Event API

This chapter describes the Event API:

- Using the Event API, page 4-1
- Event API Methods, page 4-2
- Handling Event Notifications On the Client Side, page 4-10
- Push Mechanisms, page 4-12

Using the Event API

In your CG-NMS NB API client application, use this CG-NMS server URL to access the Event API WSDL:

http://<server_address>/nbapi/event?wsdl

For example:

http://10.27.167.19/nbapi/event?wsdl
Event API Methods

- searchEvents, page 4-3
- subscribeForEvents, page 4-4
- unSubscribeForEvents, page 4-6
- subscribeForCgmeshOutage, page 4-7
- unSubscribeForCgmeshOutage, page 4-9
searchEvents

The searchEvent call lets you search for events based on device type, event name, event time, and event severity.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>query</td>
<td>string</td>
<td>Search query string.</td>
</tr>
<tr>
<td>count</td>
<td>integer</td>
<td>Number of results to be retrieved.</td>
</tr>
<tr>
<td>offset</td>
<td>integer</td>
<td>Position of the first result.</td>
</tr>
</tbody>
</table>

In the query, you can use these parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Delimiters</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>deviceType</td>
<td>=</td>
<td>Device type. Can be one of the following: asr1000, cgr1000, cgmesh.</td>
</tr>
<tr>
<td>eventName</td>
<td>=</td>
<td>User-defined name, for example: up, down, outage.</td>
</tr>
<tr>
<td>eventTime</td>
<td>&gt;</td>
<td>&lt;</td>
</tr>
<tr>
<td>eventSeverity</td>
<td>=</td>
<td>Severity level. Can be one of the following: critical, major, minor, info.</td>
</tr>
</tbody>
</table>

Results

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>subscriptionid</td>
<td>long</td>
<td>Subscription ID, which can be used by the listener to identify the subscription the response is coming from.</td>
</tr>
<tr>
<td>events</td>
<td>List&lt;EventDetail&gt;</td>
<td>EventDetail</td>
</tr>
</tbody>
</table>

searchEvents SOAP XML Request Format

```xml
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:event="http://event.nbapi.cgms.cisco.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <event:searchEvents>
      <!--Optional:-->  
      <query>deviceType:cgmesh eventName:up</query>
      <!--Optional:-->  
      <count>4</count>
      <!--Optional:-->  
      <offset>0</offset>
    </event:searchEvents>
  </soapenv:Body>
</soapenv:Envelope>
```
subscribeForEvents

The subscribeForEvents call streams a set of events to the API listener, based on the query. Event subscriptions can be based on device type, event name, or severity. Listener registers the URL and specifies the push window. After every configured eventPushWindowSec event push window, all new events received in this window are delivered to the registered URL. Subscription-based events notification uses the same query language as searchEvents, page 4-3, except that the eventTime attribute-based queries cannot be subscribed with, and should throw unsuccessful subscription errors.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>soapEndPointUrl</td>
<td>string</td>
<td>Address of the WSDL, implemented by your client, that receives event notifications from the CG-NMS NB API. For more information, see Handling Event Notifications On the Client Side, page 4-10. http://&lt;server_address&gt;:8445/&lt;path&gt;&lt;api&gt;?wsdl For example: <a href="http://localhost:8445/event?wsdl">http://localhost:8445/event?wsdl</a></td>
</tr>
<tr>
<td>query</td>
<td>string</td>
<td>Query string. Note Query language eventTime field cannot be inside the subscription.</td>
</tr>
<tr>
<td>eventPushWindowSec</td>
<td>integer</td>
<td>Event Push window time, in seconds. The query will be executed after x seconds, and the results will be pushed out to the endPointUrl wsdl format defined.</td>
</tr>
</tbody>
</table>

Table 4-5 Subscribe for Events Response

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>subscriptionId</td>
<td>long</td>
<td>Subscription ID</td>
</tr>
</tbody>
</table>

The QueryResult will have a Status field indicated if the subscription has been a success or failure.

subscribeForEvents SOAP XML Request Format

```xml
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                   xmlns:event="http://event.nbapi.cgms.cisco.com/"

  <soapenv:Header/>

  <soapenv:Body>
        <event:subscribeForEvents>

            <!--Optional:-->
            <soapEndPointUrl>http://128.107.155.95:8455/nbapi/pushevent?wsdl</soapEndPointUrl>

            <!--Optional:-->
            <query>deviceType:cgmesh eventName:registered</query>

            <!--Optional:-->
            <eventPushWindowSec>21</eventPushWindowSec>

        </event:subscribeForEvents>

  </soapenv:Body>
```

Cisco Connected Grid NMS NB API Programming Guide
</soapenv:Envelope>
unSubscribeForEvents

The unSubscribeForEvents call unsubscribes the given listener event query.

Parameters

<table>
<thead>
<tr>
<th>Table 4-6</th>
<th>unSubscribeForEvents Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
<td>Type</td>
</tr>
<tr>
<td>soapEndPointUrl</td>
<td>string</td>
</tr>
<tr>
<td>query</td>
<td>string</td>
</tr>
</tbody>
</table>

Results

<table>
<thead>
<tr>
<th>Table 4-7</th>
<th>unSubscribeForEvents Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>Type</td>
</tr>
<tr>
<td>subscriptionId</td>
<td>long</td>
</tr>
</tbody>
</table>

The QueryResult will have a Status field indicated if the subscription has been a success or failure.

unSubscribeForEvents SOAP XML Request Format

```xml
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:event="http://event.nbapi.cgms.cisco.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <event:unSubscribeForEvents>
      <!--Optional:-->  
      <soapEndPointUrl>http://128.107.155.95:8455/nbapi/pushevent?wsdl</soapEndPointUrl>
      <!--Optional:-->  
      <query>deviceType:cgmesh(eventName:registered)</query>
    </event:unSubscribeForEvents>
  </soapenv:Body>
</soapenv:Envelope>
```
subscribeForCgmeshOutage

This call is similar to subscribeForEvents, except that subscribeForCgmeshOutage is for outage and restoration events. Up to 10 subscribers (listeners) at a time can register for these events.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| soapEndPointUrl   | string| Address of the WSDL, implemented by your client, that receives outage notifications from the CG-NMS NB API. For more information, see Handling Event Notifications On the Client Side, page 4-10.  
http://<server_address>:8445/<path><api>?wsdl  
For example:  

To configure the amount of time in seconds after which CG-NMS pushes batches of outage events and Restoration Events to all subscribers, set the value of the event-Outage-push-sec parameter in the /opt/cgms/conf/cgms.properties file. For example, to set event-Outage-push-sec to 30, add this line to the file:

event-Outage-push-sec=30

When CG-NMS pushes outage events to subscribers, only those subscribers that are up receive the events. The subscribers that are down (they do not respond) do not receive those events even after they come back online, but they receive the next outage event push.

For very fast outage event push, set event-Outage-push-sec to a value as low as 1 second. If you set the push value to 1 second, CG-NMS executes a job to find and push new events in the queue.

Results

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>subscriptionId</td>
<td>long</td>
<td>Subscription ID which can be used by the listener to identify for which subscription they are getting the response from.</td>
</tr>
</tbody>
</table>

The QueryResult has a Status field indicating whether the subscription succeeded or failed.

subscribeForCgmeshOutage SOAP XML Request Format

```xml
  <soapenv:Header/>
  <even:subscribeForCgmeshOutage/>
</soapenv:Body>
```

</soapenv:Envelope>
unSubscribeForCgmeshOutage

The unSubscribeForCgmeshOutage call is used to unsubscribe the given listener.

Parameters

Table 4-10 describes the parameters in the request.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>soapEndPointUrl</td>
<td>string</td>
<td>Address of the Event WSDL service. The soapEndPointUrl identifies the subscription. http://&lt;server_address&gt;:8445/&lt;path&gt;&lt;api&gt;?wsdl</td>
</tr>
</tbody>
</table>

Results

Table 4-11 unSubscribeForCgmeshOutage Results

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>subscriptionId</td>
<td>long</td>
<td>Subscription ID</td>
</tr>
</tbody>
</table>

The QueryResult has a Status field indicating whether the unsubscribe succeeded or failed.

unSubscribeForCgmeshOutage SOAP XML Request Format

```xml
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/
xmlns:even="http://event.nbapi.cgms.cisco.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <even:unSubscribeForCgmeshOutage/>
    <!--Optional:-->
  </soapenv:Body>
</even:unSubscribeForCgmeshOutage>
</soapenv:Body>
</soapenv:Envelope>
```
Handling Event Notifications On the Client Side

When subscribing for an event type, your CG-NMS NB API client must implement a Web Service that implements the WSDL for handling event notifications sent by the CG-NMS NB API. The WSDL you implement must provide the receiveEvents() method, which the CG-NMS NB API uses to send event notifications to your client.

Example

In the example in Figure 4-1, the CG-NMS NB API client implements the Outage WSDL (outage?wsdl). When the client subscribes for outage events, CG-NMS uses the CG-NMS NB API to call the method receiveEvents() on the CG-NMS NB API client.

Figure 4-1  Event Notification Handling

This is the Event WSDL that your client must implement to receive outage notifications from CG-NMS.

Event Notification Handling WSDL (Client Side)

Your client must implement this WSDL:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<wSDL:definitions xmlns:soap="http://schemas.xmlsoap.org/soap/"
   xmlns:tns="http://pushevent.nbapi.cgms.cisco.com/"
   xmlns:wSDL="http://schemas.xmlsoap.org/wsd1/soap/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://pushevent.nbapi.cgms.cisco.com/">
   <wSDL:types>
     <xsd:schema targetNamespace="http://pushevent.nbapi.cgms.cisco.com/">
       <xsd:complexType name="receiveEvents">
         <xsd:sequence>
           <xsd:element minOccurs="0" name="eventQueryResult"
             type="tns:eventQueryResult" />
         </xsd:sequence>
       </xsd:complexType>
       <xsd:complexType name="eventQueryResult">
         <xsd:complexContent>
           <xsd:extension base="tns:queryResult">
             <xsd:sequence>
               <xsd:element minOccurs="0" name="eventQueryResult"
                 type="tns:eventQueryResult" />
             </xsd:sequence>
           </xsd:extension>
         </xsd:complexContent>
       </xsd:complexType>
     </xsd:schema>
   </wSDL:types>
</wSDL:definitions>
```
name="events" nillable="true" type="tns:eventDetail" />
</xsd:sequence>
</xsd:complexType>
</xsd:complexType>
<xsd:complexType abstract="true" name="queryResult">
<xsd:sequence>
    <xsd:element minOccurs="0" name="queryId" type="xsd:string" />
    <xsd:element minOccurs="0" name="queryStatus" type="xsd:string" />
</xsd:sequence>
</xsd:complexType>
</xsd:complexType>
<xsd:complexType name="eventDetail">
<xsd:sequence>
    <xsd:element minOccurs="0" name="eid" type="xsd:string" />
    <xsd:element minOccurs="0" name="eventMessage" type="xsd:string" />
    <xsd:element minOccurs="0" name="eventSeverity" type="xsd:string" />
    <xsd:element minOccurs="0" name="eventTime" type="xsd:long" />
    <xsd:element minOccurs="0" name="eventTypeName" type="xsd:string" />
    <xsd:element minOccurs="0" name="meterId" type="xsd:string" />
</xsd:sequence>
</xsd:complexType>
</xsd:schema>
</wsdl:types>
<wsdl:message name="receiveEvents">
    <wsdl:part element="tns:receiveEvents" name="receiveEvents" />
</wsdl:message>
<wsdl:portType name="EventPushService">
    <wsdl:operation name="receiveEvents">
        <wsdl:input>
            <soap:body use="literal" />
        </wsdl:input>
    </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="EventPushServiceBinding" type="tns:EventPushService">
    <soap:binding style="document" 
        transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="receiveEvents">
        <wsdl:input>
            <soap:body use="literal" />
        </wsdl:input>
    </wsdl:operation>
</wsdl:binding>
<wsdl:service name="EventPushService">
    <wsdl:port bindings="tns:EventPushServiceBinding" name="EventPushService">
        <soap:address location="http://127.0.0.1:8008" />
    </wsdl:port>
</wsdl:service>
</wsdl:definitions>
**Push Mechanisms**

Push mechanisms work only when the OMS Server has successfully completed the subscription, as defined in the `subscribeForEvents` and `subscribeForCgmeshOutage` API methods.

A successful subscription leads to generation of the Subscription ID, which the subscriber will be notified of. The Subscriber can use this Subscription ID later for tracking the event push it receives.

CG-NMS has a Scheduled Job that runs every $x$ seconds, and the $x$ seconds can be configurable during the subscription using the `event-Outage-push-sec` global parameter defined in the `/opt/cgms/conf/cgms.properties` file.

After every $x$ seconds, CG-NMS generates an EventList and pushes it to the subscribers defined in the `soapEndPointUrl`.

The web service that should be implemented on the OMS side is:

```java
public void receiveEvents(EventQueryResult eventQueryResult) throws java.rmi.RemoteException;
```

**QueryResult**

The query result has a Status field indicating whether the subscription succeeded or failed.

Table 4-12 describes the parameters in the response.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>subscriptionId</td>
<td>long</td>
<td>Subscription ID, which can be used by the listener to identify for which subscription they are getting the response from.</td>
</tr>
<tr>
<td>events</td>
<td>List&lt;EventDetail&gt;</td>
<td>Details about the event.</td>
</tr>
</tbody>
</table>

Table 4-13 describes the parameters in the response.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eid</td>
<td>string</td>
<td>Serial number for the CGR, and MAC address for the mesh endpoint.</td>
</tr>
<tr>
<td>eventMessage</td>
<td>string</td>
<td>Message related to the event.</td>
</tr>
<tr>
<td>eventTime</td>
<td>long</td>
<td>Time in milliseconds.</td>
</tr>
<tr>
<td>eventTypeName</td>
<td>string</td>
<td>Type of event</td>
</tr>
<tr>
<td>meterId</td>
<td>string</td>
<td>Meter ID corresponding to the EID. For this value to be returned, you must included in the import file when importing meters in CG-NMS.</td>
</tr>
</tbody>
</table>
Example

This is an example of the XML content that the subscriber receives:

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Header>
    <seam:conversationId xmlns:seam="http://www.jboss.org/seam/webservice">110</seam:conversationId>
  </env:Header>
  <env:Body>
    <ns2:searchEventsResponse xmlns:ns2="http://event.nbapi.cgms.cisco.com/">
      <eventQueryResult>
        <subscriptionId>2</subscriptionId>
        <queryId></queryId>
        <queryStatus>SUCCEEDED</queryStatus>
        <events>
          <eid>NE01</eid>
          <eventMessage>Device is Up</eventMessage>
          <eventSeverity>INFO</eventSeverity>
          <eventTime>1314656731899</eventTime>
          <eventTypeName>up</eventTypeName>
          <meterId>Sjc123</meterId>
        </events>
        <events>
          <eid>NE01</eid>
          <eventMessage>Outage detected on this device</eventMessage>
          <eventSeverity>CRITICAL</eventSeverity>
          <eventTime>1314656731908</eventTime>
          <eventTypeName>outage</eventTypeName>
          <meterId>Sjc123</meterId>
        </events>
        <events>
          <eid>NE01</eid>
          <eventMessage>Device has been Restored from Outage</eventMessage>
          <eventSeverity>INFO</eventSeverity>
          <eventTime>1314656771923</eventTime>
          <eventTypeName>restoration</eventTypeName>
          <meterId>Sjc123</meterId>
        </events>
        <events>
          <eid>NE01</eid>
          <eventMessage>Device is Up</eventMessage>
          <eventSeverity>INFO</eventSeverity>
          <eventTime>1314656771933</eventTime>
          <eventTypeName>up</eventTypeName>
          <meterId>Sjc123</meterId>
        </events>
        <subscriptionId>2</subscriptionId>
      </eventQueryResult>
    </ns2:searchEventsResponse>
  </env:Body>
</env:Envelope>
```

The subscriptionId XML element (<subscriptionId>2</subscriptionId>) tells the receiver that this push is for the subscription whose ID is equal to 2.