Hewlett Packard Enterprise

VMware event forwarding with OAVA and VISPI

2016/02/12

Auhor: Thierry Ledent

Thierry has been with HPE since 1995 and is currently working as a Software Support Technical Consultant. His focus is on Operations Agent, Infrastructure Smart Plug-In and Virtualization.

thierry.ledent@hpe.com

Abstract

VMware vSphere events are records of user actions or system actions that occur on objects in the vSphere environment. These include actions such as a lost connection to an ESXi host, a virtual machine power off, a CPU alarm state change ...etc...

This paper describes how to configure OAVA and VISPI to collect events from the vCenter and forward these events as OM alerts to the OM server. It provides basic configuration steps, detailed configuration steps and troubleshooting steps.

Get the latest updates of this document at:
https://softwaresupport.hp.com/group/softwaresupport/search-result/-/facetsearch/document/KM02133988

1 Basic configuration steps

Follow the steps below to enable basic VMware event forwarding to the OM server. Detailed configuration steps are described in "Detailed configuration steps".

1.1 Version and patch level

Although VMware event forwarding is implemented in OAVA and VISPI since the initial release 11.11, significant improvements have been added in the later patches. A major fix is available in the latest VISPI hotfix for VMware event forwarding.

VMware event forwarding will work best with following versions:

- OAVA 11.14 + hotfix HF_OA_183097_183097_VA or later
- VISPI 11.14 + hotfix HOTFIX_SPI_VI_2014-11-23_2 or later

This paper assumes that the above mentioned OAVA and VISPI versions and hotfixes are installed. Hotfixes are available from HPE Software Support.

For an optimal setup of OAVA, check the paper "How to achieve optimal performance and stability of OAVA" at https://softwaresupport.hp.com/group/softwaresupport/search-result/-/facetsearch/document/KM01013486.

Hewlett Packard Enterprise

1.2 Policy configuration and deployment

VMware event forwarding is implemented through two VISPI policies deployed on the OAVA.

1.2.1 Policy `VI-VMwareVCEventTypes`

The policy `VI-VMwareVCEventTypes` defines which types of events should be collected from the vCenter. Details on the configuration of this policy are provided in "Managing events types" and "Monitoring alarm state changes". For a basic configuration of VMware event forwarding, the default policy (typically version 1111.0000) can be deployed to the OAVA.

1.2.2 Policy `VI-VMwareVCEventMonitor`

The policy `VI-VMwareVCEventMonitor` translates the VMware events collected by OAVA into OM alerts and forwards them to the OM server. Details on the configuration of this policy are provided in "Configuring OM alerts" and "Monitoring alarm state changes".

Before deploying this policy, make sure to align its policy interval on the event collection interval. The event collection interval is defined by the XPL setting `Virt_Node_Coll_Interval` in namespace `opsagt`. Check the value of this setting with:

```
# ovconfget opsagt Virt_Node_Coll_Interval
300
```

The above example indicates an event collection interval of 300 seconds (this is also the recommended value).

Failure to align the policy interval on the event collection interval can lead to missing a random amount of events (all events can be missed in the worst case scenario).

1.3 OM configuration

When the policy `VI-VMwareVCEventMonitor` forwards a VMware event as an OM alert, it attempts to identify the name of the node that triggered the event and fills in the node name accordingly in the OM alert. If the corresponding node is not configured in the OM node bank, the alert will be dropped on the OM server. *All the nodes that may trigger an alert should be added to the OM node bank.*

VISPI discovery can help discovering the nodes in the vSphere environment (vCenter, ESXi hosts, virtual machines) and add them to the OM node bank automatically. By default, VISPI discovery does not add virtual machines to the node bank. To also add virtual machines to the node bank, change the corresponding setting in the policy `AUTO_ADDITION_SETTINGS` as below and redeploy it to the OAVA. The discovery runs every hour by default.

```
AutoAdd_Guests=true
```

Check VISPI documentation for details of VISPI discovery.

Note that, when the policy `VI-VMwareVCEventMonitor` cannot identify the name of the node that triggered the event, it will fill in the OAVA's name in the OM alert. So when searching for VMware events in the OM message browser, it is worth also looking for events that arrive under the OAVA name.

2 Detailed configuration steps

This section describes how to define the exact list of event types that should be collected and how these events should be forwarded to the OM server.

2.1 Managing event types

The policy `VI-VMwareVCEventTypes` defines the type of events that should be collected by OAVA. Events of a type that is not listed in the policy will be dropped.

The policy accepts a single event type per line, or two complementary events separated by a colon, e.g.:

```
HostShutdownEvent
VmPoweredOffEvent:VmPoweredOnEvent
```

Complementary events are also referred to as the *opted event* and the *normal event*; respectively `VmPoweredOffEvent` and `VmPoweredOnEvent` in above example. The opted events and normal events are handled in a slightly different manner in the policy `VI-VMwareVCEventMonitor`.

The default policy VI-VMwareVCEventTypes lists only a small subset of possible event types. This list should be extended with any desired additional event types. Check VMware documentation for a list of existing VMware event types. For instance:

https://pubs.vmware.com/vfabric5/index.jsp?topic=/com.vmware.vfabric.hyperic.4.6/Events.html.

2.2 Configuring OM alerts

The policy `VI-VMwareVCEventMonitor` consists of two rules.

The first rule (`Events from vCenter`) should not be modified. It reads the new events from the OAVA database and feeds them to the second rule (`Evaluate Events and send alert`). The second rule tests the event types and forwards matching events to the OM server. It contains:

- one condition per type of event that should be forwarded to the OM server
- two additional conditions for forwarding of events of type `AlarmStatusChangedEvent`
- one last catch-all condition

The catch-all condition, called "`Warning Alert - Event Occurred`", is in last position of the second rule. It matches and forwards events of any type listed in the policy `VI-VMwareVCEventTypes`, that were not matched by an earlier condition, except normal events which will be dropped.

So in the general case, there is no need to add conditions to the policy `VI-VMwareVCEventMonitor` when customizing the policy `VI-VMwareVCEventTypes`. When there is nevertheless a need to add a new condition for a specific event type, best is to copy an existing condition (for instance "`VM Disk Failed Event`"), customize it, and move it in front of the existing condition "`AlarmStatus Changed Event`".

The policy `VI-VMwareVCEventMonitor` also defines five parameters:

- The parameter `MessageGroup` is currently ignored (https://softwaresupport.hp.com/group/softwaresupport/search-result/-/facetsearch/document/KM01705141)
- The parameter `EventSource` exists for historical reasons only and is ignored.
- The parameters `AlarmFlag` and `AlarmFilterByText` are described in the next paragraph.
- The parameter `Debug` controls the logging level of the policy.
  - Set this parameter to 2 in order to generate debug information in file `/var/opt/OV/log/Infraspi.txt`.

2.3 Monitoring alarm state changes

VMware has a concept of alarms that fire when specific conditions are met. The state of the alarm is represented by the colors green (normal), yellow (warning) and red (alert). An alarm state change results into an event of type `AlarmStatusChangedEvent`, for instance:

```
Alarm 'Virtual machine cpu usage' changed from Green to Red
Alarm 'Virtual machine memory usage' changed from Yellow to Green
Alarm 'Host cpu usage' changed from Yellow to Red
```

OAVA/VISPI can collect and forward these events like any other event. The policy `VI-VMwareVCEventMonitor` implements specific logic to recognize the alarm name and state. It sends OM alerts with according severity: 'warning' for the yellow and red states, 'normal' for the green state. Additionally, the 'normal' alert is setup to acknowledge the corresponding 'warning' alert through message key correlation.

To enable forwarding of alarm state change events, take following steps:

- Add `AlarmStatusChangedEvent` to the policy `VI-VMwareVCEventType` and redeploy it to the OAVA
- Set parameter `AlarmFlag` to `true` in the policy `VI-VMwareVCEventMonitor` and redeploy it to the OAVA

Alarm state change events can also be filtered based on the alarm name. The alarm name is the part of the event text enclosed between quotes. The alarm names for the three above example alarm state change events are:

```
Virtual machine cpu usage
Virtual machine memory usage
Host cpu usage
```

For instance, if one is only interested in alarm state change events from the two first alarms shown above, the policy parameter `AlarmFilterByText` should be set to:

```
Virtual machine cpu usage,Virtual machine memory usage
```

As shown in this example, the parameter can take several alarm names, separated by a coma.

When the policy parameter `AlarmFilterByText` is not blank, only the alarm state change events that match this parameter are forwarded to the OM server. In the above example, alarm state change event for the alarm `'Host cpu usage'` would be dropped.

It is also possible to specify the severity of the alert that will be sent to the OM server, for instance:

```
Virtual machine cpu usage:Minor,Virtual machine memory usage:Major
```

When the severity is not specified, it defaults to warning.

2.4 Internationalized alarms

The event text of events of type `AlarmStatusChangedEvent` follows a standard format that makes it possible for the policy `VI-VMwareVCEventMonitor` to parse the text for specific details about the alarm. When VMware generates these events in a non-English language, the format is no longer standard and the parsing will fail. VISPI supports forwarding of events of type `AlarmStatusChangedEvent` only when VMware generates these events in English.

To ensure that the vCenter generates the events in English, configure `en_US` [English (United States)] for the default locale of the Windows user account which the vCenter Server is running under (note that the vCenter Server Virtual Appliance only supports `en_US`). For more details, check this document:

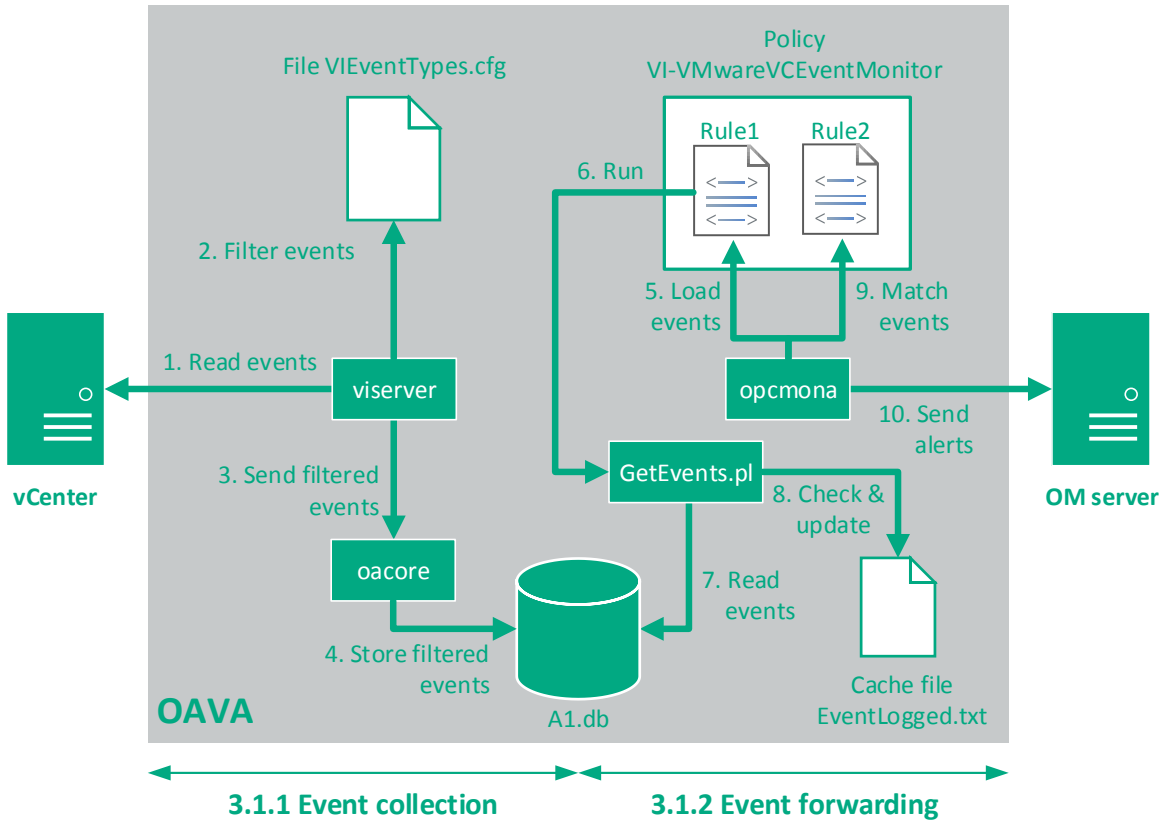
http://kb.vmware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&externalId=2121646

Hewlett Packard Enterprise

3 Troubleshooting steps

3.1 Architecture

Understanding the architecture of a solution helps to set the right troubleshooting steps when something goes wrong. Below picture describes the high level architecture of VMware event forwarding with OAVA and VISPI.



This picture shows that VMware event forwarding depends on two main steps:

3.1.1 Event collection

The Java process `viserver` is the main actor for these steps.

It reads new events from the vCenter (1) at intervals specified by the XPL setting `Virt_Node_Coll_Interval` in namespace `opsagt` (expressed in seconds).

The new events are filtered (2) by comparing their type to the event types listed in the file `VIEventTypes.cfg` (located in directory `/var/opt/conf/vispi/configuration`). This file contains a copy of the content of the policy `VI-VMwareVCEventTypes` and is automatically updated whenever the policy is redeployed. Events of a type that is not listed in the file are dropped.

The process `viserver` then sends the filtered events (3) to the process `oacore` that stores them into the OAVA database (4).

These steps can be followed in the logfile `/var/opt/perf/status.viserver`, for instance:

```
# grep -i events status.viserver
...
INFO  [2015-12-09 09:25:59,410] : Thread[pool-1-thread-212,5,main] Collecting and
sending events for vcenter1.gale2.net
INFO  [2015-12-09 09:25:59,425] : Thread[pool-218-thread-1,5,main] Starting to
collect events in EventCollector from Wed Dec 09 08:48:44 CET 2015
INFO  [2015-12-09 09:26:02,533] : Thread[pool-218-thread-1,5,main] Number of events
received are 3
```

The above line indicates that viserver received 3 new events from the vCenter.

```
INFO  [2015-12-09 09:26:02,850] : Thread[pool-218-thread-1,5,main] 1 interesting
events found by EventCollector
```

The above line indicates that 1 of the 3 events matches an event type listed in the file `VIEventTypes.cfg`.

```
DEBUG [2015-12-09 09:26:03,037] : Thread[pool-218-thread-1,5,main] Sent the
collected events [1] to localhost
```

The above line indicates that viserver sent the filtered event to process oacore.

```
INFO   [2015-12-09 09:26:03,524] : Thread[pool-1-thread-212,10,main] Waiting for
NodeDS and events tasks to get completed...
```

The above line indicates that viserver continues to wait for node related metrics (the event collection task actually completed).

Note that one of above lines is only written to the logfile if DEBUG level is enabled. For more information on how to fix OAVA collection problems and how to enable DEBUG level logging, check the paper "How to achieve optimal performance and stability of OAVA" at:

https://softwaresupport.hp.com/group/softwaresupport/search-result/-/facetsearch/document/KM01013486.

3.1.2 Event forwarding

The process opcmona is the main actor for these steps.

It executes the rules of the policy `VI-VMwareVCEventMonitor` at regular intervals. At each interval it executes the embedded script of the first rule (5). This script calls the perl script `GetEvents.pl` (6) in the instrumentation directory (`/var/opt/OV/bin/instrumentation`). `GetEvents.pl` uses OAVA's perl API to read the latest collected events from the OAVA database (7) through a time-based SQL query.

There are some further details worth noting about `GetEvents.pl`, that have significant implications on how to troubleshoot event forwarding problems:

> *The time range used in the SQL query is calculated as:*
>
> *now - 2 x* `Virt_Node_Coll_Interval` *< Event time < now*
>
> *Where* `Virt_Node_Coll_Interval` *is an XPL setting in the namespace* `opsagt`*.*
>
> *This explains why it is important that the policy interval be set equal to* `Virt_Node_Coll_Interval`*. This time range guarantees that all collected events will be processed by the policy, even if the policy runs with a slight delay and/or out of synch with the event collection driven by the process* `viserver` *(which is generally the case).*

However, as a direct consequence of this time range calculation, some events will be read twice. So to avoid sending duplicate alerts, the script `GetEvents.pl` *keeps track of recently read events in the temporary file* `/var/opt/OV/tmp/vispi/EventLogged.txt` *(8). If it finds that an event was already read during the previous interval, it will drop it.*

Consequently, running the script `GetEvents.pl` *manually (for instance to confirm that it successfully reads events from the database) would interfere with the next execution of the policy and cause it to miss alerts. Alternatively, one can simply look into the file* `/var/opt/OV/tmp/vispi/EventLogged.txt` *for the list of events that were loaded during the last execution of the policy.*

When the execution of the first rule completes, the new events have been loaded by the policy. The process opcmona will now execute the second rule for each loaded event in order to match the event with one of the conditions of this rule (9). For any match, an alert is sent to the OM server (10).

Note that versions of `GetEvents.pl` prior to 1114.1000 (delivered with hotfix HOTFIX_SPI_VI_2014-11-23_2) used a very different logic that could not cope reliably with the fact that the event collection (driven by `viserver`) and the event forwarding (driven by `opcmona`) are asynchronous processes.

3.2 Testing VMware event forwarding

When testing VMware event forwarding, it can be handy to know a simple procedure to generate a VMware event on demand. One simple way to trigger a VMware event is to open the console of a virtual machine in the vSphere client: right-click on the virtual machine label in the virtual machine inventory and select Open Console. This will trigger an event of type `VmRemoteConsoleConnectedEvent`.

This type of event is however not included in the default policy `VI-VMwareVCEventTypes`. So before testing with this type of event, the event type `VmRemoteConsoleConnectedEvent` should be added to the policy `VI-VMwareVCEventTypes` and the policy should be redeployed to the OAVA.

3.3 Differentiating between collection and forwarding problems

When troubleshooting VMware event forwarding problems, the general first step is to verify whether the problem is located in the event collection step or in the event forwarding step. This can best be done with the command `/opt/OV/contrib/OpC/oava_list_events.sh` that is part of the contributed troubleshooting tools available from https://softwaresupport.hp.com/group/softwaresupport/search-result/-/facetsearch/document/KM01570431.

This tool lists all collected events from the OAVA database, for instance:

```
# /opt/OV/contrib/OpC/oava_list_events.sh
ObservedTime,InstanceName,EventType,Description
2015-12-07 15:52:00,vm100A,AlarmStatusChangedEvent,"Alarm 'Virtual machine cpu
usage' on vm100A changed from Green to Red"
2015-12-07 15:57:03,esxhost3.gale2.net,AlarmStatusChangedEvent,"Alarm 'Host
connection and power state' on esxhost3.gale2.net changed from Green to Red"
2015-12-07 15:59:12,vm110A,VmRemoteConsoleConnectedEvent,"Remote console connected
to vm110A on host esxhost1.gale2.net"
```

If the tool does not list the expected events, the problem should be troubleshooted from the perspective of event collection (process `viserver` and policy `VI-VMwareVCEventTypes`). If the tool lists the expected events, the problem should be troubleshooted from the perspective of event forwarding (process `opcmona`, policy `VI-VMwareVCEventMonitor` and OM server).

3.4 Troubleshooting checkpoints

The list of checkpoints below helps resolve most VMware event forwarding problems with OAVA and VISPI:

- Check if the desired VMware events are collected by OAVA, in order to differentiate between collection problems and forwarding problems (see "Differentiating between collection and forwarding problems").

If the VMware events are not collected by OAVA or partially collected, check these points:

- The policy `VI-VMwareEventTypes` should contain the list of desired event types and should be deployed to the OAVA. This can be verified both with the command `ovpolicy -l` and by checking that the file `/var/opt/OV/conf/vispi/configuration/VIEventTypes.cfg` exists and contains the same event types as the policy.
- The OAVA version should be at least 11.14 with a recent OAVA hotfix.
- The OAVA collection should be healthy. Use the command `oaconfig -lt` to check that the collection is happening at every 1 or 5 minutes interval depending on the XPL setting `Virt_Node_Coll_Interval` in namespace `opsagt`, e.g.:

  ```
  # oaconfig -lt
  vcenter1.gale2.net : Data Collection Completed (0:0:51 ago)
  ```

  In case of collection problems refer to the paper "How to achieve optimal performance and stability of OAVA" at https://softwaresupport.hp.com/group/softwaresupport/search-result/-/facetsearch/document/KM01013486.

If the VMware events are collected by OAVA but some or all are missing in the OM message browser, check these points:

- The policy `VI-VMwareVCEventMonitor` should be deployed to the OAVA
  - The version of the policy should be at least 1114.1000
  - The interval of the policy should match the XPL setting `Virt_Node_Coll_Interval` in namespace `opsagt`
- The nodes that trigger VMware events should be configured in the OM node bank.
- Name resolution for VMware nodes that trigger VMware events should be consistent on the OAVA and on the OM server.
- If the Alarm state change events are missing, check that the vCenter generates these events with English text. This can best be checked with the command `/opt/OV/contrib/OpC/oava_list_events.sh`. If the event text is not in English, see "Internationalized alarms".
- If a problem with the policy `VI-VMwareVCEventMonitor` is suspected, change the policy parameter `Debug` to 2 and redeploy the policy. Check the logs in `/var/opt/OV/log/Infraspi.txt`. Logging is however relatively limited.
- Use XPL or classic tracing of the processes opcmona and opcmsga on the OAVA.
- Use typical troubleshooting/tracing steps for message problems on the OM server (remember that OAVA may be sending events that get lost somewhere in the OM server message flow).

Send feedback to thierry.ledent@hpe.com