JUNIPER NETWORKS

Junos® OS

Event Automation

Release
11.4

Published: 2011-11-08
Revision 1

Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

Junos® OS Event Automation

Revision History
October 2011—Revision 1; initial release

The information in this document is current as of the date listed in the revision history.

END USER LICENSE AGREEMENT

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at http://www.juniper.net/support/eula.html. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

List of Figures

List of Tables

PART 1

Overview

CHAPTER 1

Event Policy Overview

Event Notifications and Policies Overview

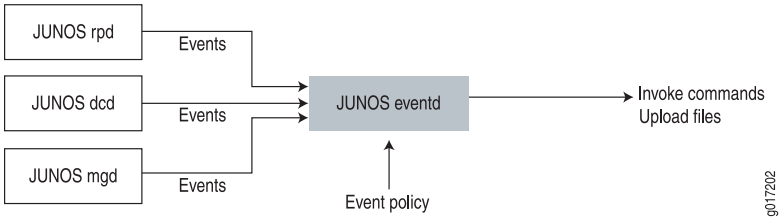
To diagnose a fault or error condition on a device, you need relevant information about the state of the platform. You can derive state information from *event notifications*. Event notifications are system log messages and SNMP traps. A Junos OS process called the *event process* (eventd) receives event notifications—henceforth simply called *events*—from other Junos OS processes.

Timely diagnosis and intervention can correct error conditions and keep the device in operation. After the eventd process receives events, *event policies* instruct the eventd process to select specific events, correlate the events, and perform a set of actions. These actions can either help you diagnose a fault or take corrective action. For example, the eventd process can upload device files to a given destination and issue operational mode commands.

Events can originate as SNMP traps or system log messages. The event process receives event messages from other Junos OS processes, such as the routing protocol process (rpd) and the management process (mgd). Depending on the custom event policy you configure, eventd listens for specific events and in response to these events might create a log file, invoke a Junos command, or invoke an event script. When an event script is invoked, event details are passed to the event script in the form of XML inputs.

Figure 1 on page 3 shows how the event process (eventd) interacts with other Junos OS processes.

Figure 1: Interaction of eventd Process with Other Junos OS Processes



How Event Policies Work

An event policy is an if-then-else construct. It defines actions to be executed by the eventd process on receipt of an event. You can configure multiple policies to be processed for an event. The policies are executed in the order in which they appear in the configuration. For each policy, you can configure multiple actions. The actions are also executed in the order in which they appear in the configuration.

To view a list of the events that can be referenced in an event policy, issue the **help syslog ?** command:

```
user@host> help syslog ?
Possible completions:
  <syslog-tag>         System log tag
  ACCT_ACCOUNTING_FERROR  Error occurred during file processing
  ACCT_ACCOUNTING_FOPEN_ERROR  Open operation failed on file
  ...
```

You can filter the output of a search by using the pipe (**|**) symbol. The following example lists the filters that can be used with the pipe symbol:

```
user@host> help syslog | ?
Possible completions:
  count                Count occurrences
  display              Show additional kinds of information
  except               Show only text that does not match a pattern
  find                 Search for first occurrence of pattern
  hold                 Hold text without exiting the --More-- prompt
  last                 Display end of output only
  match                Show only text that matches a pattern
  no-more              Don't paginate output
  request              Make system-level requests
  resolve              Resolve IP addresses
  save                 Save output text to file
  trim                 Trim specified number of columns from start of line
```

For more information about using the pipe symbol, see the *Junos OS CLI User Guide*.

You can also list multiple events as you configure the event policy. To view a partial list of the events that can be referenced in an event policy, issue the **set event-options policy *policy-name* events ?** configuration mode command:

```
[edit]
user@host# set event-options policy policy-name events ?
Possible completions:
 <event>
 [            Open a set of values
 acct_accounting_ferror
 acct_accounting_fopen_error
 ...
```

Some of the system log messages that you can reference in an event policy are not listed in the output of the **set event-options policy *policy-name* events ?** command. For information about referencing these system log messages in your event policies, see "Using Nonstandard System Log Messages to Trigger Event Policies" on page 17.

In addition, you can reference internally generated events, which are discussed in "Generating Internal Events to Trigger Event Policies" on page 16.

In response to events, the eventd process can correlate two or more events based on a policy, and execute the following actions:

- Ignore the event—Do not generate a system log message for this event and do not process any further policy instructions for this event.
- Upload a file—Upload a file to a specified destination. You can specify a transfer delay, so that, on receipt of an event, the upload of the file begins after the configured transfer delay. For example, to upload a core file, a transfer delay can ensure that the core file has been completely generated before the upload begins.
- Execute Junos OS operational mode commands—Execute commands on receipt of an event. The XML or text output of these commands is stored in a file, which is then uploaded to a specified URL. You can include variables in the command that allow data from the triggering event to be automatically included in the command syntax.
- Execute Junos OS event scripts—Execute event scripts on receipt of an event. Event scripts are Extensible Stylesheet Transformation (XSLT) or Stylesheet Language Alternative Syntax (SLAX) scripts that you write to perform any function available through Junos XML or Junos XML protocol remote procedure calls (RPCs). Additionally, you can pass to an event script a set of arguments that you define. A script can build and run an operational mode command, receive the command output, inspect the output, and determine the next appropriate action. This process can be repeated until the source of the problem is determined. The output of the scripts is stored in a file, which is then uploaded to a specified URL. You can include variables in the arguments to the scripts that allow data from the triggering event to be incorporated into the script.
- Raise an SNMP trap.

CHAPTER 2

Event Scripts Overview

Event Scripts Overview

Event Script Programming Overview

Junos OS event scripts are triggered automatically by defined event policies in response to a system event and can instruct Junos OS to take immediate action. Event scripts automate network and device management and troubleshooting. Event scripts can perform functions available through the remote procedure calls (RPCs) supported by either Junos XML management protocol or the Junos Extensible Markup Language (XML) API. Event scripts are executed by the event process (eventd).

Event scripts allow you to do the following:

- Automatically diagnose and fix problems in the network
- Monitor the overall status of a device.
- Run automatically as part of an event policy that detects periodic error conditions
- Change the configuration in response to a problem

Event scripts are based on the Junos XML management protocol and the Junos XML API, which are discussed in Junos XML API and Junos XML Management Protocol Overview. Event scripts can be written in either the Extensible Stylesheet Language Transformations (XSLT) or Stylesheet Language Alternative Syntax (SLAX) scripting language. Event scripts use XPath to locate the operational objects to be inspected and XSLT constructs to specify the actions to perform on the located operational objects. The actions can change the output or execute additional commands based on the output. For more information about XPath and XSLT, see XPath Overview.

How Event Scripts Work

Event scripts initiate operational commands when triggered by an event policy. When an event policy is triggered, this policy forwards event details to the event script. You enable event scripts by listing the names of one or more event script files within the **[edit**

event-options event-script] hierarchy level. These scripts contain instructions that execute operational mode commands and inspect the output automatically. Event scripts are invoked within an event policy. For information about event policies, see "Event Notifications and Policies Overview" on page 3 and "Executing Event Scripts in an Event Policy" on page 28.

You can use event scripts to generate changes to the device configuration by including the **<load-configuration>** tag element. Because the changes are loaded before the standard validation checks are performed, they are validated for correct syntax, just like statements already present in the configuration before the script is applied. If the syntax is correct, the configuration is activated and becomes the active, operational device configuration.

Related Documentation

- XSLT Overview
- SLAX Overview

PART 2

Configuration

CHAPTER 3

Triggering an Event Policy

Using Correlated Events to Trigger an Event Policy

You can configure a policy that correlates two or more events. If the correlated events occur as specified, they cause particular actions to be taken. For example, you might want to issue certain operational mode commands when a **UI_CONFIGURATION_ERROR** event is generated within five minutes (300 seconds) after a **UI_COMMIT_PROGRESS** event. As another example, you might want to upload a particular file if a **DCD_INTERFACE_DOWN** event is generated two times within a 60-second interval.

To configure a policy that correlates events, include the following statements at the **[edit event-options]** hierarchy level:

```
[edit event-options]
policy policy-name {
   events [ events ];
   within seconds {
      events [ events ];
      not events [ events ];
      trigger (on | after | until) event-count;
   }
   attributes-match {
      event1.attribute-name equals event2.attribute-name;
      event.attribute-name matches regular-expression;
      event1.attribute-name starts-with event2.attribute-name;
   }
   then {
      ...
   }
}
```

In the **events** statement, you can list multiple events. To view a list of the events that can be referenced in an event policy, issue the **set event-options policy *policy-name* events ?** configuration mode command:

```
user@host# set event-options policy policy-name events ?
Possible completions:
 <event>
 [               Open a set of values
 acct_accounting_ferror
 acct_accounting_fopen_error
 ...
```

Some of the system log messages that you can reference in an event policy are not listed in the output of the **set event-options policy *policy-name* events ?** command. For information about referencing these system log messages in your event policies, see "Using Nonstandard System Log Messages to Trigger Event Policies" on page 17.

In addition, you can reference internally generated events, which are discussed in "Generating Internal Events to Trigger Event Policies" on page 16.

The actions configured in the **then** statement are executed only if certain conditions are met, which you specify in the **within** and **attributes-match** statements.

You can configure a policy that is executed only if a specified event occurs within a specified time interval after another event. You do this by including the **within *seconds* events** statement. The policy is executed only if one or more of the events in the first **events** statement occur within a configured number of seconds after one or more of the events in the **within *seconds* events** statement. The number of seconds can be from 60 through 604,800. The **not** statement causes the policy to be executed only if the events do not occur within the configured time interval.

For example, the following policy is executed if ***event3***, ***event4***, or ***event5*** occurs within 60 seconds after ***event1*** or ***event2*** occurs:

```
[edit event-options]
policy 1 {
   events [ event3 event4 event5 ];
   within 60 events [ event1 event2 ];
   then {
      ...
   }
}
```

The **attributes-match** statement correlates two events as follows:

- ***event1.attribute-name* equals *event2.attribute-name***—Execute the policy only if the specified attribute of ***event1*** equals the specified attribute of ***event2***.
- ***event.attribute-name* matches *regular-expression***—Execute the policy only if the specified attribute of ***event*** matches a regular expression. For more information, see "Using Regular Expressions to Refine the Set of Events That Trigger a Policy" on page 15.
- ***event1.attribute-name* starts-with *event2.attribute-name***—Execute the policy only if the specified attribute of ***event1*** starts with the specified attribute of ***event2***.

If the **attributes-match** statement includes the **equals** or **starts-with** options, or if it includes a **matches** option that includes a clause for an event that is not specified at the **[edit event-options policy *policy-name* events]** hierarchy level, you must include one or more **within** statements in the same policy configuration.

Starting with Junos OS Release 11.1, you can use event policy variables within the **attributes-match** statement to differentiate between a trigger event attribute and a correlated event attribute. The double dollar sign (**$$**) notation represents the event that is triggering a policy, and **{$$.*attribute-name*}** resolves to the value of the attribute of the triggering event. Triggering events are those that you configure at the **[edit event-options policy *policy-name* events]** hierarchy level. For correlating events, the single dollar sign with the event name (**$*event***) notation represents the most recent event that matches the event name, and **{$*event.attribute-name*}** resolves to the value of the attribute of the correlated event.

In the following example, the policy will execute the actions under the **then** statement if four or more commits are performed within a 5-minute period, and the username of one or more of the correlated events is the same as the username of the trigger event.

```
policy multiple-commits {
  events ui_commit;
  attributes-match {
    {$$.user-name} equals {$ui_commit.user-name};
  }
  within 300 {
    trigger after 3;
    events ui_commit;
  }
  then ...
}
```

To view a list of all event attributes that you can reference, issue the **help syslog *event*** operational mode command. The output of this command shows the event attributes in angle brackets (< >). The following output shows that three attributes can be referenced for the **ACCT_ACCOUNTING_SMALL_FILE_SIZE** event: **filename**, **file-size**, and **record-size**.

```
user@host> help syslog ACCT_ACCOUNTING_SMALL_FILE_SIZE
Name:           ACCT_ACCOUNTING_SMALL_FILE_SIZE
Message:       File <filename> size (<file-size>) is smaller than record size
(<record-size>)
```

You can filter the output of a search by using the pipe (**|**) symbol. The following example lists the filters that can be used with the pipe symbol:

```
user@host> help syslog | ?
Possible completions:
  count                Count occurrences
  display              Show additional kinds of information
  except               Show only text that does not match a pattern
  find                 Search for first occurrence of pattern
  hold                 Hold text without exiting the --More-- prompt
  last                 Display end of output only
  match                Show only text that matches a pattern
  no-more              Don't paginate output
  request              Make system-level requests
  resolve              Resolve IP addresses
```

```
save                Save output text to file
trim                Trim specified number of columns from start of line
```

For more information about using the pipe symbol, see the *Junos OS CLI User Guide*.

Another way to view the attributes you can reference is by issuing the **set attributes-match *event*?** command at the **[edit event-options policy *policy-name*]** hierarchy level, as shown in the following example:

```
[edit event-options policy p1]
user@host# set attributes-match acct_accounting_small_file_size?
Possible completions:
 <from-event-attribute>  First attribute to compare
 acct_accounting_small_file_size.filename
 acct_accounting_small_file_size.filesize
 acct_accounting_small_file_size.record-size
```

NOTE: In this set command, there is no space between the event name and the question mark (?).

Related Documentation

- Representing the Correlating Event in an Event Policy on page 14
- Triggering an Event Policy Based on Event Count on page 15
- Using Regular Expressions to Refine the Set of Events That Trigger a Policy on page 15
- attributes-match on page 65
- policy on page 79
- not on page 76
- then on page 82
- within on page 88

Representing the Correlating Event in an Event Policy

As described in "Configuring an Event Policy to Execute Operational Mode Commands" on page 25, the double dollar sign ($$) notation represents the event that is triggering a policy. Triggering events are those that you configure at the **[edit event-options policy *policy-name* events]** hierarchy level.

As described in "Using Correlated Events to Trigger an Event Policy" on page 11, you can configure a policy that is executed only if a specified event occurs within a specified time interval after another event. You do this by including the **within *seconds* events** statement at the **[edit event-options policy *policy-name*]** hierarchy level:

```
[edit event-options policy policy-name ]
events [ events ];
within seconds events [ events ];
```

The policy is executed only if one or more of the events at the **[edit event-options policy *policy-name* events]** hierarchy level occur within a configured number of seconds after one or more of the events in the **within *seconds* events** statement.

For correlating events, the single dollar sign with the event name (**$*event***) notation represents the most recent event that matches the event name. The dollar sign with the asterisk (**$***) notation represents the most recent event that matches any of the correlating events.

For a configuration example, see "Example: Representing the Correlating Event in an Event Policy" on page 58.

Triggering an Event Policy Based on Event Count

You can configure an event policy to be triggered if an event or set of events occurs a specified number of times within a specified time period.

To do this, include the optional **trigger** statement at the **[edit event-options policy *policy-name* within *seconds*]** hierarchy level:

```
[edit event-options policy policy-name within seconds]
trigger (after | on | until) event-count;
```

The software counts the number of times the triggering event occurs. A triggering event can be any event configured at the **[edit event-options policy *policy-name* events]** hierarchy level. You can configure the following options:

- **after *event-count***—The policy is executed when the number of matching events received equals ***event-count*** plus one.
- **on *event-count***—The policy is executed when the number of matching events received equals ***event-count***.
- **until *event-count***—The policy is executed each time a matching event is received and stops being executed when the number of matching events received equals ***event-count***.

For a configuration example, see "Example: Triggering a Policy Based on Event Count" on page 60.

Using Regular Expressions to Refine the Set of Events That Trigger a Policy

You can use regular expression matching to specify more exactly which events cause a policy to be executed.

To specify the text string that must appear in an event attribute for the policy to be executed, include the **matches** statement at the **[edit event-options policy *policy-name* attributes-match]** hierarchy level, and specify the regular expression which the event attribute must match:

```
[edit event-options policy policy-name attributes-match]
event.attribute-name matches regular-expression;
```

When you specify the regular expression, use the notation defined in POSIX Standard 1003.2 for extended (modern) UNIX regular expressions. Explaining regular expression syntax is beyond the scope of this document. Table 1 on page 16 specifies which character or characters are matched by some of the regular expression operators that you can use in the **matches** statement. In the descriptions, the term *term* refers to

either a single alphanumeric character or a set of characters enclosed in square brackets, parentheses, or braces.

NOTE: The matches statement is not case-sensitive.

Table 1: Regular Expression Operators for the matches Statement

Operator	Matches
. (period)	One instance of any character except the space.
* (asterisk)	Zero or more instances of the immediately preceding term.
+ (plus sign)	One or more instances of the immediately preceding term.
? (question mark)	Zero or one instance of the immediately preceding term.
| (pipe)	One of the terms that appear on either side of the pipe operator.
! (exclamation point)	Any string except the one specified by the expression, when the exclamation point appears at the start of the expression. Use of the exclamation point is specific to Junos OS.
^ (caret)	The start of a line, when the caret appears outside square brackets. One instance of any character that does not follow it within square brackets, when the caret is the first character inside square brackets.
$ (dollar sign)	The end of a line.
[ ] (paired square brackets)	One instance of one of the enclosed alphanumeric characters. To indicate a range of characters, use a hyphen (-) to separate the beginning and ending characters of the range. For example, **[a-z0-9]** matches any letter or number.
( ) (paired parentheses)	One instance of the evaluated value of the enclosed term. Parentheses are used to indicate the order of evaluation in the regular expression.

For a configuration example, see "Example: Controlling Event Policy Using a Regular Expression" on page 54.

Generating Internal Events to Trigger Event Policies

Internal events are events that you create to trigger a policy to be executed. They are not generated by Junos OS processes, and they do not have any associated system log messages. You can generate an internal event based on a time interval or the time of day.

To generate an event, include the following statements at the **[edit event-options]** hierarchy level:

```
[edit event-options]
generate-event event-name {
  time-interval seconds;
  time-of-day hh:mm:ss;
}
```

In the **time-interval** statement, configure a frequency, in seconds, with which to repeatedly generate an event. The time interval can range from 60 through 2,592,000 seconds.

In the **time-of-day** statement, configure a time of day for the event to occur. Use the format ***hh*:*mm*:*ss***.

NOTE: If you modify the system time by issuing the **set date** operational mode command, we recommend that you also issue the **commit full** or the **restart event-process** command. Otherwise, an internal event based on the time of day might not be generated at the configured time.

For example, if you configure an internal event to be generated at 15:55:00, and then you modify the system time from 15:47:17 to 15:53:00, the event is generated when the system time is approximately 16:00 instead of at the configured time, 15:55:00. You can correct this problem by issuing the **commit full** or the **restart event-process** command.

You can configure up to 10 internal events. If you attempt to commit a configuration with more than 10 internal events, Junos OS generates an error, and the commit fails.

For configuration examples, see "Example: Generating an Internal Event" on page 56.

Using Nonstandard System Log Messages to Trigger Event Policies

Some of the system log messages that you can reference in an event policy are not listed in the output of the **set event-options policy *policy-name* events ?** command. These system log messages have an event ID and a **message** attribute. Event IDs are based on the origin of the message, as shown in Table 2 on page 17.

Table 2: Event ID by System Log Message Origin

Event IDs	Origin
SYSTEM	Messages from Junos daemons and utilities
KERNEL	Messages from the Junos kernel
PIC	Messages from physical interface cards (PICs)
PFE	Messages from the Packet Forwarding Engine
LCC	On a TX Matrix router, messages from a line-card chassis (LCC)

Table 2: Event ID by System Log Message Origin *(continued)*

Event IDs	Origin
SCC	On a TX Matrix router, messages from a switch-card chassis (SCC)

To base your event policy on the event types shown in Table 2 on page 17, include the **events *event-id*** statement and the **attributes-match** statement with the ***event-id*.message matches "*message*"** attribute at the **[edit event-options policy *policy-name*]** hierarchy level:

```
[edit event-options policy policy-name]
events event-id;
attributes-match {
  event-id.message matches "message";
}
```

For a configuration example, see "Example: Using Nonstandard System Log Messages to Trigger an Event Policy" on page 62.

CHAPTER 4

Configuring Event Policy File Archiving

Defining Destinations for File Archiving by Event Policies

When the action in an event policy generates output files, you might want to save them for later analysis. Similarly, you might want to save system files (such as system log or core files) from the time an event occurs. You must configure one or more *destinations* to which files can be uploaded for archiving.

To define destinations, include the **destinations** statement at the **[edit event-options]** hierarchy level:

```
[edit event-options]
destinations {
  destination-name {
    archive-sites {
      url <password password>;
    }
    transfer-delay seconds;
  }
}
```

You can then reference configured destinations in an event policy. For information about referencing destinations, see "Configuring an Event Policy to Upload Files" on page 20 and "Configuring an Event Policy to Execute Operational Mode Commands" on page 25.

The optional **transfer-delay** statement allows you to specify the number of seconds the event process (eventd) waits before beginning to upload a file or multiple files. A transfer delay allows you to make sure a large file, such as a core file, is completely generated before the upload begins. For more information, see "Configuring the Delay Before Files Are Uploaded by an Event Policy" on page 21.

In the **archive-sites** statement, you can specify a destination as a Hypertext Transfer Protocol (HTTP) URL, FTP URL, or secure copy (scp)-style remote file specification. URLs of the type **file://** are not supported; however, local device directories are supported (for example, **/var/tmp/**). When you specify the archive site, do not add a forward slash

(/) to the end of the URL. The format for the destination filename is *device-name_filename_YYYYMMDD_HHMMSS*.

Optionally, you can specify a plain-text password for login into an archive site.

For a configuration example, see "Example: Correlating Events Based on Receipt of Other Events Within a Specified Time Interval" on page 55.

Configuring an Event Policy to Upload Files

Various types of files are useful in diagnosing an event. These files include system log files, core files, and configuration files. When an event occurs, you can upload relevant files to a specified location for analysis.

To configure a policy that uploads files, include the following statements at the **[edit event-options]** hierarchy level:

```
[edit event-options]
policy policy-name {
  events [ events ];
  then {
    upload filename (filename | committed) destination destination-name {
      retry-count number retry-interval seconds;
      transfer-delay seconds;
      user-name username;
    }
  }
}
```

When an event policy uploads files for analysis, the files are named and time-stamped in the following format to ensure unique filenames:

device-name_filename_YYYYMMDD_HHMMSS

If a policy uploads multiple files within a 1-second period, the software gives each file a unique number as well, as follows:

device-name_filename_YYYYMMDD_HHMMSS_number

The number can be from 001 through 999. For example, if you have an event policy with output filename **rpd-messages** on **device1**, and this event policy is executed three times in 1 second, the files are named as follows:

- **device1_rpd-messages_20070623_132333**
- **device1_rpd-messages_20070623_132333_001**
- **device1_rpd-messages_20070623_132333_002**

In the **events** statement, you can list multiple events. If one or more of the listed events occurs, the upload action is executed. To view a partial list of the events that can be referenced in an event policy, issue the **set event-options policy *policy-name* events ?** configuration mode command:

```
[edit]
user@host# set event-options policy policy-name events ?
Possible completions:
```



```
<event>
[           Open a set of values
acct_accounting_ferror
acct_accounting_fopen_error
...
```

Some of the system log messages that you can reference in an event policy are not listed in the output of the **set event-options policy *policy-name* events ?** command. For information about referencing these system log messages in your event policies, see "Using Nonstandard System Log Messages to Trigger Event Policies" on page 17.

In addition, you can reference internally generated events, which are discussed in "Generating Internal Events to Trigger Event Policies" on page 16.

If desired, you can include multiple **upload** statements, one for each type of file to be archived. In the **filename** statement, specify a file or multiple files to be uploaded. You can specify multiple files with one **filename** configuration statement (sometimes called *filename globbing*). For example, to upload all files that are located in the **/var/log** directory and that start with the **messages** string, include the following statement:

```
upload filename /var/log/messages*;
```

To upload the committed configuration file, include the **upload filename committed destination *destination-name*** statement at the **[edit event-options policy *policy-name* then]** hierarchy level:

```
[edit event-options policy policy-name then]
upload filename committed destination destination-name;
```

For the **destination** statement, specify a destination name that you configured at the **[edit event-options destinations]** hierarchy level. For more information, see "Defining Destinations for File Archiving by Event Policies" on page 19.

Configuring the Delay Before Files Are Uploaded by an Event Policy

A transfer delay allows you to specify the number of seconds the event process (eventd) waits before beginning to upload a file or multiple files. A transfer delay allows you to ensure that a large file, such as a core file, is completely generated before the upload begins.

As described in "Defining Destinations for File Archiving by Event Policies" on page 19, you can associate a transfer delay with a destination. If you associate a transfer delay with a destination, the transfer delay applies to all file upload actions that use the destination.

In the following example, the ***some-dest*** destination is common for both event policies, **policy1** and **policy2**. A transfer delay of 2 seconds is associated with the **some-dest** destination and applies to uploading the output files to the destination for both event policies.

```
[edit event-options]
policy policy1 {
    events e1;
    then {
    execute-commands {
```

```
        commands {
            "show version";
        }
        output-filename command-output.txt;
        destination some-dest;
    }
}
policy policy2 {
    events e2;
    then {
        event-script bar.xsl {
            output-filename event-script-output.txt;
            destination some-dest;
        }
    }
}
destinations {
    some-dest {
        transfer-delay 2;
        archive-sites {
            "http://robot@my.big.com/foo/moo" password "password";
            "http://robot@my.little.com/foo/moo" password "password";
        }
    }
}
```

Suppose you have multiple event policy actions that use the same destination. For some of these event policy actions, you want a transfer delay, and for other event policy actions you want no transfer delay. To assign a transfer delay to a single event policy action, include the optional **transfer-delay** statement for each action:

```
transfer-delay seconds;
```

You can include this statement at the following hierarchy levels:

- **[edit** event-options policy *policy-name* then event-script *filename* destination *destination-name***]**
- **[edit** event-options policy *policy-name* then execute-commands destination *destination-name***]**
- **[edit** event-options policy *policy-name* then upload **filename (***filename* **| committed) destination** *destination-name***]**

If you configure a transfer delay at the **[edit event-options destinations** *destination-name***]** hierarchy level, and you also configure a transfer delay for the event policy action, the resulting transfer delay is the sum of the two:

```
Total transfer-delay =
transfer-delay (destination) + transfer-delay (event-policy-action)
```

For a configuration example, see "Example: Assigning a Transfer Delay to an Event Policy Action" on page 51.

Configuring an Event Policy to Retry the File Upload Action

Transient network problems can cause a file upload operation to fail. When this happens, you might want to retry the file upload operation. By default, if the file upload operation fails for any reason, the event policy does not retry the upload operation.

To configure the policy to retry a file upload operation, include the optional **retry-count** and **retry-interval** statements:

retry-count *number* retry-interval *seconds*;

You can include these statements at the following hierarchy levels:

- **[edit** event-options policy ***policy-name*** then event-script ***filename*** destination ***destination-name*****]**
- **[edit** event-options policy ***policy-name*** then execute-commands destination ***destination-name*****]**
- **[edit** event-options policy ***policy-name*** then upload **filename (***filename* **| committed) destination** ***destination-name*****]**

The **retry-count** statement sets the number of times the policy retries the upload operation if the upload fails. The default value for the **retry-count** statement is 0 and the maximum is 10.

If you include the **retry-count** statement, you can also include the **retry-interval** statement, which sets the time interval (in seconds) between each retry.

For a configuration example, see "Example: Retrying the File Upload Action" on page 59.

CHAPTER 5

Configuring Event Policy Actions

Configuring an Event Policy to Execute Operational Mode Commands

Operational mode commands request that the device running Junos OS perform an operation or provide diagnostic output. They allow you to view statistics and information about a device's current operating status. They also allow you to take corrective actions, such as restarting software processes, taking a PIC offline and back online, switching to redundant interfaces, and adjusting Label Switching Protocol (LSP) bandwidth. For more information about operational mode commands, see the following references:

- *Junos OS Interfaces Command Reference*
- *Junos OS Routing Protocols and Policies Command Reference*
- *Junos OS System Basics and Services Command Reference*

You can configure a policy that causes operational mode commands to be issued and the output of those commands to be uploaded to a specified location for analysis.

To configure such a policy, include the following statements at the **[edit event-options]** hierarchy level:

```
[edit event-options]
policy policy-name {
  events [ events ];
  then {
    execute-commands {
      commands {
        "command";
      }
      output-filename filename;
      output-format (text | xml);
      destination destination-name;
    }
```

```
    }
}
```

In the **events** statement, you can list multiple events. If one or more of the listed events occurs, the operational mode commands are issued. To view a list of the events that can be referenced in an event policy, issue the **set event-options policy *policy-name* events ?** configuration mode command:

```
[edit]
user@host# set event-options policy policy-name events ?
Possible completions:
 <event>
 [              Open a set of values
 acct_accounting_ferror
 acct_accounting_fopen_error
 ...
```

Some of the system log messages that you can reference in an event policy are not listed in the output of the **set event-options policy *policy-name* events ?** command. For information about referencing these system log messages in your event policies, see "Using Nonstandard System Log Messages to Trigger Event Policies" on page 17.

In addition, you can reference internally generated events, which are discussed in "Generating Internal Events to Trigger Event Policies" on page 16.

In the **commands** statement, you can issue multiple operational mode commands upon receipt of a specific event. Enclose each command in quotation marks (**" "**). The eventd process issues the commands in the order in which they appear in the configuration. For example, in the following configuration, the execution of **policy1** causes the **show interfaces** command to be issued first, followed by the **show chassis alarms** command:

```
[edit event-options policy policy1 then execute-commands]
user@host# show
commands {
   "show interfaces";
   "show chassis alarms";
}
```

You can include variables in the command to allow data from the triggering event to be automatically included in the command syntax. The eventd process replaces each variable with values contained in the event that triggers the policy. You can use command variables of the following forms:

- **{$$.*attribute-name*}**—The double dollar sign (**$$**) notation represents the event that is triggering a policy. When combined with an attribute name, the variable is replaced by the value of the attribute name in the triggering event. For example, **{$$.interface-name}** stands for the value of the **interface-name** attribute in the triggering event.
- **{$*event*.*attribute-name*}**—The **{$*event*.*attribute-name*}** notation represents the most recent event that matches the specified event. The variable is replaced by the value of the attribute name of the most recent event that matches ***event***. For example, when a policy issues the **show interfaces {$COSD_CHAS_SCHED_MAP_INVALID.interface-name}** command, the **{$COSD_CHAS_SCHED_MAP_INVALID.interface-name}** variable is substituted by the

interface-name attribute of the most recent **COSD_CHAS_SCHED_MAP_INVALID** event cached by the event process.

For a given event, you can view a list of event attributes that you can reference in an operational mode command by issuing the **help syslog *event-name*** command:

```
user@host> help syslog event-name
```

For example, in the following command output, text in angle brackets (< >) shows that **classifier-type** is an attribute of the **cosd_unknown_classifier** event:

```
user@host> help syslog cosd_unknown_classifier
Name:          COSD_UNKNOWN_CLASSIFIER
Message:       rtsock classifier type <classifier-type> is invalid
...
```

You can filter the output of a search by using the pipe (|) symbol. The following example lists the filters that can be used with the pipe symbol:

```
user@host# help syslog | ?
Possible completions:
  count                Count occurrences
  display              Show additional kinds of information
  except               Show only text that does not match a pattern
  find                 Search for first occurrence of pattern
  hold                 Hold text without exiting the --More-- prompt
  last                 Display end of output only
  match                Show only text that matches a pattern
  no-more              Don't paginate output
  request              Make system-level requests
  resolve              Resolve IP addresses
  save                 Save output text to file
  trim                 Trim specified number of columns from start of line
```

For more information about using the pipe symbol, see the *Junos OS CLI User Guide*.

Another way to view a list of event attributes is to issue the **set attributes-match *event*?** configuration mode command at the **[edit event-options policy *policy-name*]** hierarchy level:

```
[edit event-options policy policy-name]
user@host# set attributes-match event ?
```

For example, in the following command output, the ***event.attribute*** list shows that **classifier-type** is an attribute of the **cosd_unknown_classifier** event:

```
[edit event-options policy policy-name]
user@host# set attributes-match cosd_unknown_classifier?
Possible completions:
 <from-event-attribute>  First attribute to compare
 cosd_unknown_classifier.classifier-type
```

NOTE: In this set command, there is no space between the event name and the question mark (?).

To view a list of all event attributes that you can reference, issue the **set attributes-match ?** configuration mode command at the **[edit event-options policy *policy-name*]** hierarchy level:

```
[edit event-options policy policy-name]
user@host# set attributes-match ?
Possible completions:
  <from-event-attribute>  First attribute to compare
  acct_accounting_ferror
  acct_accounting_fopen_error
  ...
```

In the **output-filename** statement, assign the name of the file to which to write command output for the specified commands. The filename format is ***hostname_filename_YYYYMMDD_HHMMSS_index-number***.

For each uploaded file, a hostname and timestamp ensure that the uploaded files have unique filenames. If a policy is triggered multiple times in a 1-second period, an index number is added to ensure the filenames are unique. The index number range is 001 through 999.

For example, on a device named **r1**, if you configure the output filename to be **ifl-events**, and this event policy is triggered three times in 1 second, the files are named:

- **r1_ifl-events_20060623_132333**
- **r1_ifl-events_20060623_132333_001**
- **r1_ifl-events_20060623_132333_002**

By default, the command output format is Junos Extensible Markup Language (XML). To change this, include the **output-format text** statement. This causes the command output to be in formatted ASCII text.

In the **destination** statement, include the destination name that you configured at the **[edit event-options destinations]** hierarchy level. For more information, see "Defining Destinations for File Archiving by Event Policies" on page 19.

For a configuration example, see "Example: Correlating Events Based on Receipt of Other Events Within a Specified Time Interval" on page 55.

Executing Event Scripts in an Event Policy

Event scripts are Extensible Stylesheet Transformation (XSLT) or Stylesheet Language Alternative Syntax (SLAX) scripts that you write and that are run when triggered by an event policy. Event scripts can perform any function available through Junos XML or Junos XML protocol remote procedure calls (RPCs). Additionally, you can pass to an event script a set of arguments that you define.

A script can change the device configuration, build and run an operational mode command, receive the command output, inspect the output, and determine the next appropriate action. This process can be repeated until the source of the problem is determined. The script can then report the source of the problem to you on the CLI.

You can configure an event policy that causes event scripts to be run and the output of those scripts to be uploaded to a specified location for analysis.

To configure such a policy, include the following statements at the **[edit event-options]** hierarchy level:

```
[edit event-options]
policy policy-name {
  events [ events ];
  then {
    event-script filename {
      arguments {
        argument-name argument-value;
      }
      output-filename filename;
      output-format (text | xml);
      destination destination-name;
    }
  }
}
```

In the **events** statement, you can list multiple events. If one or more of the listed events occurs, the event script is executed. To view a list of the events that can be referenced in an event policy, issue the **set event-options policy *policy-name* events ?** configuration mode command:

```
[edit]
user@host# set event-options policy policy-name events ?
Possible completions:
 <event>
 [               Open a set of values
 acct_accounting_ferror
 acct_accounting_fopen_error
 ...
```

Some of the system log messages that you can reference in an event policy are not listed in the output of the **set event-options policy *policy-name* events ?** command. For information about referencing these system log messages in your event policies, see "Using Nonstandard System Log Messages to Trigger Event Policies" on page 17.

In addition, you can reference internally generated events, which are discussed in "Generating Internal Events to Trigger Event Policies" on page 16.

In the **event-script** statement, you can specify a script to be executed on receipt of an event. The eventd process runs the scripts in the order in which they appear in the configuration. The scripts that you reference in the **event-script** statement must be located in the **/var/db/scripts/event** directory on the device's hard drive or the **/config/scripts/event/** directory on the flash drive. Furthermore, the event scripts must be enabled at the **[edit event-options event-script file]** hierarchy level. For more information, see Storing and Enabling Scripts.

You can include arguments to the script as name/value pairs. You can include variables in the argument values to allow data from the triggering event to be automatically included in the argument. The eventd process replaces each variable with values contained in the event that triggers the policy. You can use variables of the following forms:

- **{$$*.attribute-name*}**—The double dollar sign ($$) notation represents the event that is triggering a policy. When combined with an attribute name, the variable is replaced by the value of the attribute name in the triggering event. For example, **{$$.interface-name}** stands for the value of the **interface-name** attribute in the triggering event.
- **{$*event.attribute-name*}**—The **{$*event.attribute-name*}** notation represents the most recent event that matches the specified event. The variable is replaced by the value of the attribute name of the most recent event that matches ***event***. For example, when you include an argument called **interface** and define the value as **{$COSD_CHAS_SCHED_MAP_INVALID.interface-name},** the **{$COSD_CHAS_SCHED_MAP_INVALID.interface-name}** variable is replaced by the **interface-name** attribute of the most recent **COSD_CHAS_SCHED_MAP_INVALID** event cached by the eventd process.

For a given event, you can view a list of event attributes that you can reference by issuing the **help syslog *event*** command:

```
user@host> help syslog event-name
```

For example, in the following command output, text in angle brackets (**< >**) shows attributes of the **COSD_CHASSIS_SCHEDULER_MAP_INVALID** event:

```
user@host> help syslog COSD_CHASSIS_SCHEDULER_MAP_INVALID
Name:        COSD_CHASSIS_SCHEDULER_MAP_INVALID
Message:     Chassis scheduler map incorrectly applied to interface
<interface-name>: <error-message>
...
```

You can filter the output of a search by using the pipe (**|**) symbol. The following example lists the filters that can be used with the pipe symbol:

```
user@host> help syslog | ?
Possible completions:
 count               Count occurrences
 display             Show additional kinds of information
 except              Show only text that does not match a pattern
 find                Search for first occurrence of pattern
 hold                Hold text without exiting the --More-- prompt
 last                Display end of output only
 match               Show only text that matches a pattern
 no-more             Don't paginate output
 request             Make system-level requests
 resolve             Resolve IP addresses
 save                Save output text to file
 trim                Trim specified number of columns from start of line
```

For more information about using the pipe symbol, see the *Junos OS CLI User Guide*.

Another way to view a list of event attributes is to issue the **set attributes-match *event* ?** configuration mode command at the **[edit event-options policy *policy-name*]** hierarchy level:

```
[edit event-options policy policy-name]
user@host# set attributes-match event ?
```

For example, in the following command output, the ***event.attribute*** list shows that **error-message** and **interface-name are** attributes of the **cosd_chassis_scheduler_map_invalid** event:

```
[edit event-options policy p1]
user@host# set attributes-match cosd_chassis_scheduler_map_invalid?
Possible completions:
 <from-event-attribute>  First attribute to compare
 cosd_chassis_scheduler_map_invalid.error-message
 cosd_chassis_scheduler_map_invalid.interface-name
```

In this **set** command, there is no space between the event name and the question mark (**?**).

To view a list of all event attributes that you can reference, issue the **set attributes-match ?** configuration mode command at the **[edit event-options policy *policy-name*]** hierarchy level:

```
[edit event-options policy policy-name]
user@host# set attributes-match ?
Possible completions:
 <from-event-attribute>  First attribute to compare
 acct_accounting_ferror
 acct_accounting_fopen_error
 ...
```

By default, the command output format is text. To change this, include the **output-format xml** statement.

In the optional **output-filename** statement, assign the name of the file to which to write script output for the specified script.

The filename format is ***hostname_filename_YYYYMMDD_HHMMSS_index-number***.

For each uploaded file, a hostname and timestamp are automatically added to the filename to ensure that the uploaded files have unique filenames. If a policy is triggered multiple times in a 1-second period, an index number is added to ensure the filenames are unique. The index number range is 001 through 999.

For example, on a device named **r1**, if you configure the output filename to be **ifl-events**, and this event policy is triggered three times in 1 second, the files are named:

- **r1_ifl-events_20060623_132333**
- **r1_ifl-events_20060623_132333_001**
- **r1_ifl-events_20060623_132333_002**

In the optional **destination** statement, include the destination name that you configured at the **[edit event-options destinations]** hierarchy level. For more information, see "Defining Destinations for File Archiving by Event Policies" on page 19.

For the **output-filename** and **destination** statements, there are four configuration scenarios:

- You can omit the **output-filename** and **destination** statements. This option makes sense when the event script has no output. For example, the event script might execute only **request** commands, which have no output.
- You can include the **destination** statement in the configuration. You omit the **output-filename** statement in the configuration and specify an output filename in the event script instead. The script output is sent to the destination specified in the configuration. If you do not include the **destination** statement in the configuration, the script output is not uploaded.

  In this scenario, the event policy extracts the filename from the event script. The event script writes the output filename as **STDOUT**. The XML syntax to use in the event script is:

  ```
  <output>
    <event-script-output-filename>filename</event-script-output-filename>
  </output>
  ```

  The **<event-script-output-filename>** element must be the first child tag within the **<output>** parent tag.

  On a device named **device2**, configure an event script action with a destination **host**, and omit the **output-filename** statement. Define the destination **host** as ftp://user@device1//tmp.

  In the **script1.xsl** event script, write the following output to **STDOUT**:

  ```
  <event-script-output-filename>/var/cmd.txt</event-script-output-filename>
  ```

  Configure the **policy1** event policy as follows:

  ```
  [edit event-options]
  policy policy1 {
    then {
      event-script script1.xsl {
        destination host;
      }
    }
  }
  destinations {
    host {
      archive-sites {
        "ftp://user@device1//tmp" password "$9$xkJNbYg4ZDH.oJ.fQnpuSyl";  ##
          SECRET-DATA***
      }
    }
  }
  ```

  In this example, the **/var/cmd.txt** file resides on device **device2**. The event policy uses the File Transfer Protocol (FTP) to upload this file to the **/tmp** directory on device **device1**.

  The event policy reads the output filename **/var/cmd.txt** from **STDOUT**. Then the event policy uploads the **/var/cmd.txt** file to the configured destination, which is the **/tmp** directory on device **device1**. The event policy renames the **/var/cmd.txt** file as **device2_cmd.txt_*YYYYMMDD_HHMMSS_range***.

- You can include the **output-filename** and **destination** statements. If you include the **output-filename** statement in the configuration, you must also include the **destination** statement in the configuration. In this case, the script output is redirected to the output filename specified in the configuration and is sent to the destination specified in the configuration.
- You can include the **output-filename** and **destination** statements, and also specify an output filename directly within the event script. If you do this, the output filename specified in the configuration overrides the output filename specified in the event script.

Configuring Event Policies to Ignore an Event

You can modify a policy to cause particular events to be ignored or to cause all events to be ignored during a particular time interval, to allow for maintenance for example. To configure such a policy, include the following statements at the **[edit event-options]** hierarchy level:

```
[edit event-options]
policy policy-name {
  events [ events ];
  then {
    ignore;
  }
}
```

In the **events** statement, you can list multiple events. To view a list of the events that can be referenced in an event policy, issue the **set event-options policy *policy-name* events ?** configuration mode command:

```
[edit]
user@host# set event-options policy policy-name events ?
Possible completions:
 <event>
 [              Open a set of values
 acct_accounting_ferror
 acct_accounting_fopen_error
 ...
```

Some of the system log messages that you can reference in an event policy are not listed in the output of the **set event-options policy *policy-name* events ?** command. For information about referencing these system log messages in your event policies, see "Using Nonstandard System Log Messages to Trigger Event Policies" on page 17.

In addition, you can reference internally generated events, which are discussed in "Generating Internal Events to Trigger Event Policies" on page 16.

If one or more of the listed events occur, a system log message for the event is not generated, and no further policies associated with this event are processed. If you include the **ignore** statement in a policy configuration, you cannot configure any other actions in the policy.

Changing the User Privilege Level for an Event Policy Action

Only superusers can configure event policies. Event policy actions—such as executing event scripts, uploading files, and executing operational mode commands—are by default executed by user **root**, because the event process (eventd) runs with **root** privileges.

In some cases, you might want an event policy action to be executed with restricted privileges. For example, suppose you configure an event policy that executes a script if an interface goes down. The script includes remote procedure calls (RPCs) to change the device configuration if certain conditions are present. If you do not want the script to change the configuration, you can execute the script with a restricted user profile. When the script is executed with a user profile that disallows configuration changes, the RPCs to change the configuration fail.

You can associate a user with each action in an event policy. If a user is not associated with an event policy action, then the action is executed as user **root** by default.

To specify the user under whose privileges an action is executed, include the **user-name** statement:

user-name *username*;

You can include this statement at the following hierarchy levels:

- **[edit** event-options policy ***policy-name*** then event-script ***filename*****]**
- **[edit** event-options policy ***policy-name*** then execute-commands**]**
- **[edit** event-options policy ***policy-name*** then upload **filename (*filename* | committed) destination *destination-name*]**

NOTE: **The username that you specify must be configured at the [edit system login] hierarchy level. For more information, see the** *Junos OS System Basics Configuration Guide*.

For a configuration example, see "Example: Associating an Optional User with an Event Policy Action" on page 53.

Configuring Event Policies to Raise SNMP Traps

SNMP *traps* enable an agent to notify a network management system (NMS) of significant events by way of an unsolicited SNMP message. You can configure an event policy action that raises traps for events based on system log messages. This enables notification of an SNMP trap-based application when an important system log message occurs. You can convert any system log message (for which there are no corresponding traps) into a trap. This is valuable if you use NMS traps rather than system log messages to monitor your network.

To configure a policy that raises a trap on receipt of an event, include the following statements at the **[edit event-options policy *policy-name*]** hierarchy level:

```
[edit event-options policy policy-name]
events [ events ];
then {
    raise-trap;
}
```

The MIB (**jnx-syslog.mib**) supports this policy action. For more information, see the *Junos OS SNMP MIBs and Traps Reference*.

For a configuration example, see "Example: Raising an SNMP Trap in Response to an Event" on page 58.

CHAPTER 6

Configuring Event Policy Privileges

Changing the User Privilege Level for an Event Policy Action

Only superusers can configure event policies. Event policy actions—such as executing event scripts, uploading files, and executing operational mode commands—are by default executed by user **root**, because the event process (eventd) runs with **root** privileges.

In some cases, you might want an event policy action to be executed with restricted privileges. For example, suppose you configure an event policy that executes a script if an interface goes down. The script includes remote procedure calls (RPCs) to change the device configuration if certain conditions are present. If you do not want the script to change the configuration, you can execute the script with a restricted user profile. When the script is executed with a user profile that disallows configuration changes, the RPCs to change the configuration fail.

You can associate a user with each action in an event policy. If a user is not associated with an event policy action, then the action is executed as user **root** by default.

To specify the user under whose privileges an action is executed, include the **user-name** statement:

```
user-name username;
```

You can include this statement at the following hierarchy levels:

- **[edit** event-options policy *policy-name* then event-script *filename***]**
- **[edit** event-options policy *policy-name* then execute-commands**]**
- **[edit** event-options policy *policy-name* then upload **filename (***filename* **| committed) destination** *destination-name***]**

NOTE: **The username that you specify must be configured at the [edit system login] hierarchy level. For more information, see the** *Junos OS System Basics Configuration Guide*.

For a configuration example, see "Example: Associating an Optional User with an Event Policy Action" on page 53.

CHAPTER 7

Creating and Executing Event Scripts

Required Boilerplate for Event Scripts

When you write event scripts, you use Extensible Stylesheet Language Transformations (XSLT) or Stylesheet Language Alternative Syntax (SLAX) tools provided with Junos OS. These tools include basic boilerplate that you must include in all event scripts, optional extension functions that accomplish scripting tasks more easily, and named templates that make scripts easier to read and write, which you import from a file called **junos.xsl**. For more information about the extension functions and templates, see Junos Script Automation: Extension Functions in the jcs Namespace Overview and Junos Script Automation: Named Templates in the jcs Namespace Overview.

Event scripts are based on Junos XML and Junos XML protocol tag elements. Like all XML elements, angle brackets enclose the name of a Junos XML or Junos XML protocol tag element in its opening and closing tags. This is an XML convention, and the brackets are a required part of the complete tag element name. They are not to be confused with the angle brackets used in the documentation to indicate optional parts of Junos OS CLI command strings.

You must include either XSLT or SLAX boilerplate as the starting point for all event scripts that you create. The XSLT boilerplate follows:

XSLT Boilerplate for Event Scripts

```
1  <?xml version="1.0" standalone="yes"?>
2  <xsl:stylesheet version="1.0"
3     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
4     xmlns:junos="http://xml.juniper.net/junos/*/junos"
```

```
5     xmlns:xnm="http://xml.juniper.net/xnm/1.1/xnm"
6     xmlns:jcs="http://xml.juniper.net/junos/commit-scripts/1.0">
7     <xsl:import href="../import/junos.xsl"/>

8     <xsl:template match="configuration">
9       <event-script-results>
          <!-- ... Insert your code here ... -->
10      </event-script-results>
11    </xsl:template>
      <!-- ... insert additional template definitions here ... -->
12  </xsl:stylesheet>
```

Line 1 is the Extensible Markup Language (XML) processing instruction (PI). This PI specifies that the code is written in XML using version 1.0. The XML PI, if present, must be the first noncomment token in the script file.

```
1   <?xml version="1.0"?>
```

Line 2 opens the style sheet and specifies the XSLT version as 1.0.

```
2   <xsl:stylesheet version="1.0"
```

Lines 3 through 6 list all the namespace mappings commonly used in event scripts. Not all of these prefixes are used in this example, but it is not an error to list namespace mappings that are not referenced. Listing all namespace mappings prevents errors if the mappings are used in later versions of the script.

```
3     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
4     xmlns:junos="http://xml.juniper.net/junos/*/junos"
5     xmlns:xnm="http://xml.juniper.net/xnm/1.1/xnm"
6     xmlns:jcs="http://xml.juniper.net/junos/commit-scripts/1.0">
```

Line 7 is an XSLT import statement. It loads the templates and variables from the file referenced as **../import/junos.xsl**, which ships as part of Junos OS (in the file **/usr/libdata/cscript/import/junos.xsl**). The **junos.xsl** file contains a set of named templates you can call in your scripts. These named templates are discussed in Junos Script Automation: Named Templates in the jcs Namespace Overview and Junos Named Templates in the jcs Namespace Summary.

```
7     <xsl:import href="../import/junos.xsl"/>
```

Line 8 defines a template that matches the **</>** element. The **<xsl:template match="/">** element is the root element and represents the top level of the XML hierarchy. All XML Path Language (XPath) expressions in the script must start at the top level. This allows the script to access all possible Junos XML and Junos XML protocol remote procedure calls (RPCs). For more information, see XPath Overview and xsl:template match="/" Template.

```
8     <xsl:template match="/">
```

After the **<xsl:template match="/">** tag element, the **<event-script-results>** and **</event-script-results>** container tags must be the top-level child tags, as shown in Lines 9 and 10.

```
9       <event-script-results>
          <!-- ... insert your code here ... -->
10      </event-script-results>
```


Line 11 closes the template.

```
11    </xsl:template>
```

Between Line 11 and Line 12, you can define additional XSLT templates that are called from within the **<xsl:template match="/">** template.

Line 12 closes the style sheet and the event script.

```
12  </xsl:stylesheet>
```

SLAX Boilerplate for Event Scripts

```
version 1.0;
ns junos = "http://xml.juniper.net/junos/*/junos";
ns xnm = "http://xml.juniper.net/xnm/1.1/xnm";
ns jcs = "http://xml.juniper.net/junos/commit-scripts/1.0";
import "../import/junos.xsl";

match / {
  <event-script-results> {
    /*
    * Insert your code here
    */
  }
}
```

Capturing and Using Event Details and Remote Execution Details in Event Scripts

When an event script is triggered by an event policy, the initiating event policy forwards a set of event details to the triggered event script. These event details can be captured, evaluated, and sent to log files as required. In addition, any configured remote execution details are also forwarded to the event script. The remote execution details allow the event script to invoke remote procedure calls as detailed in "Using RPCs and Operational Mode Commands in Event Scripts" on page 43.

Two types of event details are returned: triggered events and received events. *Triggered events* record the details of the event that triggered the policy. *Received events* record the details of events that happened before the triggering event. Event details and remote execution details are forwarded to the event script as XML in the following format:

```
<event-script-input>
  <junos-context>
    ...
  </junos-context>
  <trigger-event>
    <id>event-id</id>
    <type>event-type</type>
    <generation-time>timestamp</generation-time>
    <process>
      <name>process-name</name>
      <pid>pid</pid>
    </process>
    <hostname>hostname</hostname>
    <facility>facility-string</facility>
    <severity>severity-string</severity>
    <attribute-list>
```

```
      <attribute>
        <name>attribute-name</name>
        <value>attribute-value</value>
      </attribute>
    </attribute-list>
  </trigger-event>
  <received-events>
    <received-event>
      <id>event-id</id>
      <type>event-type</type>
      <generation-time>timestamp</generation-time>
      <process>
        <name>process-name</name>
        <pid>pid</pid>
      </process>
      <hostname>hostname</hostname>
      <facility>facility-string</facility>
      <severity>severity-string</severity>
      <attribute-list>
        <attribute>
          <name>attribute-name</name>
          <<value>attribute-value</value>
        </attribute>
      </attribute-list>
    </received-event>
  </received-events>
  <remote-execution-details>
    <remote-execution-detail>
      <remote-hostname>hostname</remote-hostname>
      <username>username</username>
      <passphrase>passphrase</passphrase>
    </remote-execution-detail>
  </remote-execution-details>
</event-script-input>
```

For information about the **<junos-context>** element, see Junos Script Automation: Global Parameters and Variables in the junos.xsl File.

For information about one method for using event details, see "Example: Limiting Event Script Output Based on a Specific Event Type" on page 57.

Mapping Operational Mode Commands and Output Fields to Junos XML Notation

In event scripts, you use tag elements from the Junos XML API to represent operational mode commands and output fields. For the Junos XML equivalent of commands and output fields, consult the *Junos XML API Operational Reference*.

You can also display Junos XML by directing the output from the **show** command to the **| display xml** command:

```
user@host> operational-mode-command | display xml
```

For example:

```
user@host> show interfaces terse | display xml
<rpc-reply xmlns:junos="http://xml.juniper.net/junos/10.0R1/junos">
    <interface-information
```

```
xmlns="http://xml.juniper.net/junos/10.0R10/junos-interface" junos:style="terse">
        <physical-interface>
            <name>dsc</name>
            <admin-status>up</admin-status>
            <oper-status>up</oper-status>
        </physical-interface>
        <physical-interface>
            <name>fxp0</name>
            <admin-status>up</admin-status>
            <oper-status>up</oper-status>
            <logical-interface>
                <name>fxp0.0</name>
                <admin-status>up</admin-status>
                <oper-status>up</oper-status>
                ...
```

Using RPCs and Operational Mode Commands in Event Scripts

Most Junos operational mode commands have XML equivalents. These XML commands can be executed remotely using the *remote procedure call* (RPC) protocol. All operational mode commands that have XML equivalents are listed in the *Junos XML API Operational Reference*.

RPC and operational mode command use in event scripts is discussed in more detail in the following sections:

Using RPCs in Event Scripts

You can invoke remote procedure calls (RPCs) in event scripts. For each event script that invokes RPCs, you must include the **remote-execution** statement at the **[edit event-options event-script file *filename*]** hierarchy level. For each remote device where an RPC is executed, you must configure the SSH host key information for the that device on the local device where the event script is executed.

For each remote device where an RPC is executed, specify the device hostname and the corresponding username and passphrase at the **remote-execution** level of the configuration hierarchy.

```
[edit event-options event-script file filename]
remote-execution {
  remote-hostname {
    username username;
    passphrase passphrase;
  }
}
```

The remote hostnames and their corresponding username and passphrase, in addition to the event details, are passed as input to the event script when it is triggered by an event policy. For more information about the details that are forwarded to the event script, see "Capturing and Using Event Details and Remote Execution Details in Event

Scripts" on page 41. A connection handle to the remote host is generated with the **jcs:open( )** function using ***remote-hostname***, ***username***, and ***passphrase*** as arguments; for more information about this function, see jcs:open( ) Function. The following code obtains a connection handle for each remote host included in the configuration:

XSLT Syntax

```
<xsl:for-each select="event-script-input/remote-execution-details">
  <xsl:variable name="d" select="remote-execution-detail"/>
  <xsl:variable name="connection"
      select="jcs:open($d/remote-hostname,$d/username,$d/passphrase)"/>
  ...
</xsl:for-each>
```

SLAX Syntax

```
for-each (event-script-input/remote-execution-details) {
  var $d = remote-execution-detail;
  var $connection = jcs:open($d/remote-hostname,$d/username,$d/passphrase);
   ...
}
```

To execute an RPC on a remote device, an SSH session must be established. In order for the script to establish the connection, you must either configure the SSH host key information for the remote device on the local device where the script will be executed, or the SSH host key information for the remote device must exist in the known hosts file of the user executing the script. For each remote device where the RPC is executed, configure the SSH host key information with one of the following methods:

- To configure SSH known hosts on the local device, include the **host** statement, and specify hostname and host key options for the remote device at the **[edit security ssh-known-hosts]** hierarchy level of the configuration.
- To manually retrieve SSH host key information, issue the **set security ssh-known-hosts fetch-from-server *hostname*** configuration mode command to instruct Junos OS to connect to the remote device and add the key.

```
user@host# set security ssh-known-hosts fetch-from-server router2

The authenticity of host 'router2 (10.10.10.1)' can't be established.
RSA key fingerprint is 30:18:99:7a:3c:ed:40:04:0f:fd:c1:57:7e:6b:f3:90.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'router2,10.10.10.1' (RSA) to the list of known
 hosts.
```

- To manually import SSH host key information from a file, use the **set security ssh-known-hosts load-key-file *filename*** configuration mode command and specify the known-hosts file.

```
user@host# set security ssh-known-hosts load-key-file /var/tmp/known_hosts

Import SSH host keys from trusted source /var/tmp/known_hosts ? [yes,no]
 (no) yes
```

- Alternatively, the user executing the script can log in to the local device, SSH to the remote device, and then manually accept the host key, which is added to that user's known hosts file. In the following example, root is logged in to **router1**. In order to execute a remote RPC on **router2**, root adds the host key of **router2** by issuing the **ssh router2** operational mode command and manually accepting the key.

```
root@router1> ssh router2

The authenticity of host 'router2 (10.10.10.1)' can't be established.
RSA key fingerprint is 30:18:99:7a:3c:ed:40:04:0f:fd:c1:57:7e:6b:f3:90.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'router2,10.10.10.1' (RSA) to the list of known
 hosts.
```

After configuring the required SSH host key and obtaining a connection handle to the remote device, the event script can execute RPCs with the **jcs:execute()** extension function on that remote device. This function is described in jcs:execute() Function. To use an RPC in the event script, include the RPC in a variable declaration and execute it with the **jcs:execute()** function; the connection handle and RPC variable declaration are provided as arguments to the **jcs:execute()** function.

XSLT Syntax

```
<xsl:variable name="rpc">
  <get-interface-information/> # Junos RPC for the show interfaces command
</xsl:variable>
<xsl:variable name="out" select="jcs:execute($connection, $rpc)"/>
```

SLAX Syntax

```
var $rpc = <get-interface-information>;
var $out = jcs:execute($connection, $rpc);
```

where **$connection** is the connection handle to the remote host. Any number of RPCs can be executed within the context of this connection handle until it is closed with the **jcs:close()** function.

Displaying the RPC Tags for a Command

To display the remote procedure call (RPC) XML tags for an operational mode command, enter **display xml rpc** after the pipe symbol (|).

The following example displays the RPC tags for the **show route** command:

```
user@host> show route | display xml rpc
<rpc-reply xmlns:junos="http://xml.juniper.net/junos/10.1I0/junos">
    <rpc>
        <get-route-information>
        </get-route-information>
    </rpc>
    <cli>
        <banner></banner>
    </cli>
</rpc-reply>
```

Using Operational Mode Commands in Event Scripts

Some operational mode commands do not have XML equivalents. If a command is not listed in the *Junos XML API Operational Reference*, it does not have an XML equivalent.

Another way to determine whether a command has an XML equivalent is to issue the command followed by the **| display xml** command:

```
user@host> operational-mode-command | display xml
```

If the output includes only tag elements like **<output>**, **<cli>**, and **<banner>**, the command might not have an XML equivalent. In the following example, the output indicates that the **show host** command has no XML equivalent:

```
user@host> show host hostname | display xml
<rpc-reply xmlns:junos="http://xml.juniper.net/junos/10.0R1/junos">
   <output>
       ...
    </output>
    <cli>
        <banner></banner>
    </cli>
</rpc-reply>
```

NOTE: For some commands that have an XML equivalent, the output of the piped **| display xml** command does not include tag elements other than **<output>**, **<cli>**, and **<banner>** only because the relevant feature is not configured. For example, the **show services cos statistics forwarding-class** command has an XML equivalent that returns output in the **<service-cos-forwarding-class-statistics>** response tag, but if the configuration does not include include any statements at the **[edit class-of-service]** hierarchy level then there is no actual data for the **show services cos statistics forwarding-class | display xml** command to display. The output is something like this:

```
user@host> show services cos statistics forwarding-class | display xml
<rpc-reply xmlns:junos="http://xml.juniper.net/junos/8.3I0/junos">

    <cli>

        <banner></banner>

    </cli>

</rpc-reply>
```

For this reason, the information in the *Junos XML API Operational Reference* is normally more reliable.

An event script can include commands that have no XML equivalent. Use the **<command>**, **<xsl:value-of>**, and **<output>** elements in the script, as shown in the following code snippet. This snippet is expanded and fully described in Example: Displaying DNS Hostname Information Using an Op Script.

```
<xsl:variable name="query">
  <command>
    <xsl:value-of select="concat('show host ', $hostname)"/>
  </command>
</xsl:variable>
<xsl:variable name="result" select="jcs:invoke($query)"/>
<xsl:variable name="host" select="$result"/>
<output>
  <xsl:value-of select="concat('Name: ', $host)"/>
</output>
...
```

Enabling an Event Script

Event scripts are stored on a device's hard drive in the **/var/db/scripts/event** directory or on the flash drive in the **/config/scripts/event** directory. Only users in the Junos OS **super-user** login class can access and edit files in these directories. For information about setting the storage location for scripts, see Storing Scripts in Flash Memory.

NOTE: If the device has dual Routing Engines and you want to enable an event script to execute on both Routing Engines, you must copy the script to the /var/db/scripts/event or /config/scripts/event directory on both Routing Engines. The commit synchronize command does not automatically copy scripts between Routing Engines.

You must enable an event op script before it can be executed. Include the **file *filename*** statement at the **[edit event-options events-script]** hierarchy level, specifying the name of an Extensible Stylesheet Language Transformations (XSLT) or Stylesheet Language Alternative Syntax (SLAX) file containing an event script. Only users who belong to the Junos **super-user** login class can enable event scripts.

```
[edit event-options event-script]
file filename;
```

The filename of an event script written in SLAX must include the **.slax** extension for the script to be enabled and executed. No particular filename extension is required for event scripts written in XSLT, but we strongly recommend that you append the **.xsl** extension.

To determine which event scripts are currently enabled on the device, use the **show** command to display the files included at the **[edit event-options event-script]** hierarchy level. To ensure that the enabled files are on the device, list the contents of the **/var/run/scripts/event/** directory using the **file list /var/run/scripts/event** operational mode command.

Executing an Event Script

When you issue the **commit** command, event scripts enabled at the **[edit event-options event-script]** hierarchy level are placed into system memory and enabled for execution. After the commit operation completes, an event script is executed in response to an event notification within an event policy. For more information, see "Executing Event Scripts in an Event Policy" on page 28.

Replacing an Event Script

You can update or replace an existing event script without changing the device's configuration or disrupting operations. Follow these steps:

1. Edit or write the new event script.
2. Copy the script to the **/var/db/scripts/event** directory on the hard drive or the **/config/scripts/event** directory on the flash drive; for information about setting the

storage location for scripts, see Storing Scripts in Flash Memory. Only users who belong to the Junos **super-user** login class can alter files in these directories.

NOTE: If the device has dual Routing Engines, remember to copy the script to the **/var/db/scripts/event** or **/config/scripts/event** directory on both Routing Engines. The **commit synchronize** command does not automatically copy scripts between Routing Engines.

3. Issue the **request system scripts event-scripts reload** operational mode command.

```
user@host> request system scripts event-scripts reload
```

All event scripts are reloaded into the eventd process' memory.

Configuring Checksum Hashes for an Event Script

You can configure one or more checksum hashes that can be used to verify the integrity of an event script before the script runs on the switch, router, or security device.

To configure a checksum hash:

1. Create the script.
2. Place the script in the **/var/db/scripts/even** directory on the device.
3. Run the script through one or more hash functions to calculate hash values.

   Junos OS supports MD5, SHA-1, and SHA-256 hash functions.

```
user@host> file checksum md5 /var/db/scripts/commit/script1.slax
  MD5 (/var/db/scripts/event/script1.slax) = 3af7884eb56e2d4489c2e49b26a39a97
user@host> file checksum sha1 /var/db/scripts/commit/script1.slax
  SHA1 (/var/db/scripts/event/script1.slax) =
  00dc690fb08fb049577d012486c9a6dad34212c0
user@host> file checksum sha-256 /var/db/scripts/commit/script1.slax
  SHA256 (/var/db/scripts/event/script1.slax) =
  150bf53383769f3bfedd41fe73320777f208d4fda81230cb27b8738
```

4. Configure the script.

```
[edit event-options event-script]
user@host# set file script1.slax checksum
  md5 3af7884eb56e2d4489c2e49b26a39a97
[edit event-options event-script]
user@host# set file script1.slax checksum
  sha-1 00dc690fb08fb049577d012486c9a6dad34212c0
[edit event-options event-script]
user@host# set file script1.slax checksum
  sha-256 150bf53383769f3bfedd41fe73320777f208d4fda81230cb27b8738
```

   During the execution of the script, Junos OS recalculates the checksum value using the configured hash and verifies that the calculated value matches the configured value. If the values differ, the execution of the script fails. When you configure multiple checksum values with different hash algorithms, all the configured values must match the calculated values; otherwise, the script execution fails and the event policy fails.

Related Documentation

- Configuring Checksum Hashes for a Commit Script
- Configuring Checksum Hashes for an Op Script
- file checksum md5 command in the *System Basics and Services Command Reference*
- file checksum sha-256 command in the *System Basics and Services Command Reference*
- file checksum sha1 command in the *System Basics and Services Command Reference*

CHAPTER 8

Examples

Example: Assigning a Transfer Delay to an Event Policy Action

This section discusses three examples.

Example 1 Configure two event policies, **policy1** and **policy2**. The **policy1** event policy has a 5-second transfer-delay when uploading the **process.core** file to the **some-dest** destination. The **policy2** event policy has no transfer delay when uploading the **process.core** file to the same destination.

```
[edit event-options]
policy policy1 {
   events e1;
   then {
      upload filename process.core destination some-dest {
         transfer-delay 5;
      }
   }
}
policy policy2 {
```

```
    events e2;
    then {
      upload filename process.core destination some-dest;
    }
  }
  destinations {
    some-dest {
      archive-sites {
        "http://robot@my.little.com/foo/moo" password "password";
        "http://robot@my.big.com/foo/moo" password "password";
      }
    }
  }
```

Example 2 The **policy1** event policy has a 7-second (5 seconds + 2 seconds) transfer delay when uploading the **process.core** file to the destination. The **policy2** event policy has a 2-second transfer delay when uploading the **process.core** file to the destination.

```
[edit event-options]
policy policy1 {
  events e1;
  then {
    upload filename process.core destination some-dest {
      transfer-delay 5;
    }
  }
}
policy policy2 {
  events e2;
  then {
    upload filename process.core destination some-dest;
  }
}
destinations {
  some-dest {
    transfer-delay 2;
    archive-sites {
      "http://robot@my.little.com/foo/moo" password "password";
      "http://robot@my.big.com/foo/moo" password "password";
    }
  }
}
```

Example 3 The **policy1** event-policy is executed with **user1** privileges and uploads the **process.core** file after a transfer delay of 7 seconds (5 seconds + 2 seconds). The **policy2** event policy is executed with **root** privileges and uploads the **process.core** file after a transfer delay of 6 seconds (4 seconds + 2 seconds).

```
[edit event-options]
policy policy1 {
  events e1;
  then {
    upload filename process.core destination some-dest {
      transfer-delay 5;
      user-name user1;
    }
```



```
    }
  }
  policy policy2 {
    events e2;
    then {
      upload filename process.core destination some-dest {
        transfer-delay 4;
      }
    }
  }
  destinations {
    some-dest {
      transfer-delay 2;
      archive-sites {
        "http://robot@my.little.com/foo/moo" password "password";
        "http://robot@my.big.com/foo/moo" password "password";
      }
    }
  }
```

Example: Associating an Optional User with an Event Policy Action

Configure two event policies, **policy1** and **policy2**.

In **policy1**, associate user **user1** with the **execute-commands** action. The **execute-commands** action is executed with **user1** privileges.

In **policy2**, do not explicitly associate a user with the **event-script** action. The **event-script** action is executed with **root** privileges.

```
[edit system]
login {
  user user1 {
    class operator;
  }
}
[edit event-options]
policy p1 {
  events e1;
  then {
    execute-commands {
      commands {
        "show version";
      }
      user-name user1;
      output-filename command-output.txt;
      destination some-dest;
    }
  }
}
policy p2 {
  events e2;
  then {
    event-script script.xsl {
      output-filename event-script-output.txt;
```

```
        destination some-dest;
      }
    }
  }
```

Example: Controlling Event Policy Using a Regular Expression

The following policy is executed only if the **interface-name** attribute in both traps (**SNMP_TRAP_LINK_DOWN** and **SNMP_TRAP_LINK_UP**) match each other and the **interface-name** attribute in the **SNMP_TRAP_LINK_DOWN** trap starts with letter *t*. This means the policy is executed only for T1 (**t1-**) and T3 (**t3-**) interfaces. The policy is not executed when the **eventd** process receives traps from other interfaces.

NOTE: In system log files, the message tags appear in all uppercase letters. In the command-line interface (CLI), the message tags appear in all lowercase letters.

```
[edit event-options]
policy pol6 {
  events snmp_trap_link_down;
  within 120 events snmp_trap_link_up;
  attributes-match {
    snmp_trap_link_up.interface-name equals snmp_trap_link_down.interface-name;
    snmp_trap_link_down.interface-name matches "^t";
  }
  then {
    execute-commands {
      commands {
        "show interfaces {$$.interface-name}";
        "show configuration interfaces {$$.interface-name}";
      }
      output-filename config.txt;
      destination bsd2;
      output-format text;
    }
  }
}
```

Example: Correlating Events Based on Event Attributes

In the following policy, the two events are correlated only if two of their parameter values match. Matching on attributes of both events ensures that the two events are related. In this case, the interface addresses must match and the physical interface (ifd) names must match.

The **RPD_KRT_IFDCHANGE** error occurs when the routing protocol process (rpd) sends a request to the kernel to change the state of an interface and the request fails. The **RPD_RDISC_NOMULTI** error occurs when an interface is configured for router discovery but the interface does not support IP multicast operations as required.

In this example, **RPD_RDISC_NOMULTI.interface-name** might be **so-0/0/0.0**, and **RPD_KRT_IFDCHANGE.ifd-index** might be **so-0/0/0**.

```
[edit event-options]
policy 1 {
   events rpd_rdisc_nomulti;
   within 500 events rpd_krt_ifdchange;
   attributes-match {
      rpd_rdisc_nomulti.interface-address equals rpd_krt_ifdchange.address;
      rpd_rdisc_nomulti.interface-name starts-with rpd_krt_ifdchange.ifd-index;
   }
   then {
      ... actions ...
   }
}
```

Example: Correlating Events Based on Receipt of Other Events Within a Specified Time Interval

In the following policy, a set of commands is issued and the output is logged and saved to a given location. The policy is executed if ***event3***, ***event4***, or ***event5*** occurs within 60 seconds after ***event1*** or ***event2*** occurs. The pseudocode for the policy is as follows:

```
if this event is (event3 or event4 or event5)
  and
  (event1 or event2 has been received within the last 60 seconds)
then {
  run a set of commands;
  log the output of these commands to a location;
}
```

Specify two archive sites in the configuration. The device attempts to transfer to the first archive site in the list, moving to the next site only if the transfer fails.

```
[edit event-options]
policy 1 {
   events [ event3 event4 event5 ];
   within 60 events [ event1 event2 ];
   then {
      execute-commands {
         commands {
            "command";
         }
         output-filename my_cmd_out;
         destination policy-1-command-dest;
      }
   }
}
destinations {
   policy-1-command-dest {
      archive-sites {
         http://robot@my.big.com/a/b;
         http://robot@my.little.com/a/b;
      }
   }
}
```

Example: Generating an Internal Event

The following two examples generate an internal event. In the first example, the configuration generates an internal event every hour. In the second example, the configuration generates an event every night at midnight.

In the following example, the internal event called **EVERY-ONE-HOUR** is generated every hour (3600 seconds). If 3601 seconds pass and the event has not been generated, certain actions are taken.

```
[edit event-options]
generate-event every-one-hour time-interval 3600;
policy check-heartbeat {
   events every-one-hour;
   within 3601 not events every-one-hour;
   then {
      ... actions ...
   }
}
```

In the following example, the internal event called **IT-IS-MIDNIGHT** is generated at 12:00 AM every night (00:00:00). When the eventd process receives the **IT-IS-MIDNIGHT** event, certain actions are taken.

```
[edit event-options]
generate-event it-is-midnight time-of-day 00:00:00;
policy midnight-chores {
   events it-is-midnight;
   then {
      ... actions ...
   }
}
```

Example: Ignoring Events Based on Receipt of Other Events

In the following policy, if any of ***event1***, ***event2***, or ***event3*** has occurred, and either ***event4*** or ***event5*** has occurred within the last 600 seconds, and ***event6*** has not occurred within the last 800 seconds, then the event that triggered the policy (***event1***, ***event2***, or ***event3***) is ignored, meaning system log messages are not created.

```
[edit event-options]
policy 1 {
   events [ event1 event2 event3 ];
   within 600 events [ event4 event5 ];
   within 800 not events event6;
   then {
      ignore;
   }
}
```

Sometimes events are generated repeatedly within a short period of time. In this case, it is redundant to execute a policy multiple times, once for each instance of the event. Event dampening allows you to slow down the execution of policies by ignoring instances of an event that occur within a specified time after another instance of the same event.

In the following example, an action is taken only if the eventd process has not received another instance of the event within the past 60 seconds. If an instance of the event has been received within the last 5 seconds, the policy is not executed and a system log message for the event is not created again.

```
[edit event-options]
policy dampen-policy {
   events event1;
   within 60 events event1;
   then {
      ignore;
   }
}
policy policy {
   events event1;
   then {
      ... actions ...
   }
}
```

Example: Limiting Event Script Output Based on a Specific Event Type

In situations where an event policy is triggered by multiple event types, you can limit the number of events that trigger the event script. For example, the following event policy triggers the **event-details.slax** event script whenever a **ui_login_event** or **ui_logout_event** occurs.

```
event-options {
   policy event-detail {
      events [ ui_login_event ui_logout_event ];
      then {
         event-script event-details.slax {
            output-filename systemlog;
            destination /tmp;
         }
      }
   }
}
```

The **event-details.slax** event script writes a log file only when the **ui_login_event** event occurs.

```
version 1.0;
ns junos = "http://xml.juniper.net/junos/*/junos";
ns xnm = "http://xml.juniper.net/xnm/1.1/xnm";
ns jcs = "http://xml.juniper.net/junos/commit-scripts/1.0";
ns ext = "http://xmlsoft.org/XSLT/namespace";

var $event-definition = {
   <event-options> {
      <policy> {
```

```
        <namex> "event-detail";
        <eventsx> "ui_login_event";
        <thenx> {
          <event-scriptx> {
            <namex> "event_detail.slax";
            <output-filenamex> "foo";
            <destinationx> {
              <namex> "foo";
            }
          }
        }
      }
    }

    match / {
      <event-script-resultsx> {
        <event-triggered-this-policyx> {
          expr event-script-input/trigger-event/id;
        }
        <type-of-eventx> {
          expr event-script-input/trigger-event/type;
        }
        <process-namex> {
          expr event-script-input/trigger-event/attribute-list/attribute/name;
        }
      }
    }
```

Example: Raising an SNMP Trap in Response to an Event

Raise a trap and execute an associated event script in response to an event:

```
[edit event-options]
policy p1 {
  events ui_mgd_terminate;
  then {
    raise-trap;
    event-script bgp.xsl {
      arguments {
        destination {$ui_mgd_terminate.destination};
        code 2;
      }
      output-filename bgp-out;
      destination bsd3;
    }
  }
}
```

Example: Representing the Correlating Event in an Event Policy

```
[edit event-options]
policy p1 {
  events [ e1 e2 e3 ];
  within 60 events [ e4 e5 e6 ];
```

```
    then {
      execute-commands {
        commands {
          "show interfaces {$$.interface-name}";
          "show interfaces {$e4.interface-name}";
          "show interfaces {$*.interface-name}";
        }
        output-filename command-output.txt;
        destination some-dest;
      }
    }
  }
```

In the **show interfaces {$$.interface-name}** command, the value of the **interface-name** attribute of event **e1**, **e2**, or **e3** is substituted for the **{$$.interface-name}** variable.

In the **show interfaces {$e4.interface-name}** command, the value of the **interface-name** attribute of the most recent **e4** event is substituted for the **{$e4.interface-name}** variable.

In the **show interfaces {$*.interface-name}** command, the value of the **interface-name** attribute of the most recent **e4**, **e5**, or **e6** event is substituted for the **{$*.interface-name}** variable. If one of **e4**, **e5**, or **e6** occurs within 60 seconds of **e1**, **e2**, or **e3**, the value of the **interface-name** attribute for that correlating event (**e4**, **e5**, or **e6**) is substituted for the **{$*.interface-name}** variable. If the correlating event does not have an **interface-name** attribute, the software does not execute the **show interfaces {$*.interface-name}** command.

If both **e4** and **e5** occur within 60 seconds of **e1**, then the value of the **interface-name** attribute for **e4** is substituted for the **{$*.interface-name}** variable. This is because the event process (eventd) searches for correlating events in sequential order as configured in the **within** statement. In this case, the order is **e4** > **e5** > **e6**.

Example: Retrying the File Upload Action

This section discusses two examples.

Example 1 Configure a policy that retries the file upload operation two times with a time interval of 5 seconds between retries:

```
event-options {
  policy p1 {
    events e1;
    then {
      execute-commands {
        commands {
          command1;
        }
        output-filename command-output.txt;
        destination some-dest {
          retry-count 2 retry-interval 5;
        }
      }
    }
  }
}
```

Example 2 Configure a transfer delay of 10 seconds and retry the file upload operation two times with a time interval of 5 seconds between retries:

```
event-options {
  policy p2 {
    events e1;
    then {
      execute-commands {
        commands {
          command1;
        }
        output-filename command-output.txt;
        destination some-dest {
          retry-count 2 retry-interval 5;
          transfer-delay 10;
        }
      }
    }
  }
}
```

The transfer delay is in operation for the first upload attempt only. The policy uploads the **command-output.txt** file after a 10-second transfer delay. If the event process (eventd) detects failure of the upload operation, eventd retries the upload operation after 5 seconds. The failure detection time can be in the range from 60 to 90 seconds, depending on the transmission protocol, such as FTP.

The following sequence describes the file upload operation with two failed retransmissions:

1. Policy triggers upload operation.
2. Transmission delay of 10 seconds.
3. Policy tries to upload the output file.
4. Policy detects transmission failure.
5. Retry interval of 5 seconds.
6. Policy tries to upload the output file.
7. Policy detects transmission failure.
8. Retry interval of 5 seconds.
9. Policy tries to upload the output file.
10. Policy detects transmission failure.
11. Policy declares the failure of the file upload operation.

Example: Triggering a Policy Based on Event Count

This section discusses two examples.

NOTE: The **RADIUS_LOGIN_FAIL**, **TELNET_LOGIN_FAIL**, and **SSH_LOGIN_FAIL** events are not actual Junos OS events. They are illustrative for these examples.

Example 1 Configure an event policy called **login**. The **login** policy is executed if five login failure events (**RADIUS_LOGIN_FAIL**, **TELNET_LOGIN_FAIL**, or **SSH_LOGIN_FAIL**) are generated within 120 seconds. Take action by executing the **login-fail.xsl** event script, which disables the user account.

```
[edit event-options]
policy login {
   events [ RADIUS_LOGIN_FAIL TELNET_LOGIN_FAIL SSH_LOGIN_FAIL ];
   within 120 {
      trigger after 4;
   }
   then {
      event-script login-fail.xsl {
         destination some-dest;
      }
   }
}
```

Table 3 on page 61 shows how events add to the count.

Table 3: Event Count Triggers Policy

Event Number	Event	Time	Count	Order
1	**RADIUS_LOGIN_FAIL**	00:00:00	1	[1]
2	**TELNET_LOGIN_FAIL**	00:00:20	2	[1 2]
3	**RADIUS_LOGIN_FAIL**	00:02:05	2	[2 3]
4	**SSH_LOGIN_FAIL**	00:02:40	2	[3 4]
5	**TELNET_LOGIN_FAIL**	00:02:55	3	[3 4 5]
6	**TELNET_LOGIN_FAIL**	00:03:01	4	[3 4 5 6]
7	**RADIUS_LOGIN_FAIL**	00:03:55	5	[3 4 5 6 7]

The columns in Table 3 on page 61 mean the following:

- Event number—Event sequence number.
- Event—Policy login events received by the event process (eventd).
- Time—Time (in ***hh*:*mm*:*ss*** format) when eventd receives the event.
- Count—The number of events received by eventd within the last 120 seconds.
- Order—Order of events as received by eventd within the last 120 seconds.

At time 00:03:55, the value of count is more than 4; therefore, the **login** policy executes the **login-fail.xsl** script.

Example 2 Configure an event policy called **login**. The **login** policy is executed if five login failure events (**RADIUS_LOGIN_FAIL**, **TELNET_LOGIN_FAIL**, or **SSH_LOGIN_FAIL**) are generated within 120 seconds from username **roger**. Take action by executing the **login-fail.xsl** event script, which disables the **roger** user account.

```
[edit event-options]
policy p2 {
   events [ RADIUS_LOGIN_FAIL TELNET_LOGIN_FAIL SSH_LOGIN_FAIL ];
   within 120 {
      trigger after 4;
   }
   attributes-match {
      RADIUS_LOGIN_FAIL.username matches roger;
      TELNET_LOGIN_FAIL.username matches roger;
   }
   then {
      event-script login-fail.xsl {
         destination some-dest;
      }
   }
}
```

Example: Using Nonstandard System Log Messages to Trigger an Event Policy

Reference a **KERNEL** system log message in an event policy. The **raise-trap** action in the **then** statement is executed only if a **KERNEL** event containing a message that matches "exited on signal 11" occurs.

```
[edit event-options]
policy kernel-policy {
   events KERNEL;
   attributes-match {
      KERNEL.message matches "exited on signal 11";
   }
   then {
      raise-trap;
   }
}
```

CHAPTER 9

Summary of Event Policy Configuration Statements

archive-sites

Syntax

```
archive-sites {
    url <password password>;
}
```

Hierarchy Level [edit event-options destinations *destination-name*]

Release Information Statement introduced in Junos OS Release 7.5.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Specify an archive site to which files are transferred. If you specify more than one archive site, the device attempts to transfer to the first archive site in the list, moving to the next site only if the transfer fails.

Options

url—The archive destination specified as Hypertext Transfer Protocol (HTTP) URL, FTP URL, or secure copy (scp)-style remote file specification. URLs of the type **file://** are not supported; however, local device directories are supported (for example, **/var/tmp/**).

password *password*—A plain-text password for login into the archive site.

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation

- Defining Destinations for File Archiving by Event Policies on page 19
- destinations on page 68

arguments

Syntax

```
arguments {
  argument-name argument-value;
}
```

Hierarchy Level [edit event-options policy *policy-name* then event-script *filename*]

Release Information Statement introduced in Junos OS Release 7.6.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Define command-line arguments for an event script that is invoked from an event policy.

Options ***argument-name***—Name of the argument.

argument-value—Value of the argument.

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation

- Executing Event Scripts in an Event Policy on page 28
- event-script on page 72
- policy on page 79

attributes-match

Syntax

```
attributes-match {
  event1.attribute-name equals event2.attribute-name;
  event.attribute-name matches regular-expression;
  event1.attribute-name starts-with event2.attribute-name;
}
```

Hierarchy Level [edit event-options policy *policy-name*]

Release Information Statement introduced in Junos OS Release 7.5.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Execute the policy only if the attributes of two events are correlated or if the attribute of one event matches a regular expression.

If the **attributes-match** statement includes the **equals** or **starts-with** options, or if it includes a **matches** option that includes a clause for an event that is not specified at the **[edit event-options policy *policy-name* events]** hierarchy level, you must include one or more **within** statements in the same policy configuration.

Starting with Junos OS Release 11.1, you can include event policy variables within the statement to differentiate between a trigger event attribute and a correlated event attribute. You can use variables of the following forms:

- **{$$.*attribute-name*}**—The double dollar sign ($$) notation represents the event that is triggering a policy. When combined with an attribute name, the variable is replaced by the value of the attribute of the triggering event.
- **{$*event.attribute-name*}**—The dollar sign with the event name ($*event*) notation represents the most recent event that matches the specified event. The variable is replaced by the value of the attribute of the most recent event that matches ***event***.

The statements are explained separately.

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation

- Using Correlated Events to Trigger an Event Policy on page 11
- Using Regular Expressions to Refine the Set of Events That Trigger a Policy on page 15
- equals on page 69
- matches on page 76
- starts-with on page 81
- within on page 88

commands

Syntax

```
commands {
  "command";
}
```

Hierarchy Level [edit event-options policy *policy-name* then execute-commands]

Release Information Statement introduced in Junos OS Release 7.5.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Specify an operational mode command to be issued on receipt of an event.

Options ***command***—Command to be issued. Enclose each command in quotation marks (“ ”). The event process (eventd) issues the commands in the order in which they appear in the configuration.

You can include variables in commands. The eventd process replaces each variable with values contained in the event that triggers the policy. You can use command variables of the following forms:

- **{$$.*attribute-name*}**—The double dollar sign (**$$**) notation represents the event that is triggering a policy. When combined with an attribute name, the command variable is replaced by the value of the attribute name of the triggering event.
- **{$*event*.*attribute-name*}**—The dollar sign with the event name (**$*event***) notation represents the most recent event that matches the specified event. The variable is replaced by the value of the attribute name of the most recent event that matches ***event***.
- **{$*.*attribute-name*}**—The dollar sign with the asterisk (**$***) notation represents the most recent event that matches any of the correlating events. The variable is replaced by the value of the attribute name of the most recent event that matches any of the events specified in the policy configuration.

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation

- Configuring an Event Policy to Execute Operational Mode Commands on page 25
- Representing the Correlating Event in an Event Policy on page 14

destination

Syntax

```
destination destination-name {
  retry-count count retry-interval seconds;
  transfer-delay seconds;
}
```

Hierarchy Level

[edit event-options policy *policy-name* then event-script *filename*],
[edit event-options policy *policy-name* then execute-commands]

Release Information

Statement introduced in Junos OS Release 7.5.
Support extended to the **[edit event-options policy *policy-name* then event-script *filename*]** hierarchy level in Junos OS Release 7.6.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description

Assign a location to which to upload command or script output for the specified policy.

Options

destination-name—Name of a destination defined in the **destinations** statement at the **[edit event-options]** hierarchy level.

The remaining statements are explained separately.

Required Privilege Level

maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation

- Configuring an Event Policy to Execute Operational Mode Commands on page 25
- Executing Event Scripts in an Event Policy on page 28
- **destinations on page 68**

destinations

Syntax

```
destinations {
  destination-name {
    archive-sites {
      url <password password>;
    }
    transfer-delay seconds;
  }
}
```

Hierarchy Level [edit event-options]

Release Information Statement introduced in Junos OS Release 7.5.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Define one or more destinations, each with a unique name and other attributes. You can use the destination as a storage location for command output and for various files, such as system log files and core files.

Options ***destination-name***—Name of a destination.

The remaining statements are explained separately.

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation

- Defining Destinations for File Archiving by Event Policies on page 19

equals

Syntax *event1.attribute-name* equals *event2.attribute-name*;

Hierarchy Level [edit event-options policy *policy-name* attributes-match]

Release Information Statement introduced in Junos OS Release 7.5.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Execute the policy only if the specified attribute of ***event1*** equals the specified attribute of ***event2***.

Options ***event1.attribute-name***—Attribute of one event.

event2.attribute-name—Attribute of another event.

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation

- Using Correlated Events to Trigger an Event Policy on page 11

event-options

Syntax

```
event-options {
  destinations {
    destination-name {
      archive-sites {
        url <password password>;
      }
      transfer-delay seconds;
    }
  }
  event-script {
    file filename {
      checksum (md5 | sha-256 | sha1) hash;
      refresh;
      refresh-from url;
      remote-execution {
        remote-hostname {
          passphrase user-password;
          username user-login;
        }
      }
      source url;
    }
    refresh;
    refresh-from url;
    traceoptions {
      file <filename> <files number> <size size> <world-readable | no-world-readable>;
      flag flag;
      no-remote-trace;
    }
  }
  generate-event event-name {
    time-interval seconds;
    time-of-day hh:mm:ss;
  }
  policy policy-name {
    attributes-match {
      event1.attribute-name equals event2.attribute-name;
      event.attribute-name matches regular-expression;
      event1.attribute-name starts-with event2.attribute-name;
    }
    events [ events ];
    then {
      event-script filename {
        arguments {
          argument-name argument-value;
        }
        destination destination-name {
          retry-count count retry-interval seconds;
          transfer-delay seconds;
        }
        output-filename filename;
        output-format (text | xml);
```

```
        user-name name;
      }
      execute-commands {
        commands {
          "command";
        }
        destination destination-name {
          retry-count count retry-interval seconds;
          transfer-delay seconds;
        }
        output-filename filename;
        output-format (text | xml);
        user-name username;
      }
      ignore;
      raise-trap;
      upload filename (filename | committed) destination destination-name {
        retry-count count retry-interval seconds;
        transfer-delay seconds;
        user-name username;
      }
    }
    within seconds {
      events [ events ];
      not events [ events ];
      trigger (after number | on number | until number);
    }
  }
  traceoptions {
    file filename <files number> <size size> <world-readable | no-world-readable>;
    flag flag;
  }
}
```

Hierarchy Level [edit]

Release Information Statement introduced in Junos OS Release 7.5.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Configure event policies.

The statements are explained separately.

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

event-script

Syntax

```
event-script filename {
  arguments {
    argument-name argument-value;
  }
  destination destination-name {
    retry-count count retry-interval seconds;
    transfer-delay seconds;
  }
  output-filename filename;
  output-format (text | xml);
  user-name username;
}
```

Hierarchy Level [edit event-options policy *policy-name* then]

Release Information Statement introduced in Junos OS Release 7.6.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description On receipt of an event, specify operational mode commands to be issued, the format of the command output, and a name and destination for the output file.

The statements are explained separately.

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation

- Executing Event Scripts in an Event Policy on page 28

events (Associating Events with a Policy)

Syntax events [*events*];

Hierarchy Level [edit event-options policy *policy-name*]

Release Information Statement introduced in Junos OS Release 7.5.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Create a list of events that trigger this policy. If one or more of the listed events occurs, the policy is executed.

Options **[*events*]**—List of events. Events can be internally generated, or they can be generated by Junos OS processes.

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation
- Using Correlated Events to Trigger an Event Policy on page 11
- Example: Correlating Events Based on Event Attributes on page 54

events (Correlating Events with Each Other)

Syntax events [*events*];

Hierarchy Level [edit event-options policy *policy-name* within *seconds*]

Release Information Statement introduced in Junos OS Release 7.5.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Create a list of events that must occur within a specified time interval for the policy to be triggered.

Options **[*events*]**—List of events. Events can be internally generated, or they can be generated by Junos OS processes.

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation
- Using Correlated Events to Trigger an Event Policy on page 11

execute-commands

Syntax

```
execute-commands {
  commands {
    "command";
  }
  destination destination-name {
    retry-count count retry-interval seconds;
    transfer-delay seconds;
  }
  output-filename filename;
  output-format (text | xml);
  user-name username;
}
```

Hierarchy Level [edit event-options policy *policy-name* then]

Release Information Statement introduced in Junos OS Release 7.5.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description On receipt of an event, specify operational mode commands to be issued, the format of the command output, and a name and destination for the output file.

The statements are explained separately.

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation

- Configuring an Event Policy to Execute Operational Mode Commands on page 25

generate-event

Syntax

```
generate-event event-name {
  time-interval seconds;
  time-of-day hh:mm:ss;
}
```

Hierarchy Level [edit event-options]

Release Information Statement introduced in Junos OS Release 7.5.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Generate an internal event, based on a time interval or the time of day. You can configure up to 10 internal events.

Options ***event-name***—Name of an internally generated event.

The statements are explained separately.

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation

- Generating Internal Events to Trigger Event Policies on page 16
- time-interval on page 83
- time-of-day on page 83

ignore

Syntax

```
ignore;
```

Hierarchy Level [edit event-options policy *policy-name* then]

Release Information Statement introduced in Junos OS Release 7.5.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Define a policy that ignores particular events. If one or more of the listed events occur, a system log message for the event is not generated, and no further policies associated with this event are processed. If you include the **ignore** statement in a policy configuration, you cannot configure any other actions in the policy.

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation

- Configuring Event Policies to Ignore an Event on page 33

matches

Syntax *event.attribute-name* matches *regular-expression*;

Hierarchy Level [edit event-options policy *policy-name* attributes-match]

Release Information Statement introduced in Junos OS Release 7.5.

Description Execute the policy only if the specified attribute of ***event*** matches a regular expression.

Options ***event.attribute-name***—Event attribute to compare to a regular expression.

regular-expression—Regular expression to compare.

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation
- Using Correlated Events to Trigger an Event Policy on page 11
- Using Regular Expressions to Refine the Set of Events That Trigger a Policy on page 15

not

Syntax not events [*events*];

Hierarchy Level [edit event-options policy *policy-name* within *seconds*]

Release Information Statement introduced in Junos OS Release 7.5.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Create a list of events that must not occur within the specified time interval for the policy to be triggered.

Options **[*events*]**—List of events. Events can be internally generated, or they can be generated by Junos OS processes.

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation
- Using Correlated Events to Trigger an Event Policy on page 11

output-filename

Syntax output-filename *filename*;

Hierarchy Level [edit event-options policy *policy-name* then event-script *filename*],
[edit event-options policy *policy-name* then execute-commands]

Release Information Statement introduced in Junos OS Release 7.5.
Support at the **[edit event-options policy *policy-name* then event-script *filename*]** hierarchy level introduced in Junos OS Release 7.6.

Description Assign a filename to which to write command or script output for the specified commands or script. For op scripts, this statement is optional.

Options ***filename***—Name of a file in which to write command or script output.

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation
- Configuring an Event Policy to Execute Operational Mode Commands on page 25
- Executing Event Scripts in an Event Policy on page 28

output-format

Syntax output-format (text | xml);

Hierarchy Level [edit event-options policy *policy-name* then event-script *filename*],
[edit event-options policy *policy-name* then execute-commands]

Release Information Statement introduced in Junos OS Release 7.5.
Support at the **[edit event-options policy *policy-name* then event-script *filename*]** hierarchy level introduced in Junos OS Release 8.3.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Specify the format (ASCII text or XML) for the output of the specified commands or script.

Options **text**—Formatted ASCII text.

xml—Junos Extensible Markup Language (XML) tags.

Default: **xml** at the **[edit event-options policy *policy-name* then execute-commands]** hierarchy level and **text** at the **[edit event-options policy *policy-name* then event-script *filename*]** hierarchy level.

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation

- Configuring an Event Policy to Execute Operational Mode Commands on page 25
- Executing Event Scripts in an Event Policy on page 28

policy

Syntax

```
policy policy-name {
  attributes-match {
    event1.attribute-name equals event2.attribute-name;
    event.attribute-name matches regular-expression;
    event1.attribute-name starts-with event2.attribute-name;
  }
  events [ events ];
  then {
    ... the then subhierarchy appears at the end of the [edit event-options policy policy-name]
      hierarchy level ...
  }
  within seconds {
    events [ events ];
    not events [ events ];
    trigger (on | after | until) event-count;
  }

  then {
    event-script filename {
      arguments {
        argument-name argument-value;
      }
      destination destination-name {
        retry-count count retry-interval seconds;
        transfer-delay seconds;
      }
      output-filename filename;
      output-format (text | xml);
      user-name username;
    }
    execute-commands {
      commands {
        "command";
      }
      destination destination-name {
        retry-count count retry-interval seconds;
        transfer-delay seconds;
      }
      output-filename filename;
      output-format (text | xml);
      user-name username;
    }
    ignore;
    raise-trap;
    upload filename (filename | committed) destination destination-name {
      retry-count count retry-interval seconds;
      transfer-delay seconds;
      user-name username;
    }
  }
}
```

Hierarchy Level [edit event-options]

Release Information Statement introduced in Junos OS Release 7.5.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Define an event policy to be processed by the eventd process. If you configure a policy, the **events** and **then** statements are mandatory.

You can configure multiple policies to be processed for an event. The policies are executed in the order in which they appear in the configuration. If you configure more than one policy for an event, and if one of the policies is to ignore the event, no policies that follow the **ignore** statement are executed.

Default If you do not configure a policy for an event, the event is recorded in the system log.

Options ***policy-name***—Name of an event policy.

The statements are explained separately.

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

raise-trap

Syntax raise-trap;

Hierarchy Level [edit event-options policy *policy-name* then]

Release Information Statement introduced in Junos OS Release 8.1.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Define a policy that raises an SNMP trap in response to an event. If one or more of the listed events occur, the system log message for the event is converted into a trap. This enables an agent to notify a trap-based network management system (NMS) of significant events.

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation

- Configuring Event Policies to Raise SNMP Traps on page 34

retry-count

Syntax retry-count *number* retry-interval *seconds*;

Hierarchy Level [edit event-options policy *policy-name* then event-script *filename* destination *destination-name*],
[edit event-options policy *policy-name* then execute-commands destination *destination-name*],
[edit event-options policy *policy-name* then upload filename (*filename* | committed) destination *destination-name*]

Release Information Statement introduced in Junos OS Release 8.4.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Configure an event policy to retry a file upload operation if the first attempt fails.

Default If you do not include this statement, the file upload operation is attempted one time only.

Options ***number***—Number of retries.

retry-interval *seconds*—Length of time to wait between retries.

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation
- Configuring an Event Policy to Retry the File Upload Action on page 23

starts-with

Syntax *event1.attribute-name* starts-with *event2.attribute-name*;

Hierarchy Level [edit event-options policy *policy-name* attributes-match *event1.attribute-name*]

Release Information Statement introduced in Junos OS Release 7.5.

Description Execute the policy only if the specified attribute of ***event1*** starts with the specified attribute of ***event2***.

Options ***event1.attribute-name***—Attribute of one event.

event2.attribute-name—Attribute of another event.

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation
- Using Correlated Events to Trigger an Event Policy on page 11

then

Syntax

```
then {
  event-script filename {
    arguments {
      argument-name argument-value;
    }
    destination destination-name {
      retry-count count retry-interval seconds;
      transfer-delay seconds;
    }
    output-filename filename;
    output-format (text | xml);
    user-name username;
  }
  execute-commands {
    commands {
      "command";
    }
    destination destination-name {
      retry-count count retry-interval seconds;
      transfer-delay seconds;
    }
    output-filename filename;
    output-format (text | xml);
    user-name username;
  }
  ignore;
  raise-trap;
  upload filename (filename | committed) destination destination-name {
    retry-count count retry-interval seconds;
    transfer-delay seconds;
    user-name username;
  }
}
```

Hierarchy Level

[edit event-options policy *policy-name*]

Release Information

Statement introduced in Junos OS Release 7.5.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description

Define actions to take if an event occurs. For each policy, you can configure multiple actions.

The statements are explained separately.

Required Privilege Level

maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation

- Configuring an Event Policy to Upload Files on page 20
- Configuring an Event Policy to Execute Operational Mode Commands on page 25
- Executing Event Scripts in an Event Policy on page 28

- Configuring Event Policies to Ignore an Event on page 33
- Configuring Event Policies to Raise SNMP Traps on page 34

time-interval

Syntax time-interval *seconds*;

Hierarchy Level [edit event-options generate-event *event-name*]

Release Information Statement introduced in Junos OS Release 7.5.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Configure a frequency at which to generate a particular event.

Options ***seconds***—Time interval between internally generated events.
Range: 60 through 2,592,000 seconds

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation
- Generating Internal Events to Trigger Event Policies on page 16
- generate-event on page 75
- time-of-day on page 83

time-of-day

Syntax time-of-day *hh*:*mm*:*ss*;

Hierarchy Level [edit event-options generate-event *event-name*]

Release Information Statement introduced in Junos OS Release 7.5.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Configure a time of day at which to generate a particular event.

Options ***hh*:*mm*:*ss***—Time of day at which to generate an event.

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation
- Generating Internal Events to Trigger Event Policies on page 16
- generate-event on page 75
- time-interval on page 83

traceoptions

Syntax

```
traceoptions {
  file <filename> <files number> <match regular-expression> <size size> <world-readable |
    no-world-readable>;
  flag flag;
  no-remote-trace;
}
```

Hierarchy Level [edit event-options]

Release Information Statement introduced in Junos OS Release 7.5.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Define tracing operations for event policies.

Default If you do not include this statement, no event-policy-specific tracing operations are performed.

Options ***filename***—Name of the file to receive the output of the tracing operation. All files are placed in the directory **/var/log**. By default, commit script process tracing output is placed in the file **eventd**. If you include the **file** statement, you must specify a filename. To retain the default, you can specify **eventd** as the filename.

files ***number***—(Optional) Maximum number of trace files. When a trace file named ***trace-file*** reaches its maximum size, it is renamed and compressed to ***trace-file*.0.gz**. When ***trace-file*** again reaches its maximum size, ***trace-file*.0.gz** is renamed ***trace-file*.1.gz** and ***trace-file*** is renamed and compressed to ***trace-file*.0.gz**. This renaming scheme continues until the maximum number of trace files is reached. Then the oldest trace file is overwritten.

If you specify a maximum number of files, you also must specify a maximum file size with the **size** option and a filename.

Range: 2 through 1000

Default: 3 files

flag—Tracing operation to perform. To specify more than one tracing operation, include multiple **flag** statements. You can include the following flags:

- **all**—Log all operations
- **configuration**—Log reading of configuration at the **[edit event-options]** hierarchy level
- **events**—Log eventd processing
- **database**—Log events involving storage and retrieval in events database
- **server**—Log communication with processes that are generating events
- **timer-events**—Log internally generated events

match *regular-expression*—(Optional) Refine the output to include lines that contain the regular expression.

no-world-readable—Restrict file access to owner. This is the default.

size *size*—(Optional) Maximum size of each trace file, in kilobytes (KB), megabytes (MB), or gigabytes (GB). When a trace file named ***trace-file*** reaches this size, it is renamed and compressed to ***trace-file*.0.gz**. When the ***trace-file*** again reaches its maximum size, ***trace-file*.0.gz** is renamed ***trace-file*.1.gz** and ***trace-file*** is renamed and compressed to ***trace-file*.0.gz**. This renaming scheme continues until the maximum number of trace files is reached. Then the oldest trace file is overwritten.

If you specify a maximum file size, you also must specify a maximum number of trace files with the **files** option and filename.

Syntax: ***x*k** to specify KB, ***x*m** to specify MB, or ***x*g** to specify GB

Range: 10 KB through 1 GB

Default: 128 KB

world-readable—(Optional) Enable unrestricted file access.

Required Privilege Level

maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation

- Tracing Event Policy Processing on page 107

transfer-delay

Syntax transfer-delay *seconds*;

Hierarchy Level
[edit event-options destinations *destination-name*],
[edit event-options policy *policy-name* then event-script *filename* destination *destination-name*],
[edit event-options policy *policy-name* then execute-commands destination *destination-name*],
[edit event-options policy *policy-name* then upload filename (*filename* | committed) destination *destination-name*]

Release Information
Statement introduced in Junos OS Release 7.5.
Support at the **[edit event-options policy *policy-name* then ...]** hierarchy levels introduced in Junos OS Release 8.4.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Configure a delay before transferring files. This allows the files to be completely generated before the upload starts. If you configure a transfer delay at the **[edit event-options destination *destination-name*]** hierarchy level and at one of the **[edit event-options policy *policy-name* then ...]** hierarchy levels, the resulting delay is the sum of the two delays.

Default If you do not include this statement, there is no transfer delay.

Options ***seconds***—Duration of the delay before files are uploaded.

Required Privilege Level
maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation
- Defining Destinations for File Archiving by Event Policies on page 19
- Configuring the Delay Before Files Are Uploaded by an Event Policy on page 21

trigger

Syntax trigger (on | after | until) *event-count*;

Hierarchy Level [edit event-options policy *policy-name* within *seconds*]

Release Information Statement introduced in Junos OS Release 8.4.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Configure an event policy to be triggered if an event or set of events occurs ***event-count*** times within a specified time period.

Default If you do not include this statement, the policy is executed on receipt of the first configured event.

Options

after *event-count*—The policy is executed when the number of matching events received equals ***event-count*** + 1.

on *event-count*—The policy is executed when the number of matching events received equals ***event-count***.

until *event-count*—The policy is executed each time a matching event is received and stops being executed when the number of matching events received equals ***event-count***.

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation

- Triggering an Event Policy Based on Event Count on page 15

within

Syntax

```
within seconds {
  events [ events ];
  not events [ events ];
  trigger (after | on | until) event-count;
}
```

Hierarchy Level [edit event-options policy *policy-name*]

Release Information Statement introduced in Junos OS Release 7.5.

Description Create a list of events that must (or must not) occur within a specified time interval for the policy to be triggered.

The statements are explained separately.

Options *seconds*—Interval between events.

Range: 60 through 604,800 seconds

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation

- Using Correlated Events to Trigger an Event Policy on page 11

upload

Syntax

```
upload filename  (filename | committed) destination destination-name {
    retry-count count retry-interval seconds;
    transfer-delay seconds;
    user-name username;
}
```

Hierarchy Level [edit event-options policy *policy-name* then]

Release Information Statement introduced in Junos OS Release 7.5.
committed option to **filename** statement introduced in Junos OS Release 8.1.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description On receipt of an event, upload the committed configuration file to a destination.

Options

destination *destination-name*—Name of the destination for the uploaded file. It must be defined in the **destinations** statement at the **[edit event-options]** hierarchy level.

filename (*filename* | committed)—Name of the file to upload. Specify either the word **committed** to upload the most recently committed configuration file, or the filename of another file.

The remaining statements are explained separately.

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation

- destinations on page 68
- Configuring an Event Policy to Upload Files on page 20

user-name

Syntax user-name *username*;

Hierarchy Level [edit event-options policy *policy-name* then event-script *filename*],
[edit event-options policy *policy-name* then execute-commands],
[edit event-options policy *policy-name* then upload filename (*filename* | committed) destination *destination-name*]

Release Information Statement introduced in Junos OS Release 8.4.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Associate a user with an action in an event policy. The event policy action is executed under the privileges of the associated user.

Default If you do not associate a user with an action, the action is executed as user **root**.

Options ***username***—A username that is configured at the **[edit system login]** hierarchy level.

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation

- Changing the User Privilege Level for an Event Policy Action on page 34

CHAPTER 10

Summary of Event Script Configuration Statements

checksum

Syntax checksum (md5 | sha-256 | sha1) *hash*;

Hierarchy Level [edit event-options event-script file *filename*],
[edit system scripts commit file *filename*],
[edit system scripts op file *filename*]

Release Information Statement introduced in Junos OS Release 9.5.

Description For Junos OS commit scripts and op scripts, specify the MD5, SHA-1, or SHA-256 checksum hash. When it executes a local event, commit, or op script, Junos OS verifies the authenticity of the script by using the configured checksum hash.

Options

md5 ***hash***—MD5 checksum of this script.

sha-256 ***hash***—SHA-256 checksum of this script.

sha1 ***hash***—SHA-1 checksum of this script.

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation

- Configuring Checksum Hashes for a Commit Script
- Configuring Checksum Hashes for an Event Script on page 48
- Configuring Checksum Hashes for an Op Script
- Executing an Op Script from a Remote Site
- file checksum md5 command in the *System Basics and Services Command Reference*
- file checksum sha-256 command in the *System Basics and Services Command Reference*
- file checksum sha1 command in the *System Basics and Services Command Reference*

event-script

Syntax

```
event-script {
  file filename {
    checksum (md5 | sha-256 | sha1) hash;
    refresh;
    refresh-from url;
    remote-execution {
      remote-hostname {
        passphrase user-password;
        username user-login;
      }
    }
    source url;
  }
  refresh;
  refresh-from url;
  traceoptions {
    file <filename> <files number> <size size> <world-readable | no-world-readable>;
    flag flag;
    no-remote-trace;
  }
}
```

Hierarchy Level [edit event-options]

Release Information Statement introduced in Junos OS Release 7.6.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description For Junos OS event scripts, configure scripting mechanisms.

The statements are explained separately.

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation

- Storing and Enabling Scripts

file

Syntax

```
file filename {
  checksum (md5 | sha-256 | sha1) hash;
  refresh;
  refresh-from url;
  remote-execution {
    remote-hostname {
      passphrase user-password;
      username user-login;
    }
  }
  source url;
}
```

Hierarchy Level [edit event-options event-script]

Release Information Statement introduced in Junos OS Release 7.6.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description For Junos OS event scripts, enable an event script that is located in the **/var/db/scripts/event** directory.

Options *filename*—The name of an Extensible Stylesheet Language Transformations (XSLT) or Stylesheet Language Alternative Syntax (SLAX) file containing an event script.

The statements are explained separately.

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation

- Enabling an Event Script on page 47

refresh (Event Scripts)

Syntax refresh;

Hierarchy Level [edit event-options event-script],
[edit event-options event-script file *filename*]

Release Information Statement introduced in Junos OS Release 9.6.
Statement introduced in Junos OS Release 9.6 for EX Series switches.

Description For Junos OS event scripts, overwrite the local copy of all enabled event scripts or a single enabled script located in the **/var/db/scripts/event** directory with the copy located at the source URL, specified in the **source** statement at the same hierarchy level.

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation
- Using a Master Source Location for a Script
- refresh-from (Event Scripts) on page 94
- source on page 96

refresh-from (Event Scripts)

Syntax refresh-from *url*;

Hierarchy Level [edit event-options event-script],
[edit event-options event-script file *filename*]

Release Information Statement introduced in Junos OS Release 9.6.
Statement introduced in Junos OS Release 9.6 for EX Series switches.

Description For Junos OS event scripts, overwrite the local copy of all enabled event scripts or a single enabled script located in the **/var/db/scripts/event** directory with the copy located at a URL other than the URL specified in the **source** statement.

Options *url*—Source specified as a Hypertext Transfer Protocol (HTTP) URL, FTP URL, or secure copy (scp)-style remote file specification.

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation
- Using an Alternate Source Location for a Script
- refresh (Event Scripts) on page 94
- source on page 96

remote-execution

Syntax

```
remote-execution {
  remote-hostname {
    passphrase user-password;
    username user-login;
  }
}
```

Hierarchy Level [edit event-options event-script file *filename*]

Release Information Statement introduced in Junos OS Release 9.6.
Statement introduced in Junos OS Release 9.6 for EX Series switches.

Description For Junos OS event scripts, enable event scripts to invoke RPCs on a local or remote host.

Options

passphrase *user-password*—User's password for the remote host.

remote-hostname—Name of the remote host with which the event script will communicate.

username *username*—User's login name for the remote host.

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation

- Using RPCs and Operational Mode Commands in Event Scripts on page 43

source

Syntax source *url*;

Hierarchy Level [edit event-options event-script file *filename*]

Release Information Statement introduced in Junos OS Release 9.6.
Statement introduced in Junos OS Release 9.6 for EX Series switches.

Description For Junos OS event scripts, specify the location of the source file for an enabled script located in the **/var/db/scripts/event** directory. When you include the **refresh** statement at the same hierarchy level, the local copy is overwritten by the version stored at the specified URL.

Options *url*—Master source file for an event script specified as an HTTP URL, FTP URL, or scp-style remote file specification.

Required Privilege Level maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation

- refresh (Event Scripts) on page 94
- refresh-from (Event Scripts) on page 94
- Using a Master Source Location for a Script

traceoptions (Event Scripts)

Syntax

```
traceoptions {
  file <filename> <files number> <size size> <world-readable | no-world-readable>;
  flag flag;
  no-remote-trace;
}
```

Hierarchy Level [edit event-options event-script]

Release Information Statement introduced in Junos OS Release 7.6.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Define tracing operations for event scripts.

Default If you do not include this statement, no event script–specific tracing operations are performed.

Options ***filename***—Name of the file to receive the output of the tracing operation. All files are placed in the directory **/var/log**. By default, event script process tracing output is placed in the file **escript.log**. If you include the **file** statement, you must specify a filename. To retain the default, you can specify **escript.log** as the filename.

files ***number***—(Optional) Maximum number of trace files. When a trace file named ***trace-file*** reaches its maximum size, it is renamed and compressed to ***trace-file*.0.gz**. When ***trace-file*** again reaches its maximum size, ***trace-file*.0.gz** is renamed ***trace-file*.1.gz** and ***trace-file*** is renamed and compressed to ***trace-file*.0.gz**. This renaming scheme continues until the maximum number of trace files is reached. Then the oldest trace file is overwritten.

If you specify a maximum number of files, you also must specify a maximum file size with the **size** option and a filename.

Range: 2 through 1000

Default: 10 files

flag—Tracing operation to perform. To specify more than one tracing operation, include multiple **flag** statements. You can include the following flags:

- **all**—Log all operations
- **events**—Log important events
- **input**—Log event script input data
- **offline**—Generate data for offline development
- **output**—Log event script output data
- **rpc**—Log event script RPCs
- **xslt**—Log the XSLT library

no-world-readable—Restrict file access to owner. This is the default.

size *size*—(Optional) Maximum size of each trace file, in kilobytes (KB), megabytes (MB), or gigabytes (GB). When a trace file named ***trace-file*** reaches this size, it is renamed and compressed to ***trace-file*.0.gz**. When ***trace-file*** again reaches its maximum size, ***trace-file*.0.gz** is renamed ***trace-file*.1.gz** and ***trace-file*** is renamed and compressed to ***trace-file*.0.gz**. This renaming scheme continues until the maximum number of trace files is reached. Then the oldest trace file is overwritten.

If you specify a maximum file size, you also must specify a maximum number of trace files with the **files** option and a filename.

Syntax: ***x*k** to specify KB, ***x*m** to specify MB, or ***x*g** to specify GB

Range: 10 KB through 1 GB

Default: 128 KB

world-readable—Enable unrestricted file access.

Required Privilege Level

maintenance—To view this statement in the configuration.
maintenance-control—To add this statement to the configuration.

Related Documentation

- Tracing Event Script Processing on page 110

PART 3

Administration

CHAPTER 11

Event Policy and Event Scripts Configuration Statements

Any Hierarchy Level

The following statement can be added at any level of the configuration:

```
apply-macro apply-macro-name {
  parameter-name parameter-value;
}
```

[edit event-options] Hierarchy Level

The following statements can be included at the **[edit]** hierarchy level:

```
[edit]
event-options {
  destinations {
    destination-name {
      archive-sites {
        url <password password>;
      }
      transfer-delay seconds;
    }
  }
  event-script {
    file filename {
      checksum (md5 | sha-256 | sha1) hash;
      refresh;
      refresh-from url;
      remote-execution {
        remote-hostname {
          passphrase user-password;
          username user-login;
        }
      }
      source url;
    }
```

```
    refresh;
    refresh-from url;
    traceoptions {
      file <filename> <files number> <size size> <world-readable | no-world-readable>;
      flag flag;
      no-remote-trace;
    }
  }
  generate-event event-name {
    time-interval seconds;
    time-of-day hh:mm:ss;
  }
  policy policy-name {
    attributes-match {
      event1.attribute-name equals event2.attribute-name;
      event.attribute-name matches regular-expression;
      event1.attribute-name starts-with event2.attribute-name;
    }
    events [ events ];
    then {
      event-script filename {
        arguments {
          argument-name argument-value;
        }
        destination destination-name {
          retry-count count retry-interval seconds;
          transfer-delay seconds;
        }
        output-filename filename;
        output-format (text | xml);
        user-name name;
      }
      execute-commands {
        commands {
          "command";
        }
        destination destination-name {
          retry-count count retry-interval seconds;
          transfer-delay seconds;
        }
        output-filename filename;
        output-format (text | xml);
        user-name username;
      }
      ignore;
      raise-trap;
      upload filename (filename | committed) destination destination-name {
        retry-count count retry-interval seconds;
        transfer-delay seconds;
        user-name username;
      }
    }
    within seconds {
      events [ events ];
      not events [ events ];
      trigger (after number | on number | until number);
```

```
    }
  }
  traceoptions {
    file filename <files number> <size size> <world-readable | no-world-readable>;
    flag flag;
  }
}
```

PART 4

Troubleshooting

CHAPTER 12

Troubleshooting Event Policy and Event Scripts

Tracing Event Policy Processing

Event policy tracing operations track all event policy operations and record them in a log file. The logged error descriptions provide detailed information to help you solve problems faster.

By default, no events are traced. If you include the **traceoptions** statement at the **[edit event-options]** hierarchy level, the default tracing behavior is the following:

- Important events are logged in a file called **eventd** located in the **/var/log** directory.
- When the file **eventd** reaches 128 kilobytes (KB), it is renamed and compressed to **eventd.0.gz**, then **eventd.1.gz**, and so on, until there are three trace files. Then the oldest trace file (**eventd.2.gz**) is overwritten. (For more information about how log files are created, see the *Junos OS System Log Messages Reference*.)
- Log files can be accessed only by the user who configures the tracing operation.

You cannot change the directory (**/var/log**) to which trace files are written. However, you can customize the other trace file settings by including the following statements at the **[edit event-options traceoptions]** hierarchy level:

```
[edit event-options traceoptions]
file <filename> <files number> <match regular-expression> <size size> <world-readable |
   no-world-readable>;
flag all;
flag configuration;
flag database;
flag events;
flag policy;
flag server;
flag syslog
flag timer-events;
no-remote-trace;
```

These statements are described in the following sections:

Configuring the Event Policy Log Filename

By default, the name of the file that records trace output is **eventd**. You can specify a different name by including the **file** statement at the **[edit event-options traceoptions]** hierarchy level:

```
[edit event-options traceoptions]
file filename;
```

Configuring the Number and Size of Event Policy Log Files

By default, when the trace file reaches 128 kilobytes (KB) in size, it is renamed ***filename*.0**, then ***filename*.1**, and so on, until there are three trace files. Then the oldest trace file (***filename*.2**) is overwritten.

You can configure the limits on the number and size of trace files by including the following statements at the **[edit event-options traceoptions file <*filename*>]** hierarchy level:

```
[edit event-options traceoptions file <filename>]
files number size size;
```

For example, set the maximum file size to 2 MB and the maximum number of files to 20. When the file that receives the output of the tracing operation (***filename***) reaches 2 MB, ***filename*** is renamed and compressed to ***filename*.0.gz** and a new file called ***filename*** is created.

When ***filename*** reaches 2 MB, ***filename*.0.gz** is renamed ***filename*.1.gz** and ***filename*** is renamed and compressed to ***filename*.0.gz**. This process repeats until there are 20 trace files. Then the oldest file (***filename*.19.gz**) is overwritten.

The number of files can range from 2 through 1000 files. The file size can range from 10 KB through 1 gigabyte (GB).

Configuring Access to the Log File

By default, log files can be accessed only by the user who configures the tracing operation.

To specify that any user can read all log files, include the **world-readable** statement at the **[edit event-options traceoptions file <*filename*>]** hierarchy level:

```
[edit event-options traceoptions file <filename>]
world-readable;
```

To explicitly set the default behavior, include the **no-world-readable** statement at the **[edit event-options traceoptions file <*filename*>]** hierarchy level:

```
[edit event-options traceoptions file <filename>]
```

```
no-world-readable;
```

Configuring a Regular Expression for Lines to Be Logged

By default, the trace operation output includes all lines relevant to the logged events.

You can refine the output by including the **match** statement at the **[edit event-options traceoptions file <*filename*>]** hierarchy level and specifying a regular expression to be matched:

```
[edit event-options traceoptions file <filename>]
match regular-expression;
```

Configuring the Trace Operations

By default, only important events are logged. You can configure the trace operations to be logged by including the following statements at the **[edit event-options traceoptions]** hierarchy level:

```
[edit event-options traceoptions]
flag all;
flag configuration;
flag database;
flag events;
flag policy;
flag server;
flag syslog
flag timer-events;
```

Table 4 on page 109 describes the meaning of the event policy tracing flags.

Table 4: Event Policy Tracing Flags

Flag	Description	Default Setting
all	Trace all operations.	Off
configuration	Log reading of configuration at the **[edit event-options]** hierarchy level.	Off
events	Trace important events.	Off
database	Log events involving storage and retrieval in events database.	Off
policy	Log policy processing.	Off
server	Log communication with processes that are generating events.	Off
syslogd	Log syslog related traces	Off
timer-events	Log internally generated events.	Off

To display the end of the log, issue the **show log eventd | last** operational mode command:

```
[edit]
user@host# run show log eventd | last
```

Tracing Event Script Processing

Event script tracing operations track all event script operations and record them in a log file. The logged error descriptions provide detailed information to help you solve problems faster.

The default operation of event script tracing is to log important events in a file called **escript.log** located in the **/var/log** directory. When the file **escript.log** reaches 128 kilobytes (KB), it is renamed with a number 0 through 9 (in ascending order) appended to the end of the file and then compressed. The resulting files are **escript.log.0.gz**, then **escript.log.1.gz**, until there are 10 trace files. Then the oldest trace file (**escript.log.9.gz**) is overwritten. (For more information about how log files are created, see the *Junos OS System Log Messages Reference*.)

This section discusses the following topics:

Minimum Configuration for Enabling Traceoptions for Event Scripts

If no event script trace options are configured, the simplest way to view the trace output of an event script is to configure the **output** trace flag and issue the **show log escript.log | last** command. To do this, perform the following steps:

1. If you have not done so already, enable an event script by including the **file** statement at the **[edit event-options event-script]** hierarchy level:

   ```
   [edit event-options event-script]
   user@host# set file filename
   ```

2. Enable trace options by including the **traceoptions flag output** statement at the **[edit event-options event-script]** hierarchy level:

   ```
   [edit event-options event-script]
   user@host# set traceoptions flag output
   ```

3. Issue the **commit** command:

   ```
   [edit]
   user@host# commit
   ```

4. Display the resulting trace messages recorded in the **/var/log/escript.log** file. At the end of the log is the output generated by the event script you enabled in Step 1 after a configured event policy is triggered and invokes the script. To display the end of the log, issue the **show log escript.log | last** operational mode command:

   ```
   [edit]
   user@host# run show log escript.log | last
   ```

Table 5 on page 111 summarizes useful filtering commands that display selected portions of the **escript.log** file.

Table 5: Event Script Tracing Operational Mode Commands

Task	Command
Display logging data associated with all event script processing.	**show log escript.log**
Display processing for only the most recent operation.	**show log escript.log | last**
Display processing for script errors.	**show log escript.log | match error**
Display processing for a particular script.	**show log escript.log | match *filename***

Example: Minimum Configuration for Enabling Traceoptions for Event Scripts

Display the trace output of the event script file **source-route.xsl**:

```
[edit]
event-options {
   event-script {
      file source-route.xsl;
      traceoptions flag output;
   }
}
```

```
[edit]
user@host# commit
[edit]
user@host# run show log escript.log | last
```

Configuring Tracing of Event Scripts

You cannot change the directory (**/var/log**) to which trace files are written. However, you can customize other trace file settings by including the following statements at the **[edit event-options event-script traceoptions]** hierarchy level:

```
[edit event-options event-script traceoptions]
file <filename> <files number> <size size> <world-readable | no-world-readable>;
flag all;
flag events;
flag input;
flag offline;
flag output;
flag rpc;
flag xslt;
no-remote-trace;
```

These statements are described in the following sections:

Configuring the Event Script Log Filename

By default, the name of the file that records trace output is **escript.log**. You can specify a different name by including the **file** statement at the **[edit event-options event-script traceoptions]** hierarchy level:

```
[edit event-options event-script traceoptions]
file filename;
```

Configuring the Number and Size of Event Script Log Files

By default, when the trace file reaches 128 KB in size, it is renamed and compressed to ***filename*.0.gz**, then ***filename*.1.gz**, and so on, until there are 10 trace files. Then the oldest trace file (***filename*.9.gz**) is overwritten.

You can configure the limits on the number and size of trace files by including the following statements at the **[edit event-options event-script traceoptions file <*filename*>]** hierarchy level:

```
[edit event-options event-script traceoptions file <filename>]
files number size size;
```

For example, set the maximum file size to 640 KB and the maximum number of files to 20. When the file that receives the output of the tracing operation (***filename***) reaches 640 KB, it is renamed and compressed to ***filename*.0.gz**, and a new file called ***filename*** is created. When ***filename*** reaches 640 KB, ***filename*.0.gz** is renamed ***filename*.1.gz** and ***filename*** is renamed and compressed to ***filename*.0.gz**. This process repeats until there are 20 trace files. Then the oldest file (***filename*.19.gz**) is overwritten.

The number of files can range from 2 through 1000 files. The file size can range from 10 KB through 1 gigabyte (GB).

NOTE:

If you set either a maximum file size or a maximum number of trace files, you also must specify the other parameter and a filename.

Configuring Access to Event Script Log Files

By default, access to the event script log file is restricted to the owner. You can manually configure access by including the **world-readable** or **no-world-readable** statement at the **[edit event-options event-script traceoptions file <*filename*>]** hierarchy level.

```
[edit event-options event-script traceoptions file <filename>]
(world-readable | no-world-readable);
```

The **no-world-readable** statement restricts event script log access to the owner. The **world-readable** statement enables unrestricted access to the event script log file.

Configuring the Event Script Trace Operations

By default, only important events are logged. You can configure the trace operations to be logged by including the following statements at the **[edit event-options event-script traceoptions]** hierarchy level:

```
[edit event-options event-script traceoptions]
flag all;
flag events;
flag input;
flag offline;
flag output;
flag rpc;
flag xslt;
```

Table 6 on page 113 describes the meaning of the event script tracing flags.

Table 6: Event Script Tracing Flags

Flag	Description	Default Setting
all	Trace all operations.	Off
events	Trace important events.	On
input	Trace event script input data.	Off
offline	Generate data for offline development.	Off
output	Trace event script output data.	Off
rpc	Trace event script RPCs.	Off
xslt	Trace the Extensible Stylesheet Language Transformations (XSLT) library.	Off

PART 5

Index

Index

Symbols

A

B

C

D

E

U

W

X