CHAPTER 3

MWTM 6.1.5 Event API

The Operations Support and System (OSS) sends Simple Object Access Protocol (SOAP) requests to MWTM 6.1.5. MWTM 6.1.5 responds with SOAP responses.

The MWTM 6.1.5 provides Remote Procedure Call (RPC) style Event API that accepts SOAP messages from OSS and responds with SOAP responses and traps.

The MWTM 6.1.5 sends unsolicited messages via traps to the OSS.

When generating a new event, the MWTM 6.1.5 can be configured to send this event to the Northbound OSS asynchronously via traps.

This chapter includes the following sections:

Event and Alarm Definitions

An event is defined as a singular occurrence at a specific moment in time. Each event is associated with an event ID. An alarm is defined as a sequence of events that occur over a period of time. An alarm is associated with a single alarm ID for the duration of the events that define the alarm.

An example best illustrates this concept. When the temperature in a chassis exceeds a certain threshold, the MWTM reports a Minor alarm. When the temperature increases again, the MWTM escalates the alarm to Major. When the temperature increases a third time, the MWTM escalates the alarm to Critical. The alarm ID for this sequence remains constant.

The MWTM does not consider the clearing condition in a state transition sequence like the one described in the previous paragraph. For example, an ITP link may change state from Normal to Critical to Normal to Warning within the span of an hour. An event is created for each transition: from Normal to Critical, from Critical to Normal, and from Normal to Warning. These three events make up the event sequence of an ITP Link State alarm.

The system or a user deletes (archives) alarms. Cleared alarms are alarms that have the severity Normal. The system archives cleared alarms after one day (this is the default setting). The system archives uncleared alarms after seven days. After the system or a user archives an alarm, if the condition occurs again, the MWTM raises a new alarm with a new alarm ID.

To change alarm archive values, see “Changing Event Limits” in Chapter 9 of the *User Guide for the Cisco Mobile Wireless Transport Manager 6.1.5.*

Setting up MWTM 6.1.5 to Send Asynchronous Events to Northbound OSS

To configure the MWTM 6.1.5 to send asynchronous events to the Northbound OSS, the system administrator must first configure the MWTM 6.1.5 Event Editor using the following procedure (for more information, see the *User Guide for the Cisco Mobile Wireless Transport Manager 6.1.5*).

Procedure

Step 1 Start the MWTM 6.1.5 Event Editor, and load the current running configuration. To launch the MWTM 6.1.5 Event Editor, choose **Start > Programs > Cisco MWTM Client > Launch MWTM Event Editor** from the Windows Start menu. Or, choose **Tools > Event Editor** from the MWTM 6.1.5 main menu.

Step 2 Click the turner icon beside the **Event Configuration,** then click **SNMP Servers**. MWTM 6.1.5 displays the SNMP Servers Configuration window in the right pane, which contains nine fields or buttons. Add the target hostname, port, community string, SNMP version, and trap type that you want to forward the MWTM 6.1.5 events to. The recommended trap type is **CISCO-EPM**.

Step 3 Next, enable trap forwarding for all events by clicking the **Send a trap for all events** radio button in the SNMP Servers Configuration window of the MWTM 6.1.5 Event Editor.

Step 4 Optionally (de)select individual events under **Traps**, **Status Alarms**, or **User Actions** by checking or unchecking (clearing) the check box shown for each event.

Step 5 Optionally customize the Trap Events by clicking the turner icon beside the **Event Editor**, then click the turner icon beside **Traps**. MWTM 6.1.5 lists the currently defined traps in the left pane. To change an event, select the event in the left pane. MWTM 6.1.5 displays the Event Configuration panel in the right pane, which enables you to change all aspects of that event. The Event Configuration panel contains 13 fields or buttons. Set the following fields or buttons: **Category**, **Severity**, **Message Name**, and **Message** options for each event.

Step 6 To receive a trap when an event is updated or deleted, select the **Limits** tree item and edit the **SendUpdates** variable to read true. By default, SendUpdates is false and MWTM 6.1.5 will only send a trap for an event when the event is new.

Step 7 Adjust the **TrapGenThrottle** variable to specify the number of milliseconds to delay between sending traps. This setting can help ensure that MWTM 6.1.5 sends traps at rate that the receiving application can handle. If this value is too low, MWTM 6.1.5 might send traps at a rate that is too fast for the receiving application.

Step 8 Adjust the **HeartbeatTrapInterval** variable to specify the number of seconds to delay between sending a heartbeat trap.

Step 9 Choose **File > Deploy** to save the event configuration.

Step 10 Restart the MWTM 6.1.5 server for the new event configuration to take effect.

Launching the Event and Alarm Properties Dialog

You can create a URL to launch the Properties dialog for events and alarms in the context of their associated node. The Event/Alarms Properties dialog is available at the node-level under the Events and Alarms tab and contains 4 tabs:

- Properties
- Notes
- Details
- Events for alarms (alarm-only)

For an alarm, enter the following command:

```
<protocol>://<web_server>:<web_server_port>/?FQDN=Node=<node
name>&selectionId=alarms&evid=<alarm id>&subSelectionId=<sub_selection_id>
```

where *sub_selection_id* is properties, notes, details, or events.

For an event, enter the following command:

```
<protocol>://<web_server>:<web_server_port>/?FQDN=Node=<node
name>&selectionId=events&evid=<event id>&subSelectionId=<sub_selection_id>
```

where *sub_selection_id* is properties, notes, or details.

Note The *alarm id* and *event id* are both called "event id" in the documentation for the northbound OSS API.

MWTM 6.1.5 Event API Operations

This section lists the MWTM 6.1.5 Event API operations.

All the operations are listed as pseudocode with comments. The syntax for these operations is defined as Web Services Description Language (WSDL). This syntax is described in Appendix A, "MWTM NBAPI WSDL and XSD Definitions".

All MWTM 6.1.5 Event API operations use SOAP fault for error handling. These error codes are defined in Appendix B, "MWTM NBAPI Error Codes".

Get all Events from MWTM

```
int getAllEventsAsTraps (TrapTarget target)
```

This method retrieves all the events from MWTM 6.1.5 as traps.

Parameters

TrapTarget target—Specifies the target to send the MWTM 6.1.5 event traps. The following parameters can be specified:

Hostname—Hostname or IP address to send the traps to.

Port Number —Port number to send the traps to.

Community String—Community string to fit the trap.

SNMP Version—Simple Network Management Protocol (SNMP) version for the traps: 1 or 2c.

MIB—Management Information Base (MIB) format to send the traps: CISCO-SYSLOG-MIB or CISCO-EPM-NOTIFICATION-MIB.

Return Value

Number of events sent as a result of this method.

Get Filtered Events from MWTM

```
int getFilteredEventsAsTraps (TrapTarget target, EventFilter filter)
```

This method retrieves the filtered events from MWTM 6.1.5 as traps.

Parameters

TrapTarget target—Specifies the target to send the MWTM 6.1.5 event traps. The following parameters can be specified:

Hostname—Hostname or IP address to send the traps to.

Port Number—Port number to send the traps to.

Community String—Community string to fit the trap.

SNMP Version—SNMP version for the traps: 1 or 2c.

MIB—MIB format to send the traps: CISCO-SYSLOG-MIB or CISCO-EPM-NOTIFICATION-MIB.

EventFilter filter—Specifies the filter rules to retrieve the MWTM 6.1.5 event. These filters can be specified as standalone or combined together. If multiple filters are specified, they are applied using "AND" logic. The following parameters can be specified:

Event ID—Specifies a list of event IDs to filter.

Start Date—Specifies the starting date to filter the events.

End Date—Specifies the end date to filter the events.

Severity—Specifies a list of severities to filter the events. Valid severities can be customized in the the MWTM 6.1.5 Event Editor.

Category—Specifies a list of categories to filter the events. Valid event categories can be customized in the MWTM 6.1.5 Event Editor.

Acknowledged—Filter based on whether the events are acknowledged.

Cleared—Filter based on whether the events are cleared.

Message Text—Filter based on whether the events contain a given message text.

Forward—Filter based on whether the forward option is turned on for an event. Forward option for the events is configured using the MWTM 6.1.5 Event Editor.

AlarmMode—Filter based on alarms or events.

Return Value

Number of events sent as a result of this method.

Clear Events

```
void clearEvents (EventIDList eventList, String userid, String note)
```

This method clears the specified events.

Parameters

EventIDList eventList—List of the events to clear.

String userid—User ID who cleared the events.

String note—Note explaining the reason for clearing this event.

Return Value

None

Acknowledge Events

```
void acknowledgeEvents (EventIDList eventList, String userid, String note)
```

This method acknowledges the specified events.

Parameters

EventIDList eventList—List of the events to acknowledge.

String userid—User ID who cleared the events.

String note—Note explaining the reason for acknowledging the events.

Return Value

None

Delete Events

```
void deleteEvents (EventIDList eventList)
```

This method deletes the specified events.

Parameters

EventIDList eventList—List of the events to delete.

Return Value

None

Change Event Severity

```
void changeSeverities (EventIDList eventList, String severity, String userid, String note)
```

This method changes severity of specified events.

Parameters

EventIDList eventList—List of the events to change severity

String severity—The target severity to change. Valid severities can be customized in the MWTM 6.1.5 Event Editor.

String userid—User ID who changed the event severity.

String note—Note explaining the reason for changing the severity for the events.

Return Value

None

Get Note for an Event

```
String getNote (long eventID)
```

This method gets an attached note for an event.

Parameters

Long eventID—Event ID to retrieve the note.

Return Value

None

Set Note for an Event

```
String setNote (long eventID, String userid, String note)
```

This method sets an attached note for an event.

Parameters

long eventID—Event ID to set the note.

String userid—User ID who sets the note.

String note—Note text to set to.

Return Value

None

Append Note to an Event

```
String appendNote (long eventID, String userid, String note)
```

This method appends a note to an event.

Parameters

long eventID—Event ID to append the note to.

String userid—User ID who appends the text to the event note.

String note—Text to append to the event note.

Return Value

None