Response Automation for System Events

This chapter describes how to configure the Embedded Event Manager (EEM).

About the EEM

The EEM service enables you to debug problems and provides general purpose logging for troubleshooting. There are two components: events to which the EEM responds or listens, and event manager applets that define actions as well as the events to which the EEM responds. You may configure multiple event manager applets to respond to different events and perform different actions.

Supported Events

The EEM supports the following events:

- Syslog—The ASA uses syslog message IDs to identify syslog messages that trigger an event manager applet. You may configure multiple syslog events, but the syslog message IDs may not overlap within a single event manager applet.
- Timers—You may use timers to trigger events. You may configure each timer only once for each event manager applet. Each event manager applet may have up to three timers. The three types of timers are the following:
  - Watchdog (periodic) timers trigger an event manager applet after the specified time period following the completion of the applet actions and restart automatically.
  - Countdown (one-shot) timers trigger an event manager applet once after the specified time period and do not restart unless they are removed, then re-added.
  - Absolute (once-a-day) timers cause an event to occur once a day at a specified time, and restart automatically. The time-of-day format is in hh:mm:ss.

    You may configure only one timer event of each type for each event manager applet.

- None—The none event is triggered when you run an event manager applet manually using the CLI or ASDM.
- Crash—The crash event is triggered when the ASA crashes. Regardless of the value of the **output** command, the **action** commands are directed to the crashinfo file. The output is generated before the **show tech** command.

Actions on Event Manager Applets

When an event manager applet is triggered, the actions on the event manager applet are performed. Each action has a number that is used to specify the sequence of the actions. The sequence number must be unique within an event manager applet. You may configure multiple actions for an event manager applet. The commands are typical CLI commands, such as **show blocks**.

Output Destinations

You may send the output from the actions to a specified location using the **output** command. Only one output value may be enabled at any one time. The default value is **output none**. This value discards any output from the **action** commands. The command runs in global configuration mode as a user with privilege level 15 (the highest). The command may not accept any input, because it is disabled. You may send the output of the **action** CLI commands to one of three locations:

- **None**, which is the default and discards the output
- **Console**, which sends the output to the ASA console
- **File**, which sends the output to a file. The following four file options are available:
  - **Create a unique file**, which creates a new, uniquely named file each time that an event manager applet is invoked
  - **Create/overwrite a file**, which overwrites a specified file each time that an event manager applet is invoked.
  - **Create/append to a file**, which appends to a specified file each time that an event manager applet is invoked. If the file does not yet exist, it is created.
  - **Create a set of files**, which creates a set of uniquely named files that are rotated each time that an event manager applet is invoked.

Guidelines for the EEM

This section describes guidelines and limitations that you should check before configuring the EEM.

Context Mode Guidelines

Not supported in multiple context mode.

Additional Guidelines

- During a crash, the state of the ASA is generally unknown. Some commands may not be safe to run during this condition.
- The name of an event manager applet may not contain spaces.
- You cannot modify the None event and Crashinfo event parameters.
- Performance may be affected because syslog messages are sent to the EEM for processing.
- The default output is **output none** for each event manager applet. To change this setting, you must enter a different output value.
- You may have only one output option defined for each event manager applet.

Configure the EEM

Configuring the EEM consists of the following tasks:

Procedure

Step 1 Create an Event Manager Applet and Configure Events, on page 3.
Step 2 Configure an Action and Destinations for Output from an Action, on page 5.
Step 3 Run an Event Manager Applet, on page 7.
Step 4 Track Memory Allocation and Memory Usage, on page 7.

Create an Event Manager Applet and Configure Events

To create an event manager applet and configure events, perform the following steps:

Procedure

Step 1 Create an event manager applet and enter event manager applet configuration mode.

event manager applet *name*

Example:

```
ciscoasa(config)# event manager applet exampleapplet1
```

The *name* argument may be up to 32 alphanumeric characters long. Spaces are not allowed.

To remove an event manager applet, enter the **no** form of this command.

Step 2 Describe an event manager applet.

description *text*

Example:

```
ciscoasa(config-applet)# description applet1example
```

The *text* argument may be up to 256 characters long. You may include spaces in description text if it is placed within quotes.

Step 3 To configure a specified event, enter one of the following commands. To remove the configured event, enter the **no** form of each of the commands.

- To configure a syslog event, identify a single syslog message or a range of syslog messages that trigger an event manager applet.

  event syslog id *nnnnnn* [-*nnnnnn*] [**occurs** *n*] [**period** *seconds*]

  Example:

  ```
  ciscoasa(config-applet)# event syslog id 106201
  ```

  The *nnnnnn* argument identifies the syslog message ID. The **occurs** *n* keyword-argument pair indicates the number of times that the syslog message must occur for an event manager applet to be invoked. The default is 1 occurrence every 0 seconds. Valid values are from 1 - 4294967295. The **period** *seconds* keyword-argument pair indicates the number of seconds in which the event must occur, and limits how frequently an event manager applet is invoked to at most once in the configured period. Valid values are from 0 - 604800. A value of 0 means that no period is defined.

- To configure an event to occur once per configured period and restart automatically.

  event timer watchdog time *seconds*

  Example:

  ```
  ciscoasa(config-applet)# event timer watchdog time 30
  ```

  The number of seconds may range from 1 - 604800.

- To configure an event to occur once and not restart unless it is removed, then re-added.

  event timer countdown time *seconds*

  Example:

  ```
  ciscoasa(config-applet)# event timer countdown time 60
  ```

  The number of seconds may range from 1 - 604800. Use the **no** form of this command remove a countdown timer event.

  Note This timer reruns when you reboot if it is the startup configuration.

- To configure an event to occur once a day at a specified time and restart automatically.

  event timer absolute time *hh:mm:ss*

  Example:

```
ciscoasa(config-applet)# event timer absolute time 10:30:20
```

The time-of-day format is in hh:mm:ss. The time range is from 00:00:00 (midnight) to 23:59:59.

- Trigger a crash event when the ASA crashes.

  event crashinfo

  Example:

```
ciscoasa(config-applet)# event crashinfo
```

  Regardless of the value of the **output** command, the **action** commands are directed to the crashinfo file. The output is generated before the **show tech** command.

Configure an Action and Destinations for Output from an Action

To configure an action and specific destinations for sending output from an action, perform the following steps:

Procedure

Step 1 Configure an action on an event manager applet.

action *n* **cli command** *"command"*

Example:

```
ciscoasa(config-applet)# action 1 cli command "show version"
```

The *n* option is an action ID. Valid IDs range from 0 - 4294967295. The value of the *command* option must be in quotes; otherwise, an error occurs if the command consists of more than one word. The command runs in global configuration mode as a user with privilege level 15 (the highest). The command may not accept any input, because it is disabled. Use the **noconfirm** option if the command has it available.

Step 2 Choose one of the available output destination options. Use the **no** form of each command to remove an output destination,

- The **None** option discards any output from the **action** commands, which is the default setting:

  output none

  Example:

```
ciscoasa(config-applet)# output none
```

- The **Console** option sends the output of the **action** commands to the console.

  output console

Example:

```
ciscoasa(config-applet)# output console
```

Note Running this command affects performance.

- The **New File** option sends the output of the **action** commands to a new file for each event manager applet that is invoked.

  output file new

  Example:

  ```
  ciscoasa(config-applet)# output file new
  ```

  The filename has the format of eem-*applet-timestamp*.log, in which *applet* is the name of the event manager applet and *timestamp* is a dated time stamp in the format of YYYYMMDD-hhmmss.

- The **New Set of Rotated Files** option creates a set of files that are rotated. When a new file is to be written, the oldest file is deleted, and all subsequent files are renumbered before the first file is written.

  output file rotate *n*

  Example:

  ```
  ciscoasa(config-applet)# output file rotate 50
  ```

  The newest file is indicated by 0, and the oldest file is indicated by the highest number (*n*-1). The *n* option is the rotate value. Valid values range from 2 - 100. The filename format is eem-*applet-x*.log, in which *applet* is the name of the applet, and *x* is the file number.

- The **Single Overwritten File** option writes the **action** command output to a single file, which is overwritten every time.

  output file overwrite *filename*

  Example:

  ```
  ciscoasa(config-applet)# output file overwrite examplefile1
  ```

  The *filename* argument is a local (to the ASA) filename. This command may also use FTP, TFTP, and SMB targeted files.

- The **Single Appended File** option writes the **action** command output to a single file, but that file is appended to every time.

  output file append *filename*

  Example:

  ```
  ciscoasa(config-applet)# output file append examplefile1
  ```

The *filename* argument is a local (to the ASA) filename.

Run an Event Manager Applet

To run an event manager applet, perform the following steps:

Procedure

Run an event manager applet.

event manager run *applet*

Example:

```
ciscoasa# event manager run exampleapplet1
```

If you run an event manager applet that has not been configured with the **event none** command, an error occurs. The *applet* argument is the name of the event manager applet.

Track Memory Allocation and Memory Usage

To log memory allocation and memory usage, perform the following steps:

Procedure

Step 1 Enable memory logging.

memory logging [1024-4194304] [**wrap**] [**size** [1-2147483647]] [**process** *process-name*] [**context** *context-name*]

Example:

```
ciscoasa(config)# memory logging 202980
```

The only required argument is the number of entries in the memory logging buffer. The **wrap** option tells the memory logging utility to save the buffer when it wraps. It can only be saved once.

If the memory logging buffer wraps multiple times, it can be overwritten. When the buffer wraps, a trigger is sent to the event manager to enable saving of the data. The **size** option monitors a particular size. The **process** option monitors a particular process.

Note The Checkheaps process is completely ignored as a process because it uses the memory allocator in a non-standard way.

The **context** option records memory logging for a given virtual context by the given name.

To change memory logging parameters, you must disable it, then reenable it.

Step 2 Display the memory logging results.

```
show memory logging [brief | wrap]
show memory logging include [address] [caller] [operator] [size] [process] [time] [context]
```

Example:

```
ciscoasa# show memory logging
Number of free                          6
Number of calloc                        0
Number of malloc                        8
Number of realloc-new                   0
Number of realloc-free                  0
Number of realloc-null                  0
Number of realloc-same                  0
Number of calloc-fail                   0
Number of malloc-fail                   0
Number of realloc-fail                  0
Total operations 14
Buffer size: 50 (3688 x2 bytes)
process=[ci/console] time=[13:26:33.407] oper=[malloc]
addr=0x00007fff2cd0a6c0 size=72 @ 0x00000000016466ea 0x0000000002124542
0x000000000131911a 0x0000000000442bfd process=[ci/console] time=[13:26:33.407] oper=[free]
addr=0x00007fff2cd0a6c0 size=72 @ 0x00000000021246ef 0x00000000013193e8
0x0000000000443455 0x0000000001318f5b
process=[CMGR Server Process] time=[13:26:35.964] oper=[malloc]
addr=0x00007fff2cd0aa00 size=16 @ 0x00000000016466ea 0x0000000002124542
0x000000000182774d 0x000000000182cc8a process=[CMGR Server Process]
time=[13:26:35.964] oper=[malloc]
addr=0x00007fff224bb9f0 size=512 @ 0x00000000016466ea 0x0000000002124542
0x0000000000bfef9a 0x0000000000bff606 process=[CMGR Server Process]
time=[13:26:35.964] oper=[free]
addr=0x00007fff224bb9f0 size=512 @ 0x00000000021246ef 0x0000000000bff3d8
0x0000000000bff606 0x000000000182ccb0
process=[CMGR Server Process] time=[13:26:35.964] oper=[malloc]
addr=0x00007fff224b9460 size=40 @ 0x00000000016466ea 0x0000000002124542
0x0000000001834188 0x000000000182ce83
process=[CMGR Server Process] time=[13:26:37.964] oper=[free]
addr=0x00007fff2cd0aa00 size=16 @ 0x00000000021246ef 0x0000000001827098
0x000000000182c08d 0x000000000182c262 process=[CMGR Server Process]
time=[13:26:37.964] oper=[free]
addr=0x00007fff224b9460 size=40 @ 0x00000000021246ef 0x000000000182711b
0x000000000182c08d 0x000000000182c262 process=[CMGR Server Process]
time=[13:26:38.464] oper=[malloc]
addr=0x00007fff2cd0aa00 size=16 @ 0x00000000016466ea 0x0000000002124542
0x000000000182774d 0x000000000182cc8a process=[CMGR Server Process]
time=[13:26:38.464] oper=[malloc]
addr=0x00007fff224bb9f0 size=512 @ 0x00000000016466ea 0x0000000002124542
0x0000000000bfef9a 0x0000000000bff606 process=[CMGR Server Process]
time=[13:26:38.464] oper=[free]
addr=0x00007fff224bb9f0 size=512 @ 0x00000000021246ef 0x0000000000bff3d8
0x0000000000bff606 0x000000000182ccb0
process=[CMGR Server Process] time=[13:26:38.464] oper=[malloc]
addr=0x00007fff224b9460 size=40 @ 0x00000000016466ea 0x0000000002124542
0x0000000001834188 0x000000000182ce83
process=[ci/console] time=[13:26:38.557] oper=[malloc]
addr=0x00007fff2cd0a6c0 size=72 @ 0x00000000016466ea 0x0000000002124542
0x000000000131911a 0x0000000000442bfd process=[ci/console] time=[13:26:38.557] oper=[free]
addr=0x00007fff2cd0a6c0 size=72 @ 0x00000000021246ef 0x00000000013193e8
```

```
0x0000000000443455 0x0000000001318f5b

ciscoasa# show memory logging include process operation size
Number of free                          6
Number of calloc                        0
Number of malloc                        8
Number of realloc-new                   0
Number of realloc-free                  0
Number of realloc-null                  0
Number of realloc-same                  0
Number of calloc-fail                   0
Number of malloc-fail                   0
Number of realloc-fail                  0
Total operations 14
Buffer size: 50 (3688 x2 bytes)
process=[ci/console] oper=[malloc] size=72 process=[ci/console] oper=[free]
size=72 process=[CMGR Server Process] oper=[malloc] size=16
process=[CMGR Server Process] oper=[malloc] size=512 process=[CMGR Server Process]
oper=[free] size=512 process=[CMGR Server Process] oper=[malloc] size=40
process=[CMGR Server Process] oper=[free] size=16 process=[CMGR Server Process]
oper=[free] size=40 process=[CMGR Server Process] oper=[malloc] size=16
process=[CMGR Server Process] oper=[malloc] size=512 process=[CMGR Server Process]
oper=[free] size=512 process=[CMGR Server Process] oper=[malloc] size=40
process=[ci/console] oper=[malloc] size=72 process=[ci/console]
oper=[free] size=72 ciscoasa# show memory logging brief
Number of free                          6
Number of calloc                        0
Number of malloc                        8
Number of realloc-new                   0
Number of realloc-free                  0
Number of realloc-null                  0
Number of realloc-same                  0
Number of calloc-fail                   0
Number of malloc-fail                   0
Number of realloc-fail                  0
Total operations 14
Buffer size: 50 (3688 x2 bytes)
```

Without any options, **show memory logging** displays statistics and then the recorded operations. The **brief** option shows only statistics. The **wrap** option shows the buffer upon wrap, then purges the data so that duplicate data does not appear or get saved. The **include** option includes only the specified fields in the output. You can specify the fields in any order, but they always appear in the following order:

a. Process

b. Time

c. Context (unless in single mode)

d. Operation (free/malloc/etc.)

e. Address

f. Size

g. Callers

The output format is:

process=[XXX] time=[XXX] context=[XXX] oper=[XXX] address=0xXXXXXXXX size=XX @ XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX

Up to four caller addresses appear. The types of operations are listed in the output (Number of...) shown in the example.

Step 3 Respond to memory logging wrap events.

event memory-logging-wrap

Example:

```
ciscoasa(config)# event manager applet memlog
ciscoasa(config)# event memory-logging-wrap
ciscoasa(config)# action 0 cli command "show memory logging wrap"
ciscoasa(config)# output file append disk0:/memlog.log
```

The example shows an applet that records all memory allocations. When wrap is enabled for memory logging, the memory logger sends an event to the event manager to trigger configured applets.

Examples for the EEM

The following example shows an event manager applet that records block leak information every hour and writes the output to a rotating set of log files, keeping a day's worth of logs:

```
ciscoasa(config)# event manager applet blockcheck
ciscoasa(config-applet)# description "Log block usage"
ciscoasa(config-applet)# event timer watchdog time 3600
ciscoasa(config-applet)# output rotate 24
ciscoasa(config-applet)# action 1 cli command "show blocks old"
```

The following example shows an event manager applet that reboots the ASA every day at 1 am, saving the configuration as needed:

```
ciscoasa(config)# event manager applet dailyreboot
ciscoasa(config-applet)# description "Reboot every night"
ciscoasa(config-applet)# event timer absolute time 1:00:00
ciscoasa(config-applet)# output none
ciscoasa(config-applet)# action 1 cli command "reload save-config noconfirm"
```

The following example shows event manager applets that disable the given interface between midnight and 3 am.

```
ciscoasa(config)# event manager applet disableintf
ciscoasa(config-applet)# description "Disable the interface at midnight"
ciscoasa(config-applet)# event timer absolute time 0:00:00
ciscoasa(config-applet)# output none
ciscoasa(config-applet)# action 1 cli command "interface GigabitEthernet 0/0"
ciscoasa(config-applet)# action 2 cli command "shutdown"
ciscoasa(config-applet)# action 3 cli command "write memory"

ciscoasa(config)# event manager applet enableintf
ciscoasa(config-applet)# description "Enable the interface at 3am"
```

```
ciscoasa(config-applet)# event timer absolute time 3:00:00
ciscoasa(config-applet)# output none
ciscoasa(config-applet)# action 1 cli command "interface GigabitEthernet 0/0"
ciscoasa(config-applet)# action 2 cli command "no shutdown"
ciscoasa(config-applet)# action 3 cli command "write memory"
```

Monitoring the EEM

See the following commands to monitor the EEM.

- **clear configure event manager**

  This command removes the event manager running configuration.

- **clear configure event manager applet** *appletname*

  This command removes the named event manager applet from the configuration.

- **show counters protocol eem**

  This command shows the counters for the event manager.

- **show event manager**

  This command shows information about the configured event manager applets, including hit counts and when the event manager applets were last invoked.

- **show memory logging, show memory logging include**

  These commands show statistics about the memory allocations and memory usage.

- **show running-config event manager**

  This command shows the running configuration of the event manager.

History for the EEM

Table 1: History for the EEM

Feature Name	Platform Releases	Description
Embedded Event Manager (EEM)	9.2(1)	The EEM service enables you to debug problems and provides general purpose logging for troubleshooting. There are two components: events to which the EEM responds or listens, and event manager applets that define actions as well as the events to which the EEM responds. You may configure multiple event manager applets to respond to different events and perform different actions. We introduced or modified the following commands: **event manager applet**, **description**, **event syslog id**, **event none**, **event timer** {**watchdog time** *seconds* | **countdown time** *seconds* | **absolute time** *hh:mm:ss*}, **event crashinfo**, **action cli command**, **output** {**none** | **console** | **file** {**append** *filename* | **new** | **overwrite** *filename* | **rotate** *n*}}, **show running-config event manager**, **event manager run**, **show event manager**, **show counters protocol eem**, **clear configure event manager**, **debug event manager**, **debug menu eem**.
Memory tracking for the EEM	9.4(1)	We have added a new debugging feature to log memory allocations and memory usage, and to respond to memory logging wrap events. We introduced or modified the following commands: **memory logging**, **show memory logging**, **show memory logging include**, **event memory-logging-wrap**.