *<RCP_OvCoreID>* specifies the core ID of the RCP.

You must use the `-ovrg server` option with the `ovconfchg` command if the OVO server runs on a High Availability (HA) cluster. If the OVO server runs as an HA resource group, then use the ovconfchg -ovrg server -ns bbc.cb -set RC_CHANNELS <value> command, where *<value>* specifies the RCPs specified in the `RC_CHANNELS` option.

RC_MAX_WORKER_THREADS/RC_MIN_WORKER_THREADS- The Communication Broker uses different threads to enhance the performance of a reverse channel connection. The `RC_MAX_WORKER_THREADS` option specifies the maximum number of threads that can be used by the Communication Broker and the `RC_MIN_WORKER_THREADS` option specifies the number of threads that will always remain active. By default, `RC_MAX_WORKER_THREADS` is set to one and `RC_MIN_WORKER_THREADS` is set to zero. You can set these options to higher values to enhance the reverse channel communication.

RC_CHANNELS_CFG_FILES- Use this option to specify the list of configuration files. A configuration file can contain a list of one or more RCPs with which you can establish reverse channels. You must place the specified configuration files in the `<OvDataDir>/conf.bbc` directory, where *<OvDataDir>* specifies the name of the data directory. You must use this option in place of the `RC_CHANNELS` option if you use multiple RCPs that require a frequent hostname change. You can specify a list of configuration files by separating the configuration file names using the comma (,) in the following format:

<filename>[,<filename>....], where *<filename>* specifies the name of the configuration file.

Each line in the configuration file can contain only one RCP name. For each RCP, you must specify a port number. The OvCoreID is an optional parameter that you can specify, which must be separated from the port number by a comma as follows. <RCP_hostname>:<port>[,<RCP_OvCoreID>]

If you change only a few RCP host names inside one or more files specified in the RC_CHANNELS_CFG_FILES option, you must use the `ovconfchg` command to trigger the BBC server to refresh the configuration as follows.

ovconfchg ns bbc.cb -set ENABLE_REVERSE_ADMIN_CHANNELS true.

RETRY_INTERVAL- Use this option to specify the retry interval in minutes to establish a reverse channel with an RCP.

RC_ENABLE_FAILED_OVEVENT- Set this option to 'true' to forward the RCP connection failure messages to the HPOM message browser.

Enabling Communication Broker Connections to the RCP

The Communication Broker (ovbbccb) runs with `/var/opt/OV` as the root directory. The name service relevant configuration files that are necessary to open Transmission Control Protocol (TCP) connections are present in the `/etc directory`. This prevents ovbbccb from creating connections to the RCP. You must do as follows to resolve this problem:

Create the directory named `etc under /var/opt/OV`

Copy the name service relevant configuration files (for example, files such as `resolv.conf, hosts, nsswitch.conf)` from `/etc to /var/opt/OV/etc`

Alternatively, you can also disable the `ovbbccb chroot` feature by running the following command. This method resolves the problem of preventing ovbbccb from creating connections to the RCP.

```
ovconfchg -ns bbc.cb -set CHROOT_PATH /
```

Configuring a BBC Client to Enable RCP Communication

To configure a BBC client, you must specify the hosts that must be connected through an RCP. You can specify the list of RCPs in the XPL configuration database under the bbc.http namespace. Use the syntax of the normal proxy configuration to specify the RCP configuration. If you do not specify the port number of the RCP, it is assumed that BBC CB is running on the current node. If you configure the OvCoreID, BBC Client verifies the OvCoreID of the RCP. If the port number of the RCP is not specified in the configuration file or BBC CB, BBC fails to open the connection to RCP.

You can configure a BBC client using the following options:

`PROXY`- Use this option to specify the RCP and port name for a hostname. The format to specify this option is shown in the following example:

`PROXY=pnode.hp.com:9090-(pnode.hp.com,*.noallow.hp.com)+(*.hp.com)`

In the example shown above, the parameters specified are as follows:

`pnode.hp.com` is the name of the RCP

`9090` is the port number

-(*.noallow.hp.com) specifies that the RCP must not be used to connect to all hostnames ending with `.noallow.hp.com`. You can separate multiple hostnames with commas (,) or semicolons (;).

+(*.hp.com) specifies that the specified RCP must be used to connect to all hostnames ending with .hp.com. You can separate multiple hostnames with commas (,)or semicolons (;).

The BBC client connects to the RCP that first matches the specified set of conditions.

In the example shown in this section, the BBC client connects to any host name that ends with `.hp.com` by using the RCP on the system `pnode` and the port `9090`.

You can also use IP addresses instead of hostnames to specify the hosts. For example, +(15.*.*.*) specifies that the RCP must be used to connect to hosts with an IP address that starts with 15. You must not configure a normal proxy server and an RCP on the same system. You must also make sure that you specify the RCP system name in the list of hostnames for which the RCP must not be used. This helps to ease the communication through the RCP.

Configuring RCP

You can use the following option in the `bbc.rcp` namespace to configure RCP.

`SERVER_PORT`- Use this option to specify the RCP port number.

Starting and Stopping RCPs

You can start or stop the RCP process by using the `ovc` command. This command registers the RCP process as `ovbbcrcp` under the `RCP` category.

By default, the `ovbbcrcp` process is not registered with HP Operations Control (OvCtrl). You must register the `ovbbcrcp` process with the `ovctrl daemon` by using the following command.

```
$OvInstallDir/bin/ovcreg -add
$OvInstallDir/newconfig/DataDir/conf/bbc/ovbbcrcp.xml
```

`$OvInstallDir` is the directory in which HP BTO Software is installed.

Refer to the following commands to start or stop an process:

`ovc -start ovbbcrcp`- Use this command to start the RCP process.

`ovc -stop ovbbcrcp`- Use this command to stop the RCP process.

Parameters

The `ovbbcrcp` command recognizes the following options:

`-h|-help`
Displays and describes the available options for the `ovbbcrcp` tool.

`-v|version`
Displays the version of the HP Software RCP.

`-kill`
Stops the RCP on the local node.

`-status`
Displays the RCP status.

AUTHOR

`ovbbcrcp` is developed by Hewlett-Packard Company.

EXIT STATUS

The following exit values are returned:

`0`
`ovbbcrcp` exited normally with no error.

`1`
Command syntax error encountered. Refer to command syntax for more details on possible values.

`2`
Command partially successful.

`3`
Command failed. See command output for additional information.

`4`
The command to start RCP failed due to an existing RCP process.

`6`
The RCP failed to start due to a bind exception on the RCP port to be opened.

`100`
An exception encountered resulted in an RCP exit.

Corresponding error messages are written to stderror.

EXAMPLES

The following example shows you how to use the `ovbbcrcp` tool.

To display the status of the RCP:

```
ovbbcrcp -status
```

Status: OK

(Namespace, Port, Bind Address, Open Sockets)

bbc.rcp 9090 ANY 1

Admin Reverse Channel Connections Accepted

machine.example.hp.com:383 e91b67e4-a337-750a-163c-c3bbd2c257cc BBC 06.00.030; ovbbccb 06.00.030

Admin Reverse Channel Connections Opened

Normal Connections

Incoming

localhost:55464 e91b67e4-a337-750a-163c-c3bbd2c257cc BBC 06.00.030; ovbbcrcp 06.00.030

Outgoing

Queued CONNECT connections

+----------+----------+

|Source Address | Target Address

+----------+----------

HTTP Tunnelled Connections

+----------+----------+--+

| Source Address | Destination Address | Target Address|

+----------+----------+--+

See Also

ovbbccb(1)

OVC

NAME

ovc

- perform actions on local components

SYNOPSIS

`ovc -h|-help`

`ovc -start [<target> ... ] [-boot]{[-async]|[-verbose]}`

`ovc -stop [<target> ... ][-nostart]{[-async]| [-verbose]}`

`ovc -restart [<target> ... ]`

`ovc -kill [-verbose]`

`ovc -status [<target> ... ] [-level <level>]`

`ovc -trace [<target> ... ]`

`ovc -notify <event> [<target> ...] [-value <value>]`

`ovc -version`

DESCRIPTION

`ovc` controls the starting and stopping, event notification, and status reporting of all components registered with the HP Operations Control service.

A component can be a server process belonging to any of the products such as HP Operations Manager for Windows, HP Operations agents (for example, the Performance Agent or the Discovery Agent), an event interceptor, or an application delivered by an integrator. Each component must have an associated registration file providing HP Operations Manager with configuration and process information about the component. For more information about registration, *ovcreg(1)*.

A target can be either a component or a group of components, defined as a category. The `ovc` command first tries to initiate action on the category specified in `target`. If the category called *`target`* is not found,

ovc then tries the individual component called *target*. Note that a category name must not match any component name.

The HP Operations Control daemon or service automatically restarts any component that terminates unexpectedly if the *AutoRestart* option in the registration file of the component is set to *true*. If the HP Operations Control daemon or service is stopped using the -kill option, all registered components are stopped, too.

Parameters

ovc recognizes the following options:

-h|-help
Displays *all* available options for the ovc command.

-start [*<target> ...*] [-boot]{[-async]|[verbose]}
Starts the selected components. *<target>* specifies a component or category. If *<target>* is not used, all components are started. If -boot is used, only components that start at boot time are started.

The -async option starts the components asynchr- -onously. If you use the -verbose option, ovc command displays the progress of the command execution. You can use the -async or the -verbose option, but you must not include these options together in a command.

-stop [*<target> ...*] [-nostart]{[-async]|[verbose]}
Stops the selected components. *<target>* specifies a component or category. If *<target>* is not used, all components are stopped *except* components, which belong to the CORE component group. If you specify the -nostart option and if the control daemon is not running, the command does not perform any action. If you do not specify the -nostart option, the ovc -stop command starts the control daemon and ovbbccb components if these components are not running. The -async option starts the components asynchronously. If you use the -verbose option, the ovc command displays the progress of the command execution. You can use the -async or the -verbose option, but you must not include these options together in a command.

-restart [*<target> ...*]
Stops components before they are restarted. *<target>* specifies a component or category. If *<target>* is not used, all components are stopped and restarted.

-kill [-verbose]
Stops all components registered with the HP Operations Control service. If you use the -verbose option, the ovc command displays the progress of the command execution.

-notify *<event>* [*<target> ...*] [-value *<value>*]
Sends notification of an event with the value of *<value>* to the component or category specified by *<target> ...* . You can specify the <value> to the component that generates the event (event generator) and sends the event-related information to all components that request the event information (event subscribers). If target is not used, the event notification is sent to all components. If *<value>* is not used, only the event notification is sent.

-trace *<target> ...*]
This option is reserved for use by HP Support.

-status [*<target> ...*] [-level *<level>*]
Reports the status of a component or category specified by *<target>*. The status report contains the component's label, description, category, process ID, and STATE. Components can be in state: Stopped (0 in numeric format), Starting (1), Initializing (2), Running (3), Stopping (4), N/A (5) or Aborted (6). If *<target>* is not specified, the status of *all* components is returned. *<level>* specifies the type and quantity of information to display, as follows:

Level 0
Status of registered components monitored by HP Operations Manager.

Level 1
Status of registered components whether they are monitored by HP Operations Manager or not.

`Level 2`
Status of registered components and a dump of their registration information.

`Level 3`
ID of core processes. 0 (zero) indicates root, non-zero indicates non-root ownership.

`Level 4`
Similar to level 0, but the STATE is reported in numeric format.

`Level 5`
Similar to level 1, but the STATE is reported in numeric format.

`Level 6`
Similar to level 0, but the output is not formatted

`Level 7`
Similar to level 1, but the output is not formatted

`-version`
Prints the version of `ovc`

AUTHOR

`ovc` was developed by Hewlett-Packard Company.

EXIT STATUS

The following exit values are returned:

`0`
Success.

`1`
Not defined.

`2`
Ignored.

`62`
The UNIX daemon or Windows service is not running.

`63`
The Control daemon is being initialized.

`64`
Generic error.

`65`
Invalid target.

`67`
Operation aborted.

`69`
Missing prerequisite.

`70`
Authorization error.

`71`
Operation on prerequisite failed.

`73`
Invalid event.

EXAMPLES

The following examples show how to use the `ovc` command and some of its options to control and display important information about registered components.

- To start the component registered as `opcle`:

  ```
  ovc -start opcle
  ```

  Before `opcle` itself starts, all the components that `opcle` depends on are started.

- To start the component registered as `opcle` and display the progress of the command execution:

  ```
  ovc -start opcle -verbose
  ```

  Before `opcle` itself starts, all the components that opcle depends on are started.

- To print the status of all registered components:

  ```
  ovc -status
  ```

- To stop the component registered as `opcle`:

  ```
  ovc -stop opcle -verbose
  ```

  Before `opcle` itself stops, all the components that depend on `opcle` are stopped. This command starts the control daemon and `ovbbccb` components if these components are not running.

- To stop the component registered as opcle using the `ovc -stop[<target>...] -nostart` option:

  ```
  ovc -stop opcle -nostart
  ```

  Before opcle itself stops, all the components that depend on opcle are stopped. This command does not perform any action if the control daemon is not running.

- To send the event `RECONFIGURE` to all running components:

  ```
  ovc -notify RECONFIGURE
  ```

- To start all components (and their dependents) belonging to category SERVER and AGENT.

  ```
  ovc -start SERVER AGENT
  ```

- To print the status of the component `opcle` and display the registration details:

  ```
  ovc -status opcle -level 2
  ```

SEE ALSO

ovcreg(1)

ovcert

NAME

ovcert

- Manages certificates with the Certificate Client on an HTTPS-based node.

SYNOPSIS

```
ovcert -h|-help

ovcert -importcert -file <file> [-pass <passphrase>] [-ovrg <ov_resource_group>]

ovcert -exportcert -file <file> [-alias <alias>] [-pass <passphrase>] [-ovrg <ov_resource_group>

ovcert -importtrusted -file <file> [-ovrg <ov_resource_group>]

ovcert -exporttrusted -file <file> [-alias <alias>] [-ovrg <ov_resource_group>]
```

```
ovcert -certreq [-instkey <file> [-pass <passphrase>]]

ovcert -list [-ovrg <ov_resource_group>]

ovcert -remove <alias> [-f] [-ovrg <ov_resource_group>]

ovcert -certinfo <alias> [-ovrg <ov_resource_group>]

ovcert -check

ovcert -status

ovcert -updatetrusted

ovcert -version
```

DESCRIPTION

The `ovcert` command is used to manage certificates with the Certificate Client on an HTTPS-based node. You can execute tasks such as initiating a new certificate request to the Certificate Server, adding node certificates and importing the private keys, and adding certificates to the trusted root certificates.

Parameters

The `ovcert` command incorporates the following options:

`-h|-help`
Displays usage help for the `ovcert` command options.

`-importcert -file <file> [-pass <passphrase>] [-ovrg <ov_resource_group>]`
Adds the certificate located in the file *<file>* (in *PKCS12* format) as node certificate and imports the private key which must be located in the same file as the private key for the node. The pass phrase for protecting the exported data using encryption specified during creation of the data to import must be specified as parameter *<passphrase>*.

The optional *<ov_resource_group>* parameter can be specified to import an additional certificate on an HA system. As a result, the specified certificate will not be imported to the default location but to the HA default location for the specified package on the shared disk.

-exportcert -file *<file>* [-alias *<alias>*] [-pass *<passphrase>*] [-ovrg *<ov_resource_group>*]
Exports the currently installed node certificate together with its private key to the file system location specified as parameter *<file>* (in *PKCS12* format). The pass phrase for protecting the exported data using encryption specified during creation of the data to import must be specified as parameter *<passphrase>*.

The optional *<ov_resource_group>* parameter can be specified to export an additional certificate on an HA system. As a result, not the default node certificate but the certificate installed for the specified HA package from the shared disk will be exported.

`-importtrusted -file <file> [-ovrg <ov_resource_group>]`
Adds the certificate located in the specified file (in PEM format) to the trusted root certificates.

The optional *<ov_resource_group>* parameter can be specified to import an additional root certificate on an HA system. As a result, the specified root certificates will not be imported to the default location but to the HA default location for the specified package on the shared disk.

`-exporttrusted -file <file> [-alias <alias>] [-ovrg <ov_resource_group>`
Exports the trusted certificate to the file system location specified as parameter *<file>* (in PEM format). The pass phrase for protecting the exported data using encryption specified during creation of the data to import must be specified as parameter *<passphrase>*.

The optional *<ov_resource_group>* parameter can be specified to export an additional certificate on an HA system. As a result, not the default node certificate but the certificate installed for the specified HA package from the shared disk will be exported.

`-certreq [-instkey <file> [-pass <passphrase>]]`
Initiates a new certificate request that is sent to the Certificate Server.

The optional parameters *<file>* and *<passphrase>* can be used to initiate a certificate request that will be based on the installation key that is contained in the specified file. Such an installation key file can be generated with the `ovcm` tool on the certificate server.

The installation key can be used to authenticate the node on the certificate server. Therefore, such a request may be granted automatically without human interaction.

`-list [-ovrg <ov_resource_group>]`
Displays the aliases of the installed certificates andtrusted certificates.

`-certinfo <alias> [-ovrg <ov_resource_group>]`
Displays information such as serial number, issuer, subject, and fingerprint for the certificate specified by *<alias>*.

`-remove <alias> [-ovrg <ov_resource_group>]`
Removes the certificate specified by *<alias>*.

`-check`
Checks whether all prerequisites for SSL communication are fulfilled, such as assigned OvCoreId, installed and valid certificate and private key, and installed and valid trusted certificate.

On completion, the components checked and their status along with the final result are displayed.

`-status`
Contacts the Certificate Client and displays the current certificate status, which can one of the following possible values:

- certificate installed

- no certificate

- pending certificate request

- certificate request denied

- undefined (if Certificate Client can not be contacted)

`-updatetrusted`
Retrieves the currently trusted certificates from the Certificate Server and installs them as trusted certificates on the node.

`-version`
Returns the version of the tool (the component version).

AUTHOR

`ovcert` was developed by Hewlett-Packard Company.

EXIT STATUS

The following exit values are returned:

`0`
All steps were successful.

`1`
One or more steps were not successful.

Corresponding error messages are written to stderror.

EXAMPLES

The following examples show how to use the ovcert command:

- To import the certificate, private key, and trusted certificates located in the file <*file*> to the system's key store:

  ```
  ovcert -importcert -file <file>
  ```

- To add the certificate(s) located in <*file*> to the trusted certificates:

  ```
  ovcert -importtrusted -file <file>
  ```

ovcm

NAME

ovcm

- manages certificates with the Certificate Server in an HTTPS-based environment.

SYNOPSIS

```
ovcm -h|-help

ovcm -version

ovcm -newcacert [-ni]

ovcm -importcacert -file <file> [-pass <passphrase>]

ovcm -exportcacert -file <file> [-pass <passphrase>]

ovcm -listpending [-l]

ovcm -grant <reqid>

ovcm -deny <reqid>

ovcm -remove <reqid>

ovcm -issue -file <file> -name <nodename> [-pass <passphrase>] [-coreid <OvCoreId>] [-ca]

ovcm -genInstKey -file <file> [-context <context>] [-pass <passphrase>]
```

DESCRIPTION

The `ovcm` command is used to manage certificates with the Certificate Server in an HTTPS-based environment. You can execute tasks such as creating public/private key pairs for signing certificates, granting and issuing signed certificates and the corresponding private keys against certificate requests from HTTPS nodes.

Parameters

The `ovcm` command incorporates the following options:

`-h|-help`
Displays all the command-line options for the `ovcm` command.

`-version`
Returns the version of the tool (the component version).

-newcacert [-ni]
Creates a new public/private key pair for signing certificates. If there is already a public/private key pair in use by the certification authority, you are asked whether this should be replaced. Use this option with

care! An initial public/private key pair is automatically created when the Certificate Management component is installed.

The `-ni` non-interactive option creates a new public/private key pair without operator interaction. If a public/private key pair already exists, the request is cancelled.

`-importcacert -file <file> [-pass <passphrase>]`
Imports a certificate for signing certificate requests together with its private key (both contained in one file in PKCS12 format). Use this option with care as the existing certificate and private key are replaced. This option is intended for restoring a backup of the current private key/certificate, for example, if the originals are damaged or destroyed, or for setting up a backup system.

Use *<file>* to specify the name of the file (in PKCS12 format) to import from.

Use *<passphrase>* to specify the text string you use to protect the data. If the `-pass` option is not used, you are prompted to enter the value of the pass phrase.

`-exportcacert -file <file> [-pass <passphrase>]`
Exports the certificate and the corresponding private key of the current certification authority to a file. This option is intended to be used for creating backups. The certification authority private key must be handled very carefully because of its importance to the whole communication environment. It should never be transmitted over the network or stored in an insecure place.

Use *<file>* to specify the name of the file where the certificate data should be written to (in PKCS12 format).

Use *<passphrase>* to specify the text string you use to protect the data. If the `-pass` option is not used, you are prompted to enter the value of the pass phrase.

`-listPending [-l]`
Displays the request IDs of all pending certificate requests.

With the `-l` option, detailed information on every pending request is listed.

`-grant <reqid>`
The selected certificate request is granted and a signed certificate is sent to the requesting certificate client.

The state of the pending certificate request with the request ID *<reqid>* is changed to granted.

`-deny <reqid>`
The selected certificate request is denied and a message is sent to the requesting certificate client.

The state of the pending certificate request with the request ID *<reqid>* is changed to denied.

`-remove <reqid>`
The selected certificate request is removed from the pending pool. No message is sent to the requesting certificate client.

The state of the pending certificate request with the request ID *<reqid>* is changed to removed.

`-issue -file <file> -name <nodename> [-pass <passphrase>] [-coreid <OvCoreId>] [-ca]`
Issues a signed certificate and the associated private key for a node and writes both to the file *<file>* (in PKCS12 format). The file can then be moved to a portable medium and taken to the corresponding node.

The *<nodename>* must be specified as additional information.

The optional *<OvCoreId>* parameter can be used to specify the unique ID of the certificate. If this parameter is empty, a new `OvCoreId` value is generated for the certificate.

The *<passphrase>* parameter is required to protect the generated certificate data. The pass phrase entered is used to calculate an encryption key that then is used to encrypt the generated certificate data. If the `-pass` option is not used, you are prompted to enter the value of the pass phrase.

If you use the `-ca` option, you can use the issued certificate to sign other certificates. This may be necessary if you want to set up a second Certificate Server, which creates certificates that are trusted by all nodes that trust the root Certificate Server.

`-genInstKey -file` *<file>* `[-context` *<context>*`] [-pass` *<passphrase>*`]`

Creates a new installation key, which, together with some additional information is stored in the file *<file>*. The created file should then be securely transferred to the node system.

On the target node, it can then be used to initiate a new certificate request that will be encrypted with the installation key. The certificate server will accept only one request that is encrypted with this key.

This approach offers the advantage that the certificate request (including the private key) is generated on the node system and the system can be authenticated by using the installation key.

The optional parameter *<context>* can be used to add additional (application specific) information that is contained in the certificate request.

The *<passphrase>* parameter is required to protect the generated installation key. The pass phrase entered is used to calculate an encryption key that then is used to encrypt the generated installation key. If the `-pass` option is not used, you are prompted to enter the value of the pass phrase.

AUTHOR

`ovcm` was developed by Hewlett-Packard Company.

EXIT STATUS

The following exit values are returned:

`0`
All steps were successful.

`1`
One or more steps were not successful.

Corresponding error messages are written to stderror.

EXAMPLES

The following examples show how to use the `ovcm` command:

- To create a new public/private key pair for the signing of certificates on the management-server system:

  ```
  ovcm -newcacert
  ```

- To grant the certificate request *<reqid>* and send a signed certificate to the requesting certificate client:

  ```
  ovcm -grant <reqid>
  ```

ovconfchg

NAME

ovconfchg

- manipulates settings files, updates the configuration database, and triggers notification scripts

SYNOPSIS

```
ovconfchg -h | -help

ovconfchg -version

ovconfchg [-ovrg <OVRG>] [-edit | -job {-ns namespace {-set <attr> <value> | -clear <attr> | -cl
```

DESCRIPTION

Installed HP Operations Manager components have associated configuration settings files that contain one or more namespaces. A namespace is a group of configuration settings that belong to a component.

`ovconfchg` manipulates the settings in either the system-wide configuration file or the configuration file for the specified OV Resource Group, `local_settings.ini`, updates the configuration database, `settings.dat`, and triggers notification scripts. If `ovconfchg` is called without options, or only with `-ovrg`, no settings are changed but an update is triggered anyway. This is to allow updating after default settings files have been added, removed, or updated.

When `ovconfchg` runs, all configuration settings are read and merged in memory. Default definitions are used to make corresponding checks, as well as to emit and log warnings in the event of a violation. During this process, file locks are used to prevent parallel updates. A new configuration database is then created containing the merged data.

Parameters

`ovconfchg` recognizes the following options:

`-h | -help`
Displays all the options for the `ovconfchg` command.

`-version`
Displays the version of the `ovconfchg` command.

`-ovrg <OVRG>`
If the parameter you want to change belongs to an OV resource group, use `-ovrg` to specify the name of the resource group. Otherwise, system-wide settings files are opened.

`-edit`
Starts a text editor to edit the settings file, local_settings.ini. The text editor used is determined by the $EDITOR environment variable. If $EDITOR is not set, vi starts on UNIX and Notepad starts on Windows.

A temporary copy of the file is created for editing. After the changes are made, the file is validated for syntax errors. The syntax rule for validation is that the namespace and attribute names should contain only letters (a-z, A-Z), digits (0-9), period(.) and underscore(_) characters.

If the validation fails, the line number of the error is reported and the user will be prompted to correct the file. If Yes, the file will be reopened for making the necessary changes. If No, the original settings file remains unchanged. If the validation is successful, the changes are saved into the original settings file.

Do not configure binary values using this option. This can corrupt the file. It is also recommended to restrict the data entered using this option to US-ASCII (7-bit only) subset.

Do not open the settings file directly in a text editor and change it. This can corrupt the file.

`-job`
Create an update job file only and do not synchronize.

`-ns | -namespace <namespace>`
Sets a namespace for the `-set` and `-clear` options.

`-set <attr> <value>`
Sets an attribute value in the namespace specified by the `-namespace` option. The local or OV resource settings file is updated accordingly.

`-clear <attr>`
Clears the local setting for the attribute *attr* in the namespace specified by the `-namespace` option. The local settings file is updated accordingly.

`-clear -all`
Clears all local settings. The local settings file is updated accordingly.

AUTHOR

`ovconfchg` was developed by Hewlett-Packard Company.

FILES

The `ovconfchg` command uses the following files to store local settings:

- `<DataDir>/conf/xpl/config/local_settings.ini`
- `<ShareDir>/<OVRG>/conf/xpl/config/local_settings.ini`

The `ovconfchg` command uses the following files to store database configuration settings:

- `<DataDir>/datafiles/xpl/config/settings.dat`
- `<ShareDir>/<OVRG>/datafiles/xpl/settings.dat`

EXAMPLES

The following examples show how to use the `ovconfchg` command:

- To assign the value `12` to the attribute `COUNT`, and assign the value `"red blue white"` to the attribute `COLORS` in the namespace, `tst.lib`:

  ```
  ovconfchg -ns tst.lib -set COUNT 12 -set COLORS "red blue white"
  ```

- To clear the attribute `COUNT` in the namespace `tst.lib`:

  ```
  ovconfchg -ns tst.lib -clear COUNT
  ```

- To remove all locally configured attributes from the namespace `tst.lib`:

  ```
  ovconfchg -ns tst.lib -clear '*'
  ```

- For the OV resource group `server`, assign the value `50` to the attribute `COUNT` in the namespace `tst.lib`:

  ```
  ovconfchg -ovrg server -ns tst.lib -set COUNT 50
  ```

SEE ALSO

ovconfget(1)

ovconfget

NAME

ovconfget

- returns specified attributes from the configuration database.

SYNOPSIS

```
ovconfget  -h | -help

ovconfget  -version

ovconfget  [-ovrg <OVRG>] [<namespace> [<attr>]]
```

DESCRIPTION

Installed HP Software components have associated configuration settings files that contain one or more namespaces and apply system wide or for a specified OV Resource Group. A namespace is a group of configuration settings that belong to a component. All configurations specified in the settings files are duplicated in the `settings.dat` configuration database.

For each specified namespace, `ovconfget` returns the specified attribute or attributes and writes them to stdout. Used without arguments, `ovconfget` writes all attributes in all namespaces to stdout.

Parameters

`ovconfget` recognizes the following options:

`-h | -help`
Displays the options for the `ovconfget` command

`-version`
Displays the component version

`-ovrg <OVRG>`
Specifies the named OV Resource Group `<OVRG>`.

<namespace> <attr>
Obtains the specified attribute in the specified namespace for the named OV Resource Group `<OVRG>` and writes them to stdout. If *`namespace`* is used without specifying an attribute, `<attr>`, `ovconfget` writes the contents of the database for the specified namespace. If neither `<attr>` nor `<namespace>` is specified, `ovconfget` writes the complete contents of the configuration database to stdout.

AUTHOR

`ovconfget` was developed by Hewlett-Packard Company.

FILES

The `ovconfget` command uses the following files to read configuration-database settings:

- `<DataDir>/datafiles/xpl/config/settings.dat`
- `<ShareDir>/<OVRG>/datafiles/xpl/settings.dat`

EXAMPLES

The following examples show how to use the `ovconfget` command:

- To return the value of the `Port` attribute in the `tst.settings` namespace, for example: `9012`

  ```
  ovconfget tst.settings Port

  9012
  ```

- To return all attributes in the `tst.settings` namespace as multiple lines in the form of *`attr`*=*`value`*, for example:

  ```
  ovconfget tst.settings

  Port=9012

  Protocols=HTTP FTP HTTPS

  MaxFileSize=128
  ```

- To return all attributes in all namespaces on multiple lines, for example:

  ```
  ovconfget
  ```

```
[tst.lib]
LibraryPath=/opt/OV/lib:/opt/OV/lbin/tst/var/opt/OV/tmp
[tst.settings]
Port=9012
Protocols=HTTP FTP HTTPS
MaxFileSize=128
```

SEE ALSO

ovconfchg(1)

ovcoreid

NAME

ovcoreid

- Manages the unique node identifier `OvCoreId` on the local node.

SYNOPSIS

`ovcoreid -show [-ovrg <OV_Resource_Group>]`

`ovcoreid -create [-force] [-ovrg <OV_Resource_Group>]`

`ovcoreid -set <OvCoreId> [-force] [-ovrg <OV_Resource_Group>]`

`ovcoreid -version`

`ovcoreid -h|-help`

DESCRIPTION

The `ovcoreid` command is used to display existing `OvCoreId` values and, in addition, create and set new `OvCoreId` values on the local node.

Parameters

The `ovcoreid` command accepts the following parameters and options:

`-show [-ovrg <OV_Resource_Group>]`
Displays the current `OvCoreId` of the system (configuration setting `CORE_ID` in namespace `[sec.core]`). This is the default if no parameters are specified. If the OvCoreId you want to show belongs to an OpenView Resource Group, use the `-ovrg` option to specify the name of the Resource Group. If an OV Resource Group is specified, the corresponding configuration settings will be read or modified as well.

If you specify a non-existent OV Resource Group, `ovcoreid` displays the local `OvCoreId`.

`-create [-force] [-ovrg <OV_Resource_Group>]`
Generates a new `OvCoreId`. If a CORE_ID value already exists, the existing OvCoreId is only overridden when `-force` is specified. If the OvCoreId you want to show belongs to an OpenView Resource Group, use the `-ovrg` option to specify the name of the Resource Group. If an OV Resource Group is specified, the corresponding configuration settings will be read or modified as well.

If you specify a non-existent OV Resource Group, `ovcoreid` displays an error.

`-set [-force] [-ovrg <OV_Resource_Group>]`

Sets a specific `OvCoreId`. The `-force` option must be used if an `OvCoreId` value has already been set. If the OvCoreId you want to show belongs to an OpenView resource group, use the `-ovrg` option to specify the name of the resource group. If an OV Resource Group is specified, the corresponding configuration settings will be read or modified as well.

`-version`
Returns the version of the tool (the component version).

`-h|-help`
Display all available command options.

AUTHOR

`ovcoreid` was developed by Hewlett-Packard Company.

EXIT STATUS

The following exit values are returned:

`0`
All steps were successful.

`1`
If `-create` or `-set` is used without `-force` and a value for `OvCoreId` already exists.

`2`
One or more steps were not successful.

Corresponding error messages are written to stderror.

Note: Changing the `OvCoreId` of a system is analogous to giving the system a new identity and is an action that should only be executed if the consequences are fully understood. Changing the `OvCoreId` of a system requires a number of significant changes including the need for a new certificate, and having to do appropriate reconfiguration of the HP Software server(s).

EXAMPLES

The following examples show you how to use the `ovcoreid` command:

- To display the `OvCoreId` for the local node:

  ```
  ovcoreid -show
  ```

- To create and set a new `OvCoreId` on the local node:

  ```
  ovcoreid -create
  ```

- To set the specified `OvCoreId` on the local node:

  ```
  ovcoreid -set <OvCoreId>
  ```

SEE ALSO

ovconfget(1), ovconfchg(1).

ovcreg

NAME

ovcreg

- component registration tool

SYNOPSIS

```
ovcreg -h|-help
```

```
ovcreg -check [<filename>]

ovcreg -add [<filename>]

ovcreg -del [<component>]

ovcreg -version
```

DESCRIPTION

`ovcreg` is used to register a component with (and de-register the component from) the OvCtrl. The ovcreg command can also be used to check a component registration file for syntactical correctness.

If the OvCtrl daemon (`ovcd`) is running at the time of registration, it will be informed about the new component only if the `-add` option was applied and the component is not started. The OvCtrl shows the new component the next time the `ovc` command is called with the `-status` option.

If the OvCtrl daemon (`ovcd`) is running, the component will be stopped if the `-del`(ete) option was applied. NOTE: this option will *not* stop CORE components, which are denoted by the option *CoreProcess* in the registration file. CORE components should be stopped with `ovc` command and the `-kill` option.

Parameters

`ovcreg` recognizes the following options:

`-h|-help`
Displays *all* available options for the `ovcreg` command.

`-check [<filename>]`
Checks the syntax of `<filename>`. `<filename>` must not contain more than one component.

`-add [<filename>]`
Checks the syntax of `<filename>` and stores a copy in the configuration directory. Adding a component with a name which is already registered with the OvCtrl will overwrite the original registration with the new one. `<filename>` must not contain more than one component.

`-del [<component>]`
Stops and de-registers the specified `<component>` from the OvCtrl and deletes the specified `<component>` registration file. NOTE: the delete option does not stop CORE components.

`-version`
Displays the version of ovcreg

AUTHOR

`ovcreg` was developed by Hewlett-Packard Company.

EXIT STATUS

The following exit values are returned:

`0`
Success - The syntax of the file is correct and the registration file is successfully added or deleted.

`1`
Wrong usage

`2`
Parsing error

`3`
Error deleting registration file

`5`
Error writing XML file

`6`
Component is not registered

7
Error stopping component

8
Error deleting component

FILES

Registration files for components registered with the OvCtrl for the supported platforms reside in the following locations:

- AIX, HP-UX, Linux, Solaris:

  `/var/opt/OV/conf/ctrl/*.xml`

- True64:

  `/usr/var/opt/OV/conf/ctrl/*.xml`

- Windows:

  `C:\Program Files\HP\HP BTO Software\conf\ctrl\*.xml`

Note that the user can change the specified default location for the registration files on machines running Microsoft Windows.

EXAMPLES

The following examples show how to use the `ovcreg` command and some of its options to control and display important information about registered components.

- To check the syntax of the component registration file: `opcle.xml`:

  ```
  ovcreg -check opcle.xml
  ```

- To check the syntax of the component registration file, `opcle.xml`, and add the component defined in the component registration file, `opcle.xml` to the OvCtrl:

  ```
  ovcreg -add opcle.xml
  ```

- To stop and de-register the component registered as `opcle`:

  ```
  ovcreg -del opcle
  ```

SEE ALSO

ovc(1)

ovlogdump

NAME

ovlogdump

- dumps a specified binary log file as text in the current locale to the console

SYNOPSIS

```
ovlogdump -h|-help

ovlogdump -version

ovlogdump [<binary_logfile_name>]

ovlogdump -merge -tofile <binary_logfile_name> -fromfiles <binary_logfile1_name> <binary_logfile
```

DESCRIPTION

The ovlogdump command dumps a binary log file as text in the current locale to the console. To view the contents of a log file, specify its location and name; else, the system.bin file is dumped to the console by default.

By default, all the log files are stored in the following location:

On Windows:

```
C:\Documents and Settings\All Users\Application Data\HP\HP BTO Software\log
```

On UNIX:

```
/var/opt/OV/log
```

If permissions are inadequate for the default locations, the log files are stored in the <*OvDataDir*>/log/public directory.

During application logging, if multiple log files are created, you can use the -merge option to merge these files into a single binary log file.

Parameters

ovlogdump recognizes the following options:

[<*binary_logfile_name*>]
The name and location of the binary log file to be dumped. If the log file name is not specified, system.bin file in the <*OVDataDir*>/log/ directory is displayed on the console by default.

-merge -tofile <*binary_logfile_name*> -fromfiles <*binary_logfile1_name*> <*binary_logfile2_name*>....
Merges application log files specified by <*binary_logfile1_name*>.... into a single binary log file specified by <*binary_logfile_name*>. This option is not supported for merging system log files.

-h|-help
Displays all available options for the ovlogdump command.

-version
Displays the version of the ovlogdump command.

AUTHOR

ovlogdump was developed by Hewlett-Packard Company.

ovpolicy

NAME

ovpolicy

- installs, manages, and removes both local and remote policies.

SYNOPSIS

```
ovpolicy -help

ovpolicy -version

ovpolicy -install [-host <hostname> [-targetid [<id>]...] {-enabled|-disabled} -chkvers -add-category [
|-remove-all-categories} -force-cat -add-attribute [<name> <value>]... -remove-attribute [<name> <value
-force-attr -set-owner <owner> -force-owner -no-notify]
{-file [<file>]...|-dir [<dir>]...} [-ovrg <ov_res_group>]

ovpolicy -remove [-no-notify -host <hostname> [-targetid [<id>]...] [-ovrg <ov_res_group>] <SELECTION>
```

```
ovpolicy [-enable |-disable] [-no-notify -host <hostname> [-targetid [<id>]...] [-ovrg <ov_res_g
ovpolicy [-addcategory |-removecategory] <cat>... [-no-notify -host <hostname> [-targetid [<id>]
ovpolicy -removeallcategories [<cat>]... [-no-notify -host <hostname> [-targetid [<id>]...] [-ov
ovpolicy [-addattribute |-removeattribute] <name> <value>... [-no-notify -host <hostname> [-targ
ovpolicy -removeallattributes [-no-notify -host <hostname> [-targetid [<id>]...][-ovrg <ov_res_g
ovpolicy [-setowner | -removeowner <owner>] [-no-notify -host <hostname> [-targetid [<id>]...][-
ovpolicy -notify [-host <hostname> [-targetid [<id>]...][-ovrg <ov_res_group>]]
ovpolicy -list [-level <0|1|2|3|4> -host <hostname> [-targetid [<id>]...][-ovrg <ov_res_group>]]
```

DESCRIPTION

`ovpolicy` installs, manages, and removes, local and remote policies. A policy is a set of one or more specifications rules and other information that help automate network, system, service, and process management. Policies can be deployed to managed systems, providing consistent, automated administration across the network. Policies can be grouped into categories, for example; to assign policies to a special policy group for simple enable and disable actions. Each category can have one or more policies. Policies can also have one or more attributes, an attribute being a name value pair.

You use `ovpolicy` to, among other functions, install, remove, enable, and disable local policies. For information about the parameters supported by the `ovpolicy` command, see "Parameters": for information about parameter options, see "Options".

Parameters

`ovpolicy` recognizes the following parameters:

`-install`
Installs one or more policies using a single policy file specified with `-file` or multiple policy files specified with `-dir`.

`-remove`
Removes one or more policies.

`-enable`
Enables one or more policies.

`-disable`
Disables one or more policies. Note that the `-disable` option only disables a policy, it does not remove a policy from the file system.

`-addcategory`
Adds all category strings to the policy. You can add multiple categories using a blank-separated list.

`-removecategory`
Removes the specified category strings from the policy. You can remove multiple categories using a blank-separated list.

`-removeallcategories`
Deletes *all* categories.

`-addattribute`
Adds a category attribute to the policy. You can add multiple attribute names using a blank-separated list.

`-removeattribute`
Removes category attribute from the policy. You can remove multiple attribute names using a blank-separated list.

`-removeallattributes`

Deletes *all* category attributes.

`-setowner`
Sets the owner of a policy.

`-removeowner`
Removes the owner of a policy.

`-list`
Lists the installed policies.

`-notify`
Triggers any notifications to the OV control service, if there are any outstanding or suppressed notifications from previous policy operations.

`-version`
Displays the version number of the command.

`-h | -help`
Displays the help information.

Options

You can use the following options with the allowed `ovpolicy` command parameters:

`-add-attribute`
Add an attribute *<name>* with the value defined in *<value>* to the specified installed policy.

`-add-category <cat1> [<cat2> ... <catN>]`
Adds all category strings to the policy. This is a blank-separated list.

`-chkvers`
Check and compares the version of the already installed policy and the policy you want to install. If `-chkvers` is used, the new policy is not installed if the current installed version is the same or higher. If `-chkvers` is not used, the new policy overwrites the current policy with the same policy_id, regardless of the version number. `-chkvers` does not overwrite the categories, owner, or status of a current policy. To overwrite the categories, owner, and status associated with a policy owner, use `-forcecat`, and `-forceowner` respectively.

`-dir <dirname>`
If you specify a directory name, all policy files from that directory are used. A line is printed to stdout for each successfully installed policy.

`-enabled|-disabled`
If either `-enabled` or `-disabled` is used, the new policy acquires the status that is defined in the policy header. If neither `-enabled` nor `-displayed` is used, the new policy acquires the status of the currently installed policy (if any).

Note that this option overwrites the status defined in the policy-header installation file. So, if the new policy is already installed on the target system, the new version assumes the status of the installed version.

`-file <filename>`
Specifies a policy file name to be used. A line is printed to stdout for the successfully installed policy.

`-force-attr`
Allows you to remove category attributes that are set on a current installed policy. By default, the attributes from current installed policies are used. If there is no current installed policy, the attributes set in the header file of the new policy are used.

`-force-cat`
Allows you to remove categories that are set on a current installed policy. By default, the categories from current installed policies are used. If there is no current installed policy, the categories set in the header file of the new policy are used.

`-force-owner`
Overwrites the policy owner regardless of the settings for the installed policy.

`-host <hostname> [-targetid <ids>]`
This option specifies the hostname of the managed node. If no hostname is specified, the local host is assumed. `-targetid` specifies one or more target IDs.

`-level`
Specifies the type of information to be returned with the `-list` parameter, as follows:

`0`
Policy type, policy name, status, policy version. This is the default setting.

`1`
Policy type, policy name, status, policy version, policy_ID.

`2`
Policy type, policy name, status, policy version, policy_ID, category.

`3`
Policy type, policy name, status, policy version, policy_ID, category, owner.

`4`
Policy type, policy name, status, policy version, policy_ID, category, owner, attributes.

`-no-notify`
When `-no-notify` is used, `ovpolicy` does not trigger any notifications.

`-remove-category <cat1> [<cat2> ... <catN>]`
Removes the specified category strings from the policy. Using the `-remove-category` option with an empty string deletes *all* categories. This is a blank-separated list.

`-remove-all-categories`
Removes the specified category strings from the policy.

`-remove-attribute`
Remove the category attribute `<name>` with the value defined in `<value>` from the specified installed policy.

`-remove-all-attributes`
Allows you to remove *all* category attributes that are set on a current installed policy. If there is no current installed policy, the attributes set in the header file of the new policy are used.

`-set-owner <owner>`
Sets the owner of a policy. `-set-owner` with an empty string deletes the owner.

`-ovrg <ovrg_res_group>`
Sets the name of the OV resource group.

`The <SELECTION>` option is one of the following:

`<SELECTION>-all|-owner <owner>|-owner <owner> -polname <name>|-polid <uuid> |-polname <[type:]name>|-poltype <typename>|-category <category> |-attribute <name> [value]`

`-all`
All installed policies.

`-owner <owner>`
The policy owner `<owner>`

`-owner <owner> -polname <name>`
The policy owner `<owner>` and the policy name `-owner <name>`

`-polid <id>`
The ID of the policy.

`-polname [<policy_type_name>:]<policy name>`
The name of the policy. If `policy_type_name` is used, the section applies to all policies of the specified type.

`-poltype <policy_type_name>`

The name of the type of policy.

`-category <category_name>`
The name of the category to be used.

`-attribute <name> <value>`
The name of the policy attribute and value to be used.

Return Codes

`ovpolicy` recognizes the following return codes:

`0`
All steps were successful.

`1`
One or more steps were not successful.

AUTHOR

`ovpolicy` was developed by Hewlett-Packard Company.

EXAMPLES

The following examples show you how to use the `ovpolicy` command:

- To list all policies on a node.

  ```
  ovpolicy -list
  ```

- To disable the HP-UX `syslog` policy.

  ```
  ovpolicy -disable -polname "HPUX ovsyslog"
  ```

- To enable all trap policies.

  ```
  ovpolicy -enable -poltype ovsnmptrap
  ```

- To install all policies located in the current working directory.

  ```
  ovpolicy -install -dir .
  ```

- To install all policies located in the `/tmp/sap_policies` directory with a status of disabled.

  ```
  ovpolicy -install -disable -dir /tmp/sap_policies
  ```

- To reinstall all policies located in the `/tmp/xyz` directory, independent of the former owner.

  ```
  ovpolicy -install -forceowner -dir /tmp/xyz
  ```

- To remove all policies from the local host.

  ```
  ovpolicy -remove -all
  ```

- To remove all installed policies that are owned by the management server

  ```
  ovpolicy -remove -owner mgtsvr
  ```

ovtrccfg

NAME

ovtrccfg

- enables the tracing mechanism for supported applications on the local machine.

SYNOPSIS

```
ovtrccfg -app|-application <application_name>
[-cm|-component <component_name>]
[-gc|-generate_configuration <filename>][-sink
<filename>]

ovtrccfg -cf|-configuration <filename>

ovtrccfg -off

ovtrccfg -version

ovtrccfg -h|-help

ovtrccfg -vc
```

DESCRIPTION

The **ovtrccfg** command helps you enable and configure the tracing mechanism to record the state of a supported application on the system where an HP Software product is installed. By default, trace log files are placed into the application's home directory after you enable the tracing mechanism. When you configure the tracing mechanism with the **gc** option, all configuration details are directed to a trace configuration (`.tcf`) file. You can create and modify trace configuration files with the command or with a text editor.

In the trace configuration file, you can specify the location of trace log files with the sink option. When you start the tracing process without a configuration file, all available trace levels and categories are enabled. If you want to enable only select levels of tracing, you must use a trace configuration file.

The tracing mechanism provides the following different levels of tracing:

`Info`
Enable traces marked as information.

`Warn`
Enable traces marked as warning.

`Error`
Enable traces marked as error.

`Support`
Enable the normal tracing. The trace output includes informational notifications, warnings, and error messages. This option is recommended for troubleshooting problems. This level of tracing can be enabled for a long duration as the overhead to capture the trace output is minimal with this option.

In addition, you can use the location, stack, developer, and verbose levels when detailed trace messages are requested by HP Support.

Parameters

The `ovtrccfg` command accepts the following parameters and options:

`-app|-application <application_name>`
This option helps you enable the tracing mechanism for select HP Software applications. These applications are essentially programs, daemons, processes, and services that are used by different HP Software products.

`-cm|-component <component_name>`
You can enable tracing of select components of an application with the **cm** option. By default, all components of an application are traced by the tracing mechanism. You can use the wildcard character (`*`) with this option. For example, the `ovtrccfg -app coda -cm xpl*` command starts tracing for all the components, which belong to the coda application, with the names that begin with `xpl`.

`-cf|-configuration <filename>`

You can enable the tracing mechanism according to the rules specified in a configuration file. The configuration files are stored on the same system with the extension .tcf.

-sink *<filename>*
The **sink** option helps you direct the trace log files to a location of your choice on the local system. All trace log files generated with the command are placed into the location specified with the sink option.

-gc|-generate_configuration *<filename>*
The gc option creates a trace configuration file (.tcf) that can be edited to set the desired tracing configuration.

-off
The off option helps you disable the tracing process. If you use the off option without any other options, the entire tracing mechanism stops. You can use the app and cm options with the off option to conditionally exclude select applications and components when you enable tracing. For example, the "ovtrccfg -app o* -off ovc*" command enables tracing for all applications with the names that begin with "o," but excludes the applications with the names that begin with "ovc." Similarly, the "ovtrccfg -app ovoadif -cm e* -off eaagt.misc" command enables the tracing mechanism for all components with the names that begin with "e," which belong to the application "ovoadif," except the component eaagt.misc.

-vc
This option displays the current tracing status of all the supported applications available on the system.

-version
This option displays the version of this command.

-h|-help
Displays all available command options.

AUTHOR

ovtrccfg is developed by Hewlett-Packard Company.

EXAMPLE

The following examples show how to use the ovtrccfg command:

- Enable the tracing mechanism for all applications with the names that begin with o:

  ovtrccfg -app o*

- Enable the tracing mechanism for the **coda** application and direct the trace log files to the /opt/OV/support directory:

  ovtrccfg -app coda -sink /opt/OV/support/output.trc

- Enable the tracing mechanism on the local system based on the rules set in the trace configuration file config.tcf:

  ovtrccfg -cf config.tcf

ovtrcmon

NAME

ovtrcmon

- helps you view the trace messages from trace files and enables you to store the trace messages into another file on the same system.

SYNOPSIS

```
ovtrcmon [-h|-help] -fromfile <source_file>
-tofile <target_file>] -short|-long|-verbose|[-fmt
<format_name>]
```

DESCRIPTION

The **ovtrcmon** command helps you view the contents of a trace file and lets you store the file content into another file on the same machine. When you start the tracing mechanism with the `ovtrccfg`command, trace messages get captured into trace files in the binary format. To read the contents of a trace file, you can use the "`ovtrcmon -fromfile <source_file> -fmt <format>`" command. Alternatively, you can store the contents of a trace file into a new file in a readable format with the "`ovtrcmon -fromfile <source_file> -tofile <target_file> -fmt <format>`" command. With the help of the configuration file `$OvDataDir/conf/xpl/trc/ovtrcmon.cfg`, you can specify a customized format of your choice that you want to use while viewing and storing the contents of trace files. You can use the following keywords while configuring this file:

`Severity`
: The trace file captures trace messages with different severity levels. This keyword helps you filter the trace messages based on the severity level. Available severity levels are: Info, Warn, Error, Support, Location, Stack, Developer, and Verbose.

`Count`
: The serial number for a particular trace message.

`Tic`
: A high-resolution elapsed time value.

`LocalTime`
: The local equivalent date and time of the trace message.

`UTCTime`
: The UTC time of the trace message.

`Pid`
: The process ID of the traced application.

`Tid`
: The thread ID of the traced application.

`Component`
: The name of the component issuing the trace message.

`Category`
: An arbitrary name assigned by the traced application or one of several categories provided by the tracing mechanism.

`Source`
: The line number and file name of the source generating the trace.

`Stack`
: A description of the calling stack in the traced application.

`TrcMsg`
: Trace message description.

`Attribute`
: Attribute of the trace message.

`Application`
: Name of the traced application.

`Machine`
: Name of the machine where the traced application resides.

`Formatting`
: You can use one of four types of formatting on the trace output.

The Formatting keyword helps you generate the output in the following formats:

CSV
: Comma separated values. This keyword presents the output in a standard delimited format with double quotes (") around the text.

formatted
: A *printf*-like output format.

fixed
: This keyword presents the output with fixed-width fields and whitespace padding. Field widths are specified after the keyword `fixed` with commas. For example, **`fixed,w1,w2,..wn ]`**.

xml
: Presents the trace output in the XML format.

Parameters

The `ovtrcmon` command accepts the following parameters:

`-fromfile <source_file>`
: With this parameter, you can specify the name of the binary trace file.

`-tofile <target_file>`
: With this parameter, you can specify the name of the file where you want to direct the contents of the trace file.

`-long`
: Displays or stores the following details from the trace file: Severity, Component, Category, and trace description.

`-short`
: Displays or stores only the trace description from the trace file.

`-verbose`
: Displays or stores all details available in the trace file.

`-fmt`
: With this parameter, you can view the contents of the trace file in a pre-configured format. You must specify the format definitions in the `$OvDataDir/conf/xpl/trc/ovtrcmon.cfg` file. You must declare `<format_name>` in this configuration file.

`-h|-help`
: Displays all available command options.

AUTHOR

`ovtrcmon` is developed by Hewlett-Packard Company.

EXAMPLE

The following examples show how to use the `ovtrcmon` command:

- View the trace messages in the `$OvDataDir/log/example1.trc`file in the format `format1`, which is defined in the `$OvDataDir/conf/xpl/trc/ovtrcmon.cfg` file:

  `ovtrcmon -fromfile $OvDataDir/log/example1.trc -fmt format1`

- View only the descriptions of the trace messages in the `$OvDataDir/log/example1.trc` file:

  `ovtrcmon -fromfile $OvDataDir/log/example1.trc -short`

- Store the available trace messages in the `$OvDataDir/log/example1.trc` file into the `$OvDataDir/log/trace.txt` file in the format `format1`, which is defined in the `$OvDataDir/conf/xpl/trc/ovtrcmon.cfg` file:

```
ovtrcmon -fromfile $OvDataDir/log/example1.trc -tofile $OvDataDir/log/trace.txt
-fmt format1
```