Prime Performance Manager Event API

Prime Performance Manager provides Remote Procedure Call (RPC) Event API that accepts Simple Object Access Protocol (SOAP) OSS requests and responds with SOAP responses and traps. Prime Performance Manager also sends unsolicited messages through traps to the OSS. You can configure Prime Performance Manager to send new events to the northbound OSS asynchronously through traps.

Additionally, you can cross-launch into Prime Performance Manager from an outside application, using

The following topics describe the Prime Performance Manager events API:

Overview to Events and Alarms

An event is a single occurrence at a specific moment in time. Each event has an event ID. An alarm is a sequence of events that occur over a period of time. An alarm has a single alarm ID that exists for the duration of the events that define the alarm. For example, when the chassis temperature exceeds a certain threshold, Prime Performance Manager reports a minor alarm. When the temperature increases again, Prime Performance Manager escalates the alarm to major. When the temperature increases a third time, Prime Performance Manager escalates the alarm to critical. However, the alarm ID for this event sequence remains the same.

Prime Performance Manager does not consider the clearing condition in a state transition sequence like the one described in the example. For example, an ITP link may change state from normal to critical to normal to warning within 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 clears alarms after one day (this is the default setting). The system archives unclears alarms after seven days. After the system or a user archives an alarm, if the condition occurs again, Prime Performance Manager raises a new alarm with a new alarm ID.

Configuring OSS Hosts

Configuring Prime Performance Manager to send events to a northbound OSS is performed using the Prime Performance Manager GUI. For information see "Adding Upstream OSS Hosts" procedure in the *Cisco Prime Performance Manager 1.4 User Guide*.

Prime Performance Manager Event API Operations

The following topics describe Prime Performance Manager event API operations. All the operations are listed as pseudocode with comments. The operations syntax is defined as Web Services Description Language (WSDL). The syntax is described in Event API WSDL and XSD Definitions, page 5-8. Event API operations use SOAP (Simple Object Access Protocol). Error codes are described in Event API Errors, page 5-16.

Event API operations:

Get all Alarms from Prime Performance Manager

```
int getAllAlarmsAsTraps (TrapTarget target)
```

This method retrieves all the alarms from Prime Performance Manager as traps.

Parameters

TrapTarget target—Specifies the target to send the Prime Performance Manager alarm 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-PRIME, CISCO-SYSLOG or CISCO-EPM-2.

Return Value

Number of alarms sent as a result of this method.

Example

```
<message name="getFilteredEventsRequest">
    <part name="filter" type="tns:EventFilter"></part>
  </message>
  <message name="getFilteredEventsResponse">
    <part name="events" type="tns:EventList"></part>
  </message>
```

Get all Open Alarms from Prime Performance Manager

`int getAllOpenAlarmsAsTraps (TrapTarget target)`

This method retrieves all the open alarms from Prime Performance Manager as traps.

Parameters

TrapTarget target—Specifies the target to send the Prime Performance Manager alarm 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-PRIME, CISCO-SYSLOG or CISCO-EPM-2.

Return Value

Number of open alarms sent as a result of this method.

Get all Events from Prime Performance Manager

`int getAllEventsAsTraps (TrapTarget target)`

This method retrieves all the events from Prime Performance Manager as traps.

Parameters

TrapTarget target—Specifies the target to send the Prime Performance Manager 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-PRIME, CISCO-SYSLOG or CISCO-EPM-2.

Return Value

Number of events sent as a result of this method.

Get Filtered Events from Prime Performance Manager

`EventList getFilteredEvents (EventFilter filter)`
This method retrieves a list of filtered events from Prime Performance Manager.

Parameters

- EventFilter—Specifies the rules to retrieve Prime Performance Manager events. Filters can be standalone or combined. 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 Prime Performance Manager Event Editor.
  - Category—Specifies a list of categories to filter the events. Valid event categories can be customized in the Prime Performance Manager 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 Prime Performance Manager Event Editor.
- AlarmMode—Filter based on alarms or events.

Return Value

A list of events sent as a result of this method.

Return Value

```
<message name="getFilteredEventsRequest">
    <part name="filter" type="tns:EventFilter"></part>
  </message>
  <message name="getFilteredEventsResponse">
    <part name="events" type="tns:EventList"></part>
  </message>
```

Get Filtered Events as Traps from Prime Performance Manager

`int getFilteredEventsAsTraps (TrapTarget target, EventFilter filter)`

This method retrieves the filtered events from Prime Performance Manager as traps.

Parameters

TrapTarget target—Specifies the target to send Prime Performance Manager 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-PRIME. CISCO-SYSLOG or CISCO-EPM-2.

EventFilter filter—Specifies the filter rules to retrieve Prime Performance Manager 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 Prime Performance Manager Event Editor.

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

AlarmMode—Filter based on alarms or events.

Return Value

Number of events sent as a result of this method.

Resend Traps

```
void resendTraps(String host, String port)
```

This method resends the traps.

Parameters

- host—Trap destination host.
- port—Trap destination port.

Returns

void

Example

```
<message name="resendTrapsRequest">
  <part name="host" type="xsd:string"/>
```

```
  <part name="port" type="xsd:string"/>
</message>
<message name="resendTrapsResponse">
</message>
```

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

Unacknowledge Events

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

This method unacknowledges the specified events.

Parameters

EventIDList eventList—List of the events to unacknowledge.

String userid—User ID who cleared the events.

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

Return Value

void

Example

```
<message name="unAcknowledgeEventsRequest">
    <part name="eventList" type="tns:EventIDList"/>
    <part name="userid" type="xsd:string"/>
    <part name="note" type="xsd:string"/>
  </message>
  <message name="unAcknowledgeEventsResponse">
  </message>
```

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 Prime Performance Manager 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

Event API WSDL and XSD Definitions

The following topics provide the Prime Performance Manager Event API WSDL and XSD definitions.

- EventAPI.wsdl, page 5-8
- Event.xsd, page 5-14

EventAPI.wsdl

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- Copyright (c) 2006-2013 Cisco Systems, Inc.    All rights reserved. -->

<definitions targetNamespace="http://cisco.com/ppm" name="EventAPIService"
xmlns:tns="http://cisco.com/ppm"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

```
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns="http://schemas.xmlsoap.org/wsdl/">

<types>
<xsd:schema>
<xsd:import namespace="http://cisco.com/ppm" schemaLocation="MWTM.xsd" />
<xsd:import namespace="http://cisco.com/ppm" schemaLocation="Event.xsd" />
<xsd:import namespace="http://cisco.com/ppm" schemaLocation="Common.xsd" />
</xsd:schema>
</types>

<message name="APIStatus">
<part name="APIStatus" element="tns:APIStatus"/>
</message>
<message name="resendTrapsRequest">
<part name="host" type="xsd:string"/>
<part name="port" type="xsd:string"/>
</message>
<message name="resendTrapsResponse">
</message>
<message name="getFilteredEventsRequest">
<part name="filter" type="tns:EventFilter"></part>
</message>
<message name="getFilteredEventsResponse">
<part name="events" type="tns:EventList"></part>
</message>
<message name="getAllOpenAlarmsAsTrapsRequest">
<part name="target" type="tns:TrapTarget"/>
</message>
<message name="getAllOpenAlarmsAsTrapsResponse">
<part name="eventCount" type="xsd:int"/>
</message>
<message name="getAllEventsAsTrapsRequest">
<part name="target" type="tns:TrapTarget"/>
</message>
<message name="getAllEventsAsTrapsResponse">
<part name="eventCount" type="xsd:int"/>
</message>
<message name="getFilteredEventsAsTrapsRequest">
<part name="target" type="tns:TrapTarget"/>
<part name="filter" type="tns:EventFilter"/>
</message>
<message name="getFilteredEventsAsTrapsResponse">
<part name="eventCount" type="xsd:int"/>
</message>
<message name="clearEventsRequest">
<part name="eventList" type="tns:EventIDList"/>
<part name="userid" type="xsd:string"/>
<part name="note" type="xsd:string"/>
</message>
<message name="clearEventsResponse">
</message>
<message name="acknowledgeEventsRequest">
<part name="eventList" type="tns:EventIDList"/>
<part name="userid" type="xsd:string"/>
<part name="note" type="xsd:string"/>
</message>
<message name="acknowledgeEventsResponse">
</message>
<message name="unAcknowledgeEventsRequest">
<part name="eventList" type="tns:EventIDList"/>
<part name="userid" type="xsd:string"/>
<part name="note" type="xsd:string"/>
</message>
```

```
<message name="unAcknowledgeEventsResponse">
</message>
<message name="deleteEventsRequest">
<part name="eventList" type="tns:EventIDList"/>
</message>
<message name="deleteEventsResponse">
</message>
<message name="changeSeveritiesRequest">
<part name="eventList" type="tns:EventIDList"/>
<part name="severity" type="xsd:string"/>
<part name="userid" type="xsd:string"/>
<part name="note" type="xsd:string"/>
</message>
<message name="changeSeveritiesResponse">
</message>
<message name="getNoteRequest">
<part name="eventID" type="xsd:long"/>
</message>
<message name="getNoteResponse">
<part name="note" type="xsd:string"/>
</message>
<message name="setNoteRequest">
<part name="eventID" type="xsd:long"/>
<part name="userid" type="xsd:string"/>
<part name="note" type="xsd:string"/>
</message>
<message name="setNoteResponse">
</message>
<message name="appendNoteRequest">
<part name="eventID" type="xsd:long"/>
<part name="userid" type="xsd:string"/>
<part name="note" type="xsd:string"/>
</message>
<message name="appendNoteResponse">
</message>

<portType name="EventAPI">
<operation name="resendTraps">
<input message="tns:resendTrapsRequest"/>
<output message="tns:resendTrapsResponse"/>
<fault name="APIStatus" message="tns:APIStatus"/>
</operation>
<operation name="getFilteredEvents">
<input message="tns:getFilteredEventsRequest" />
<output message="tns:getFilteredEventsResponse" />
<fault name="APIStatus" message="tns:APIStatus" />
</operation>
<operation name="getAllOpenAlarmsAsTraps">
<input message="tns:getAllOpenAlarmsAsTrapsRequest"/>
<output message="tns:getAllOpenAlarmsAsTrapsResponse"/>
<fault name="APIStatus" message="tns:APIStatus"/>
</operation>
<operation name="getAllEventsAsTraps">
<input message="tns:getAllEventsAsTrapsRequest"/>
<output message="tns:getAllEventsAsTrapsResponse"/>
<fault name="APIStatus" message="tns:APIStatus"/>
</operation>
<operation name="getFilteredEventsAsTraps">
<input message="tns:getFilteredEventsAsTrapsRequest"/>
<output message="tns:getFilteredEventsAsTrapsResponse"/>
<fault name="APIStatus" message="tns:APIStatus"/>
</operation>
<operation name="clearEvents">
<input message="tns:clearEventsRequest"/>
```

```
<output message="tns:clearEventsResponse"/>
<fault name="APIStatus" message="tns:APIStatus"/>
</operation>
<operation name="acknowledgeEvents">
<input message="tns:acknowledgeEventsRequest"/>
<output message="tns:acknowledgeEventsResponse"/>
<fault name="APIStatus" message="tns:APIStatus"/>
</operation>
<operation name="unAcknowledgeEvents">
<input message="tns:unAcknowledgeEventsRequest"/>
<output message="tns:unAcknowledgeEventsResponse"/>
<fault name="APIStatus" message="tns:APIStatus"/>
</operation>
<operation name="deleteEvents">
<input message="tns:deleteEventsRequest"/>
<output message="tns:deleteEventsResponse"/>
<fault name="APIStatus" message="tns:APIStatus"/>
</operation>
<operation name="changeSeverities">
<input message="tns:changeSeveritiesRequest"/>
<output message="tns:changeSeveritiesResponse"/>
<fault name="APIStatus" message="tns:APIStatus"/>
</operation>
<operation name="getNote">
<input message="tns:getNoteRequest"/>
<output message="tns:getNoteResponse"/>
<fault name="APIStatus" message="tns:APIStatus"/>
</operation>
<operation name="setNote">
<input message="tns:setNoteRequest"/>
<output message="tns:setNoteResponse"/>
<fault name="APIStatus" message="tns:APIStatus"/>
</operation>
<operation name="appendNote">
<input message="tns:appendNoteRequest"/>
<output message="tns:appendNoteResponse"/>
<fault name="APIStatus" message="tns:APIStatus"/>
</operation>
</portType>

<binding name="EventAPIPortBinding" type="tns:EventAPI">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc" />
<operation name="resendTraps">
<soap:operationsoapAction="" />
<input>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</input>
<output>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</output>
<fault name="APIStatus">
<soap:fault use="literal" name="APIStatus" />
</fault>
</operation>
<operation name="getFilteredEvents">
<soap:operationsoapAction="" />
<input>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</input>
<output>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</output>
<fault name="APIStatus">
<soap:fault use="literal" name="APIStatus" />
```

```
</fault>
</operation>
<operation name="getAllOpenAlarmsAsTraps">
<soap:operationsoapAction="" />
<input>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</input>
<output>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</output>
<fault name="APIStatus">
<soap:fault name="APIStatus" use="literal" />
</fault>
</operation>
<operation name="getAllEventsAsTraps">
<soap:operationsoapAction="" />
<input>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</input>
<output>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</output>
<fault name="APIStatus">
<soap:fault name="APIStatus" use="literal" />
</fault>
</operation>
<operation name="getFilteredEventsAsTraps">
<soap:operationsoapAction="" />
<input>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</input>
<output>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</output>
<fault name="APIStatus">
<soap:fault name="APIStatus" use="literal" />
</fault>
</operation>
<operation name="clearEvents">
<soap:operationsoapAction="" />
<input>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</input>
<output>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</output>
<fault name="APIStatus">
<soap:fault name="APIStatus" use="literal" />
</fault>
</operation>
<operation name="acknowledgeEvents">
<soap:operationsoapAction="" />
<input>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</input>
<output>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</output>
<fault name="APIStatus">
<soap:fault name="APIStatus" use="literal" />
</fault>
</operation>
<operation name="unAcknowledgeEvents">
<soap:operationsoapAction="" />
```

```
<input>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</input>
<output>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</output>
<fault name="APIStatus">
<soap:fault name="APIStatus" use="literal" />
</fault>
</operation>
<operation name="deleteEvents">
<soap:operationsoapAction="" />
<input>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</input>
<output>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</output>
<fault name="APIStatus">
<soap:fault name="APIStatus" use="literal" />
</fault>
</operation>
<operation name="changeSeverities">
<soap:operationsoapAction="" />
<input>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</input>
<output>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</output>
<fault name="APIStatus">
<soap:fault name="APIStatus" use="literal" />
</fault>
</operation>
<operation name="getNote">
<soap:operationsoapAction="" />
<input>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</input>
<output>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</output>
<fault name="APIStatus">
<soap:fault name="APIStatus" use="literal" />
</fault>
</operation>
<operation name="setNote">
<soap:operationsoapAction="" />
<input>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</input>
<output>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</output>
<fault name="APIStatus">
<soap:fault name="APIStatus" use="literal" />
</fault>
</operation>
<operation name="appendNote">
<soap:operationsoapAction="" />
<input>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</input>
<output>
```

```
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</output>
<fault name="APIStatus">
<soap:fault name="APIStatus" use="literal" />
</fault>
</operation>
</binding>

<service name="EventAPIService">
<port name="EventAPIPort" binding="tns:EventAPIPortBinding">
<soap:address location="REPLACE_WITH_ACTUAL_URL"/>
</port>
</service>

</definitions>
```

Event.xsd

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- Copyright (c) 2006-2013 Cisco Systems, Inc.    All rights reserved. -->

<xs:schema version="1.0"
targetNamespace="http://cisco.com/ppm"
xmlns:tns="http://cisco.com/ppm"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

<!-- A list of event IDs -->
<xs:complexType name="EventIDList">
<xs:sequence>
<xs:element name="ID" type="xs:long" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>

<xs:element name="EventIDList" type="tns:EventIDList"/>

<!-- A list of Events -->
<xs:complexType name="EventList">
<xs:sequence>
<xs:element name="Events" type="tns:Event" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>

<xs:element name="EventList" type="tns:EventList"/>

<!-- Event : derived from com.cisco.mwg.sgm.core.SgmEvent -->
<xs:complexType name="Event">
<xs:sequence>
<xs:element name="Id" type="xs:long" />
<xs:element name="AlarmId" type="xs:long" />
<xs:element name="Owner" type="xs:string" />
<xs:element name="Category" type="xs:string" />
<xs:element name="Severity">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="Critical" />
<xs:enumeration value="Major" />
<xs:enumeration value="Minor" />
<xs:enumeration value="Warning" />
<xs:enumeration value="Informational" />
<xs:enumeration value="Normal" />
<xs:enumeration value="Indeterminate" />
```

```
<xs:enumeration value="Normal" />
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="Message" type="xs:string" />
<xs:element name="NotesExist" type="xs:boolean" />
<xs:element name="AckBy" type="xs:string" />
<xs:element name="ClearBy" type="xs:string" />
<xs:element name="CreateTimestamp" type="xs:dateTime" />
<xs:element name="AckTimestamp" type="xs:dateTime" />
<xs:element name="ClearTimestamp" type="xs:dateTime" />
<xs:element name="ChangeTimestamp" type="xs:dateTime" />
<xs:element name="AlarmNature" type="xs:string" />
<xs:element name="AlarmType" type="xs:string" />
<xs:element name="ProbableCause" type="xs:string" />
<xs:element name="DeviceType" type="xs:string" />
<xs:element name="Condition" type="xs:string" />
<xs:element name="TcaName" type="xs:string" />
<xs:element name="TcaMetric" type="xs:string" />
<xs:element name="ElementName" type="xs:string" />
<xs:element name="OriginalSeverity" type="xs:string" />
<xs:element name="Count" type="xs:long" />
<xs:element name="isAlarm" type="xs:boolean" />
</xs:sequence>
</xs:complexType>

<xs:element name="Event" type="tns:Event"/>

<!-- Trap Target specifies target host/port and SNMP parameters to send
       SNMP Trap notification to -->
<xs:complexType name="TrapTarget">
<xs:sequence>
<xs:element name="Hostname" type="xs:string"/>
<xs:element name="Port" type="xs:int"/>
<xs:element name="Community" type="xs:string"/>
<xs:element name="SNMPVersion">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="1"/>
<xs:enumeration value="2c"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="MIB">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="CISCO-PRIME"/>
<xs:enumeration value="CISCO-EPM-2"/>
<xs:enumeration value="CISCO-SYSLOG"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
</xs:sequence>
</xs:complexType>

<xs:element name="TrapTarget" type="tns:TrapTarget"/>

<!-- Event Filter -->
<!-- If more than one conditions are specified,
EventFilter applies "AND" on all specified conditions
   -->
<xs:complexType name="EventFilter">
<xs:sequence>
<xs:element name="EventIDs" type="tns:EventIDList" minOccurs="0"/>
```

```
<xs:element name="StartAlarmId" type="xs:int" minOccurs="0"/>
<xs:element name="EndAlarmId" type="xs:int" minOccurs="0"/>
<xs:element name="StartAlarmChangeTime" type="xs:dateTime" minOccurs="0"/>
<xs:element name="EndAlarmChangeTime" type="xs:dateTime" minOccurs="0"/>
<xs:element name="StartDate" type="xs:dateTime" minOccurs="0"/>
<xs:element name="EndDate" type="xs:dateTime" minOccurs="0"/>
<xs:element name="Severity" type="xs:string"
minOccurs="0" maxOccurs="unbounded"/>
<xs:element name="Category" type="xs:string"
minOccurs="0" maxOccurs="unbounded"/>
<xs:element name="Acknowledged" type="xs:boolean" minOccurs="0"/>
<xs:element name="Cleared" type="xs:boolean" minOccurs="0"/>
<!-- filter "text" is based on whether event message contains
given text -->
<xs:element name="MessageText" type="xs:string" minOccurs="0"/>
<xs:element name="NetworkElement" type="xs:string" minOccurs="0"/>
<xs:element name="Forward" type="xs:boolean" minOccurs="0"/>
<xs:element name="AlarmMode" type="xs:int" minOccurs="0"/>
</xs:sequence>
</xs:complexType>

<xs:element name="EventFilter" type="tns:EventFilter"/>

</xs:schema>
```

Event API Errors

Detailed error information is defined as APIStatus in WSDL (for Event API WSDL definitions, see Appendix 5, "Event API WSDL and XSD Definitions"):

```
<xs:complexType name="APIStatus">
  <xs:sequence>
    <xs:element name="StatusCode" type="xs:int"/>
    <xs:element name="Message" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

APIStatus contains a status code and a message. Table 5-1 lists the possible Event API status codes.

Table 5-1 Status Codes

Status Code	Error	Description
1000	UNEXPECTED_ERROR	An unexpected error occurred that stops the requested operation.
2000	INVALID_PARAMETER	The requestor provided one or more invalid parameters in the requested operation.

Cross Launching Prime Performance Manager

To cross launch into Prime Performance Manager, generate URLs in the following format:

<server>:<port>/ppm/jsp/main.jsp?

Required parameters:

- displayType [results,reportTab]—Shows only the table and graphs; reportTab also includes the header bar. This parameter shows the entire Prime Performance Manager inner frame without the XWT shell.
- FQDN [objectId]—Specifies an object by FQDN.
- resultsType <string>—The report to run. The string must match the list provided in the Type field in the Prime Performance Manager GUI, for example, Interface Utilization Hourly. You must specify a reportID with either FQDN or resultsType.

Optional Parameters:

- durationSelect <string>—Sets a report duration. You can use this parameter instead of a start date and end date to the dates you want from the following:
  - lastHour—valid for hourly reports only
  - last24Hours—valid for hourly or daily reports
  - last7Days—valid for all reports
  - last30Days—valid for daily reports only
  - last90Days—valid for daily reports only
- startDate <msecs>—Report start date in milliseconds since 1970. If this parameter is provided, the endDate must be provided.
- endDate <msecs>—Report end date in milliseconds since 1970. The endDate must be greater than the start date. If this parameter is provided, the startDate must be provided.
- outputType <string>—Output type: CSV, TABLE, or GRAPH.
- topStatSel <string>—Default data sorting. The provided string must match the string that appears in the GUI drop down.
- screenWidth <integer>—The screen width. The default is 1000 pixels.
- screenHeight <integer>—The screen height. The default is 700 pixels.
- clientOffSetTime <msecs>—The offset for the the client time zone. If not provided, the report start and end dates use the server time zone.
- fullScreen <integer>—The graph index to show as full screen. This is a 0 index number.
- hideButtons<boolean>—Hides the graph Zoom, Graph Style, and Export options.
- hideTitle <boolean>—Hides the graph title.
- hideNavigator <boolean>—Hides the navigator in leaf and full screen graphs.
- hideLegend <boolean>—Hides the legend in graphs.
- hideDateString <boolean>—Hides the date in graphs.