Fehlerbehebung API-basierte EPNM-Benachrichtigungen

Inhalt

Einleitung

In diesem Dokument wird die Fehlerbehebung für EPNM-Benachrichtigungen beschrieben, wenn die REST-API für den Zugriff auf Gerätefehlerinformationen verwendet wird.

Voraussetzungen

Der von Ihnen implementierte Client muss in der Lage sein, die beiden Mechanismen zu verarbeiten und zu abonnieren, die vom Evolved Programmable Network Manager (EPNM) zum Senden von Benachrichtigungen verwendet werden.

Anforderungen

Es gibt keine spezifischen Anforderungen für dieses Dokument.

Verwendete Komponenten

Dieses Dokument ist nicht auf bestimmte Software- und Hardware-Versionen beschränkt.

Die Informationen in diesem Dokument beziehen sich auf Geräte in einer speziell eingerichteten Testumgebung. Alle Geräte, die in diesem Dokument benutzt wurden, begannen mit einer gelöschten (Nichterfüllungs) Konfiguration. Wenn Ihr Netzwerk in Betrieb ist, stellen Sie sicher, dass Sie die möglichen Auswirkungen aller Befehle kennen.

EPNM-API-Benachrichtigungen

Benachrichtigungen benachrichtigen Netzwerkadministratoren und -betreiber über wichtige Ereignisse oder Probleme im Zusammenhang mit dem Netzwerk. Diese Benachrichtigungen tragen dazu bei, dass potenzielle Probleme schnell erkannt und behoben werden, wodurch Ausfallzeiten reduziert und die Netzwerkleistung insgesamt verbessert wird.

EPNM kann verschiedene Methoden verarbeiten, z. B. Benachrichtigungen per E-Mail, SNMP-Traps (Simple Network Management Protocol) an bestimmte Empfänger oder Syslog-Meldungen an externe Syslog-Server. Zusätzlich zu diesen Methoden bietet EPNM auch eine REST-API (Representational State Transfer Application Programming Interface), mit der Informationen zu Bestand, Alarmen, Serviceaktivierung, Vorlagenausführung und Hochverfügbarkeit abgerufen werden können.

API-basierte Benachrichtigungen werden derzeit unter Verwendung von zwei verschiedenen Mechanismen unterstützt:

- **Verbindungsorientierte Benachrichtigungen:**Der Client abonniert eine vordefinierte URL und verwendet einen WebSocket-Client mit grundlegender Authentifizierung über einen sicheren HTTPS-Kanal.
- **Verbindungslose Benachrichtigungen:** Es wird erwartet, dass der Benutzer über einen REST-Webdienst verfügt, der in der Lage ist, Payloads für erweiterbare Markup-Sprachen (XML) und/oder JavaScript Object Notation (JSON) als POST-Anforderung zu akzeptieren.

Alle Benachrichtigungen haben dasselbe Schema und können im JSON- oder XML-Format abgerufen werden.

EPNM-Basiskonfiguration

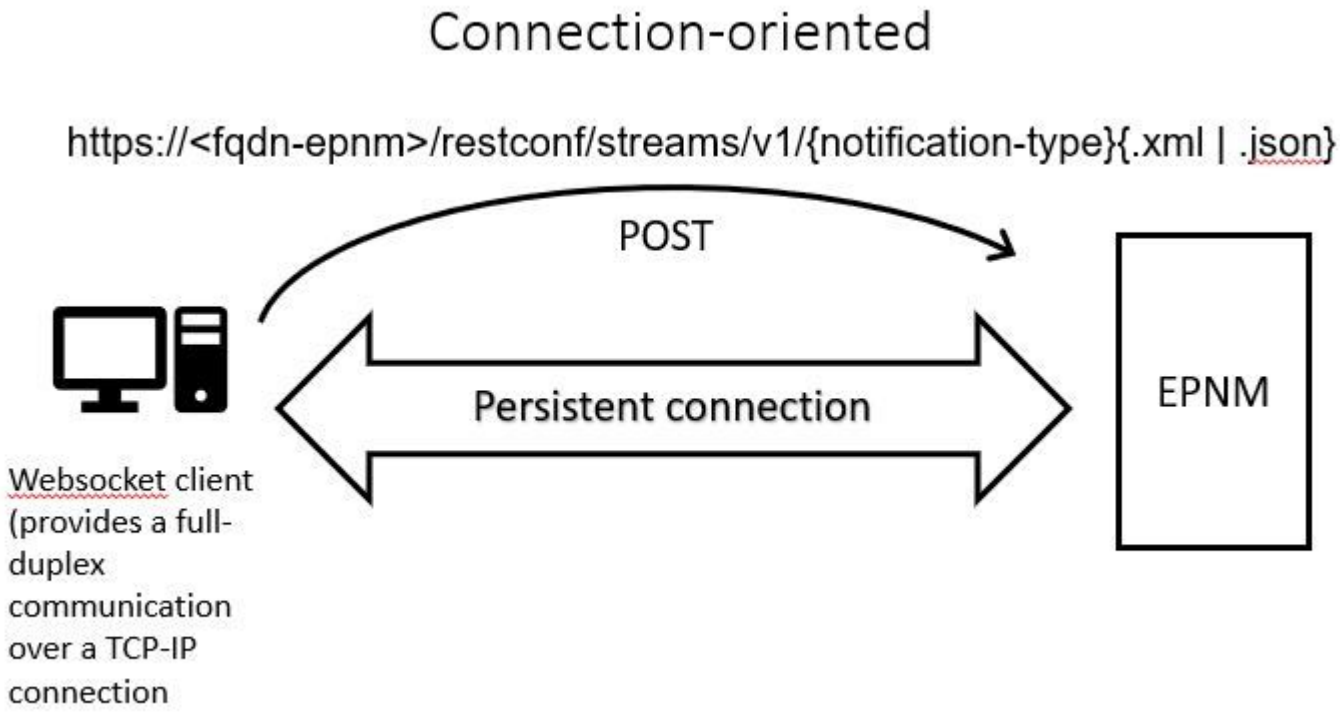
Standardmäßig sind Alarm- und Bestandsbenachrichtigungen deaktiviert. Um sie zu aktivieren, ändern Sie die restconf-config.properties Datei wie angegeben (es ist nicht erforderlich, die EPNM-Anwendung neu zu starten):

```
/opt/CSCOlumos/conf/restconf/restconf-config.properties

epnm.restconf.inventory.notifications.enabled=true
epnm.restconf.alarm.notifications.enabled=true
```

Verbindungsorientierte Benachrichtigungen

In der Abbildung führt der Client-Computer einen WebSocket aus und abonniert den EPNM mit einer vordefinierten URL, einer grundlegenden Authentifizierung und über einen sicheren HTTPS-Kanal.



Ausführen eines WebSockets Python-Clients

Die WebSocket-Clientbibliothek in Python kann verwendet werden, um ein WebSocket auf dem Clientcomputer zu erstellen.

```
import websocket
import time
import ssl
import base64

def on_message(ws, message):
    print(message)

def on_error(ws, error):
    print(error)

def on_close(ws, close_status_code, close_msg):
    print("### closed \###")

def on_open(ws):
    ws.send("Hello, Server!")

if __name__ == "__main__":
    username = "username"
    password = "password"
    credentials = base64.b64encode(f"{username}:{password}".encode("utf-8")).decode("utf-8")
    headers = {"Authorization": f"Basic {credentials}"}
    websocket.enableTrace(True)
    ws = websocket.WebSocketApp("wss://10.122.28.3/restconf/streams/v1/inventory.json",
                                on_message=on_message,
                                on_error=on_error,
                                on_close=on_close,
                                header=headers)
```

```
    ws.on_open = on_open
    ws.run_forever(sslopt={"cert_reqs": ssl.CERT_NONE})
```

Abonnement eines verbindungsorientierten Clients

Mit diesem Code wird ein WebSocket-Client eingerichtet, der EPNM abonniert unter **wss://10.122.28.3/restconf/streams/v1/inventory.json**. Es verwendet den Python WebSocket-Bibliothek, um die Verbindung herzustellen und ein- und ausgehende Nachrichten zu verarbeiten. Das Abonnement kann auch sein (auf der Grundlage der Art der Benachrichtigung, die Sie abonnieren möchten):

/restconf/streams/v1/alarm{.xml | .json}

/restconf/streams/v1/service-activation{.xml | .json}

/restconf/streams/v1/template-execution{.xml | .json}

/restconf/streams/v1/all{.xml | .json}

Die Fehlermeldung on_message, on_error und on_close -Funktionen sind Rückruffunktionen, die aufgerufen werden, wenn die WebSocket-Verbindung eine Nachricht empfängt, auf einen Fehler trifft bzw. geschlossen wird. Die Fehlermeldung **on_open** -Funktion ist ein Rückruf, der aufgerufen wird, wenn die WebSocket-Verbindung hergestellt und einsatzbereit ist.

Die Fehlermeldung username und password -Variablen werden auf die Anmeldeinformationen festgelegt, die für den Zugriff auf den Remoteserver erforderlich sind. Diese Anmeldeinformationen werden dann mit dem base64 -Moduls und wurde den Headern der WebSocket-Anforderung hinzugefügt.

Die Fehlermeldung run_forever -Methode wird vom WebSocket-Objekt aufgerufen, um die Verbindung zu starten und unbegrenzt geöffnet zu halten und auf vom Server ausgehende Nachrichten zu warten. Die Fehlermeldung sslopt -Parameter wird verwendet, um die SSL/TLS-Optionen für die Verbindung zu konfigurieren. Die Fehlermeldung CERT_NONE deaktiviert die Zertifizierungsvalidierung.

Führen Sie den Code aus, damit WebSocket für den Empfang der Benachrichtigungen bereit ist:

```
(env) devasc@labvm:~/epnm$ python conn-oriented.py
--- request header ---
GET /restconf/streams/v1/inventory.json HTTP/1.1
Upgrade: websocket
Host: 10.122.28.3
Origin: https://10.122.28.3
Sec-WebSocket-Key: shY1K9SqXTphBqaZFh/iMQ==
Sec-WebSocket-Version: 13
Connection: Upgrade
Authorization: Basic cm9vdDpQYXNzMTIzNA==


-----------------------
--- response header ---
HTTP/1.1 101
Set-Cookie: JSESSIONID=5BFB68B0126226A0A13ABE595DC63AC9; Path=/restconf; Secure; HttpOnly
Strict-Transport-Security: max-age=31536000;includeSubDomains
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Upgrade: websocket
Connection: upgrade
```

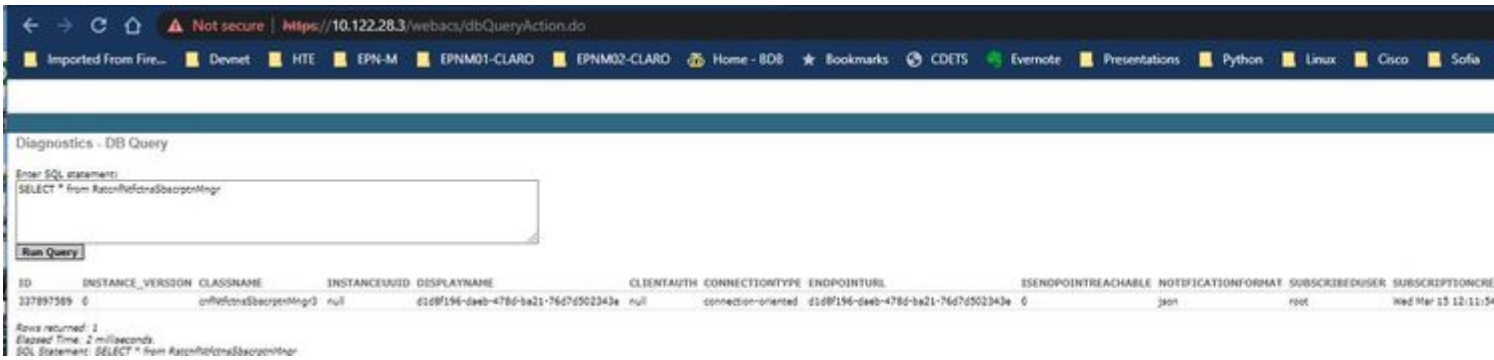
```
Sec-WebSocket-Accept: Ozns7PGgHjrXj0nAgnlhbyVKPjc=
Date: Thu, 30 Mar 2023 16:18:19 GMT
Server: Prime
----------------------
Websocket connected
++Sent raw: b'\x81\x8es\x99ry;\xfc\x1e\x15\x1c\xb5R*\x16\xeb\x04\x1c\x01\xb8'
++Sent decoded: fin=1 opcode=1 data=b'Hello, Server!'
++Rcv raw: b'\x81\x0eHello, Server!'
++Rcv decoded: fin=1 opcode=1 data=b'Hello, Server!'
Hello, Server!
```

Mit dieser DB-Abfrage können Sie die Benachrichtigungsabonnements für den Server überprüfen (navigieren Sie zu https://

/webacs/dbQueryAction.do

).

```
SELECT * from RstcnfNtfctnsSbscrptnMngr;
```



Aus der EPNM-Online-Dokumentation wird nach der Einrichtung die gleiche Verbindung über den gesamten Lebenszyklus der Anwendung aufrechterhalten:

- bis der Client die Verbindung zum Server trennt
- bis der Server entweder aus Wartungsgründen oder während eines Failovers ausfällt

Wenn Sie aus irgendeinem Grund ein bestimmtes Abonnement löschen müssen, können Sie eine HTTP DELETE Anforderung an die SUBSCRIPTIONID angegeben in der URL https://

. Beispiele:

```
devasc@labvm:~/epnm$ curl --location --insecure --request DELETE 'https://10.122.28.3/restconf/data/v1/c
> --header 'Accept: application/json' \
> --header 'Content-Type: application-json' \
> --header 'Authorization: Basic cm9vdDpQYXNzMTIzNA==' \
> --header 'Cookie: JSESSIONID=2CB4F43E3D4BCE5F42411114065F6292'
```

Überprüfung von Meldungen, DEBUG-Einträge, show log**, Verwendeter Dateiname, SQL-Ausgaben**

Um zu ermitteln, warum ein Client, der einen verbindungsorientierten Mechanismus verwendet, keine richtigen Benachrichtigungen empfängt, können Sie die angegebene DB-Abfrage ausführen und überprüfen, ob das Abonnement vorhanden ist. Wenn es nicht vorhanden ist, bitten Sie den Client-Besitzer, sicherzustellen, dass das Abonnement ausgegeben wird.

In der Zwischenzeit können Sie die DEBUG-Stufe in com.cisco.nms.nbi.epnm.restconf.notifications.handler.NotificationsHandlerAdapter So können Sie es abfangen, wenn das Abonnement gesendet wird:

```
[root@epnm-spo-lab-host SCRIPTS]# sudo /opt/CSCOlumos/bin/setLogLevel.sh com.cisco.nms.nbi.epnm.restconf
Loglevel set to DEBUG for com.cisco.nms.nbi.epnm.restconf.notifications.handler.NotificationsHandlerAdap
```

Nachdem das Abonnement gesendet wurde, können Sie überprüfen, ob ein Eintrag mit der IP-Adresse des WebSocket Clients in localhost_access_log.txt:

```
[root@epnm-spo-lab-host SCRIPTS]# zgrep -h '"GET /restconf/streams/.* HTTP/1.1" 101' $(ls -1t /opt/CSCOl
```

```
10.134.4.35 - - [30/Mar/2023:15:33:43 -0300] "GET /restconf/streams/v1/all.json HTTP/1.1" 101 -
```

Überprüfen Sie abschließend noch einmal die DB (beachten Sie, dass der Zeitstempel mit dem Eintrag in localhost_access_log.txt.

Das nächste Protokoll zeigt, wann die POST-Abonnementanforderungen gesendet werden:

```
[root@epnm-spo-lab-host SCRIPTS]# grep -Eh 'DEBUG com.cisco.nms.nbi.epnm.restconf.notifications.handler.
```

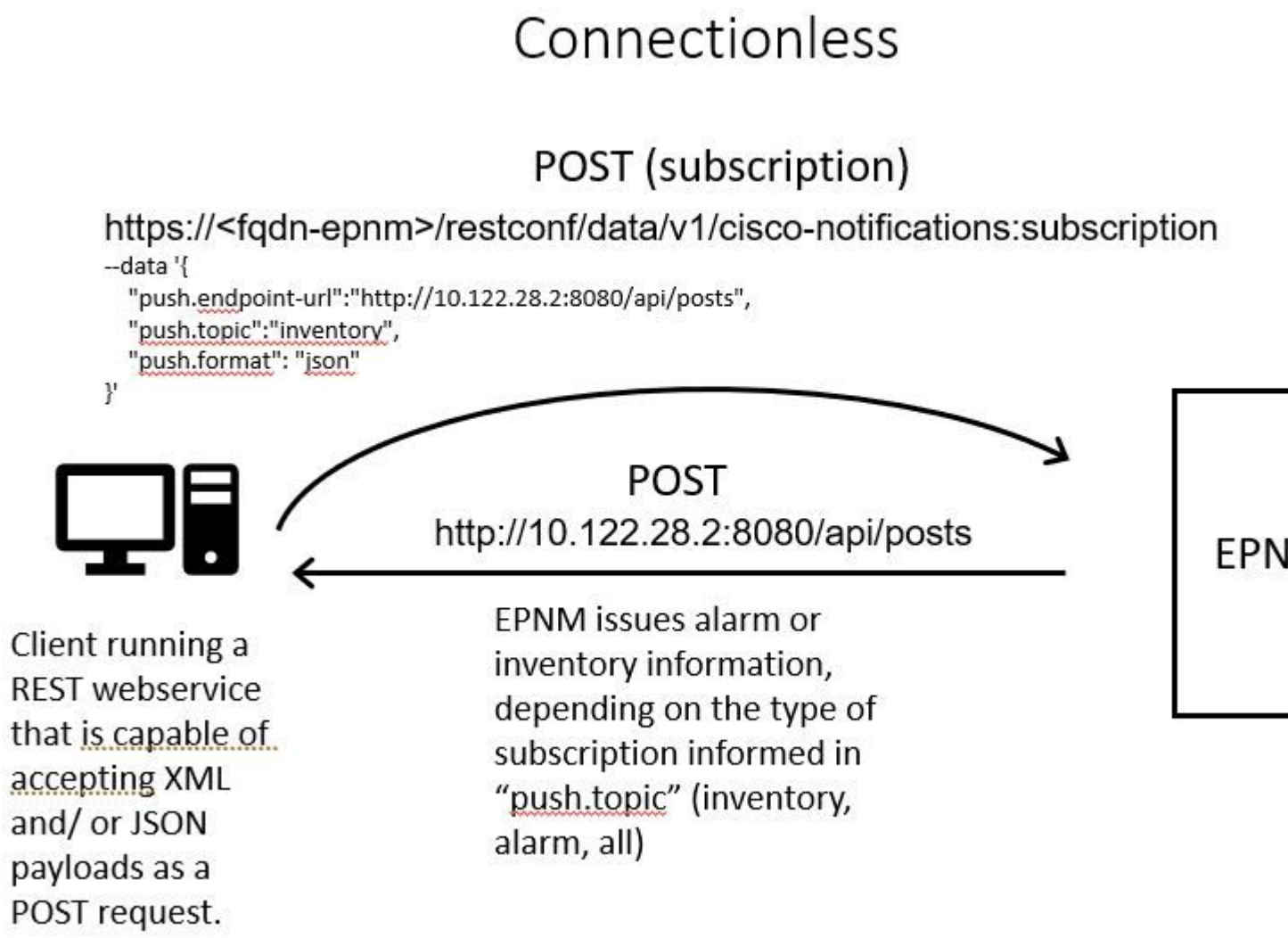
```
2023-03-30 15:33:43,399: DEBUG com.cisco.nms.nbi.epnm.restconf.notifications.handler.Notification
```

Solange die Verbindung aufrechterhalten wird, wird eine Benachrichtigung vom Typ "push-change-update" vom EPN-M-Server an alle Clients gesendet, die Benachrichtigungen abonniert haben. Das Beispiel zeigt eine der Benachrichtigungen, die vom EPNM gesendet werden, wenn der Hostname eines NCS2k geändert wird:


```
{
  "push.push-change-update":{
    "push.notification-id":2052931975556780123,
    "push.topic":"inventory",
    "push.time-of-update":"2023-03-31 13:50:36.608",
    "push.time-of-update-iso8601":"2023-03-31T13:50:39.681-03:00",
    "push.operation":"push:modify",
    "push.update-data":{
      "nd.node":{
        "nd.description":"SOFTWARE=ONS,IPADDR=10.10.1.222,IPMASK=255.255.255.0,DEFRTR=255.255.255.255,IP
        "nd.equipment-list":"",
        "nd.fdn":"MD=CISCO_EPNM!ND=tcc222c",
        "nd.sys-up-time":"217 days, 14:40:170.00"
      }
    }
  }
}
```


Verbindungslose Benachrichtigungen

Als Nächstes wird der Workflow im Fall von connectionless Benachrichtigungen:



Ausführen eines REST-Webservice-Python-Clients

Es wird erwartet, dass der Benutzer über einen REST-Webdienst verfügt, der XML- und/oder JSON-Payloads als POST-Anforderung akzeptieren kann. Dieser REST-Dienst ist der Endpunkt, an den der Cisco EPNMDas restconf-Benachrichtigungs-Framework veröffentlicht Benachrichtigungen. Dies ist eine Beispiel für einen auf dem Remote-Computer zu installierenden REST-Webdienst:

```
from flask import Flask, request, jsonify

app = Flask(__name__)

@ app.route('/api/posts', methods=['POST'])
def create_post():
    post_data = request.get_json()
    response = {'message': 'Post created successfully'}
    print(post_data)
    return jsonify(response), 201

if __name__ == '__main__':
    app.run(debug=True, host='10.122.28.2', port=8080)
```

Dies ist eine Python Flask-Webanwendung, die einen einzelnen Endpunkt definiert. /api/posts die **HTTP POST** Anforderungen. Die Fehlermeldung create_post() -Funktion immer dann aufgerufen, wenn eine **HTTP POST** Anforderung erfolgt an /api/posts. Innerhalb des create_post() -Funktion, werden die Daten aus der eingehenden Anforderung unter Verwendung von **request.get_json()**, der ein Dictionary der JSON-Nutzlast zurückgibt. Die Nutzlast wird dann mit **print(post_data)** für Debugzwecke. Anschließend wird eine Antwortnachricht mit dem Schlüssel erstellt message und Wert **Post created successfully** (im Wörterbuchformat). Diese Antwortnachricht wird dann an den Client mit dem HTTP-Statuscode 201 (erstellt) zurückgegeben.

Die Fehlermeldung if __name__ == '__main__': block ist ein standardmäßiges Python-Konstrukt, das überprüft, ob das Skript als Hauptprogramm ausgeführt wird, anstatt als Modul importiert zu werden. Wenn das Skript als Hauptprogramm ausgeführt wird, startet es die Flask-Anwendung und führt sie an der angegebenen IP-Adresse und dem angegebenen Port aus. Die Fehlermeldung **debug=True** -Argument aktiviert den Debugmodus, der detaillierte Fehlermeldungen und ein automatisches Neuladen des Servers bereitstellt, wenn Änderungen am Code vorgenommen werden.

Führen Sie das Programm aus, um das REST Webdienst:

```
(venv) [apinelli@centos8_cxlabs_spo app]$ python connectionless.py
 * Serving Flask app 'connectionless' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://10.122.28.2:8080/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 117-025-064
```

Abonnement eines verbindungslosen Clients

Der Benutzer abonniert Benachrichtigungen: die REST-Dienstendpunkt wird zusammen mit dem Thema gesendet, für das Sie ein Abonnement erstellen möchten. In diesem Fall lautet das Thema all.

```
curl --location -X POST --insecure 'https://10.122.28.3/restconf/data/v1/cisco-notifications:subscriptio
--header 'Accept: application/json' \
--header 'Content-Type: application-json' \
--header 'Authorization: Basic cm9vdDpQYXNzMTIzNA==' \
--data '{
    "push.endpoint-url":"http://10.122.28.2:8080/api/posts",
    "push.topic":"all",
    "push.format": "json"
}'
```

Die erwartete Antwort ist eine Antwort aus dem Jahr 2010, zusammen mit den Details aus dem Abonnement im Text der Antwort:

```
{
   "push.notification-subscription":{
      "push.subscription-id":6243853653106271664,
      "push.subscribed-user":"root",
      "push.endpoint-url":"http:\/\/10.122.28.2:8080\/api\/posts",
      "push.topic":"all",
      "push.creation-time":"Fri Mar 31 17:07:48 BRT 2023",
      "push.creation-time-iso8601":"2023-03-31T17:07:48.159-03:00",
      "push.time-of-update":"Fri Mar 31 17:07:48 BRT 2023",
      "push.time-of-update-iso8601":"2023-03-31T17:07:48.159-03:00",
      "push.format":"json",
      "push.connection-type":"connection-less"
   }
}
```


Die Liste der Benachrichtigungen, für die der Benutzer mit einer GET-Anforderung abonniert wurde, kann abgerufen werden:

```
curl --location --insecure 'https://10.122.28.3/restconf/data/v1/cisco-notifications:subscription' \
--header 'Accept: application/json' \
--header 'Content-Type: application-json' \
--header 'Authorization: Basic cm9vdDpQYXNzMTIzNA=='
```

Die Antwort lautet:


```
{
   "com.response-message":{
      "com.header":{
         "com.firstIndex":0,
         "com.lastIndex":0
      },
      "com.data":{
         "push.notification-subscription":{
            "push.subscription-id":6243853653106271664,
            "push.subscribed-user":"root",
            "push.endpoint-url":"http:\/\/10.122.28.2:8080\/api\/posts",
            "push.session-id":0,
            "push.topic":"all",
            "push.creation-time":"Fri Mar 31 17:07:48 BRT 2023",
            "push.time-of-update":"Fri Mar 31 17:07:48 BRT 2023",
            "push.format":"json",
            "push.connection-type":"connection-less"
         }
      }
   }
}
#2
```

Überprüfung von Meldungen, DEBUG-Einträge, show log, **Verwendeter Dateiname, SQL-Ausgaben**

Beachten Sie, dass es nur ein Abonnement für all **("push.topic": "all")**. Sie können dies mit einer Abfrage an die Datenbank bestätigen (beachten Sie, dass der Abonnementtyp 'verbindungslos' ist und die SUBSCRIPTIONID entspricht der Ausgabe des GET (gelb hervorgehoben):

Wenn Sie ein verbindungsloses Abonnement löschen müssen, können Sie eine HTTP DELETE-Anforderung mit der Abonnement-ID senden, die Sie löschen möchten. Angenommen, Sie möchten die **Abonnement-ID** 6243853653106271664 löschen:

```
curl --location --insecure --request DELETE 'https://10.122.28.3/restconf/data/v1/cisco-notifications:su
--header 'Accept: application/json' \
--header 'Content-Type: application-json' \
--header 'Authorization: Basic cm9vdDpQYXNzMTIzNA=='
```

Wenn Sie die DB erneut abfragen, werden keine Einträge angezeigt:

Wenn eine Bestandsänderung erfolgt, druckt der Client die Benachrichtigungen (die vom gleichen Typ sind wie die connection-oriented Benachrichtigungen, die im Abschnitt über connected-oriented Clients), gefolgt von der Antwort 201:

```
(venv) [apinelli@centos8_cxlabs_spo app]$ python connectionless.py
 * Serving Flask app 'connectionless' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://10.122.28.2:8080/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 117-025-064
{'push.push-change-update': {'push.notification-id': -2185938612268228828, 'push.topic': 'inventory', 'p
10.122.28.3 - - [31/Mar/2023 16:47:23] "POST /api/posts HTTP/1.1" 201 -
{'push.push-change-update': {'push.notification-id': -1634959052215805274, 'push.topic': 'inventory', 'p
10.122.28.3 - - [31/Mar/2023 16:47:27] "POST /api/posts HTTP/1.1" 201 -
```

Schlussfolgerung

In diesem Dokument werden die beiden Typen von API-basierten Benachrichtigungen beschrieben, die in EPNM konfiguriert werden können (connectionlessund connection-oriented) werden erläutert und die Beispiele der jeweiligen Clients, die als Basis für Simulationszwecke genutzt werden können, angegeben.

Referenzen

- https://<fqdn-epnm>/nbi_help/component.html?comp_id=Notification%20Subscriptions%20Retrieval&api=restconf
- https://www.cisco.com/c/dam/en/us/td/docs/net_mgmt/epn_manager/RESTConf/Cisco_EPN_Manager_RESTConf_NBI_Guide_5_1_2.zip
- https://www.cisco.com/c/dam/en/us/td/docs/net_mgmt/epn_manager/RESTConf/Cisco_Evolved_Programmable_Network_Manager_5_1_2_REST
- Technischer Support und Dokumentation für Cisco Systeme

Informationen zu dieser Übersetzung

Cisco hat dieses Dokument maschinell übersetzen und von einem menschlichen Übersetzer editieren und korrigieren lassen, um unseren Benutzern auf der ganzen Welt Support-Inhalte in ihrer eigenen Sprache zu bieten. Bitte beachten Sie, dass selbst die beste maschinelle Übersetzung nicht so genau ist wie eine von einem professionellen Übersetzer angefertigte. Cisco Systems, Inc. übernimmt keine Haftung für die Richtigkeit dieser Übersetzungen und empfiehlt, immer das englische Originaldokument (siehe bereitgestellter Link) heranzuziehen.