对基于API的EPNM通知进行故障排除

目录

简介

本文档介绍在使用REST API访问设备故障信息时，如何排除EPNM通知故障。

背景信息

您实施的客户端必须能够处理和订用演进可编程网络管理器(EPNM)用来发送通知的两种机制中的任何一种。

EPNM API通知

通知可提醒网络管理员和操作员注意与网络相关的重要事件或问题。 这些通知有助于确保快速检测和解决潜在问题，从而减少停机时间并提高整体网络性能。

EPNM可以处理不同的方法，例如通过电子邮件发送通知、发送到指定接收者的简单网络管理协议(SNMP)陷阱或发送到外部系统日志服务器的系统日志消息。除了这些方法外，EPNM还提供一个具象状态传输应用编程接口(REST API)，可用于检索有关库存、警报、服务激活、模板执行和高可用性的信息。

当前支持基于API的通知使用两种不同的机制：

- 面向连接的通知：客户端订用预定义的URL，并通过安全的HTTPS通道使用具有基本身份验证的WebSocket客户端。
- 无连接通知：用户应具有能够接受可扩展标记语言(XML)和/或JavaScript对象表示法(JSON)负载作为POST请求的REST Web服务。

所有通知共享相同的架构，可以以JSON或XML格式检索。

基本EPNM配置

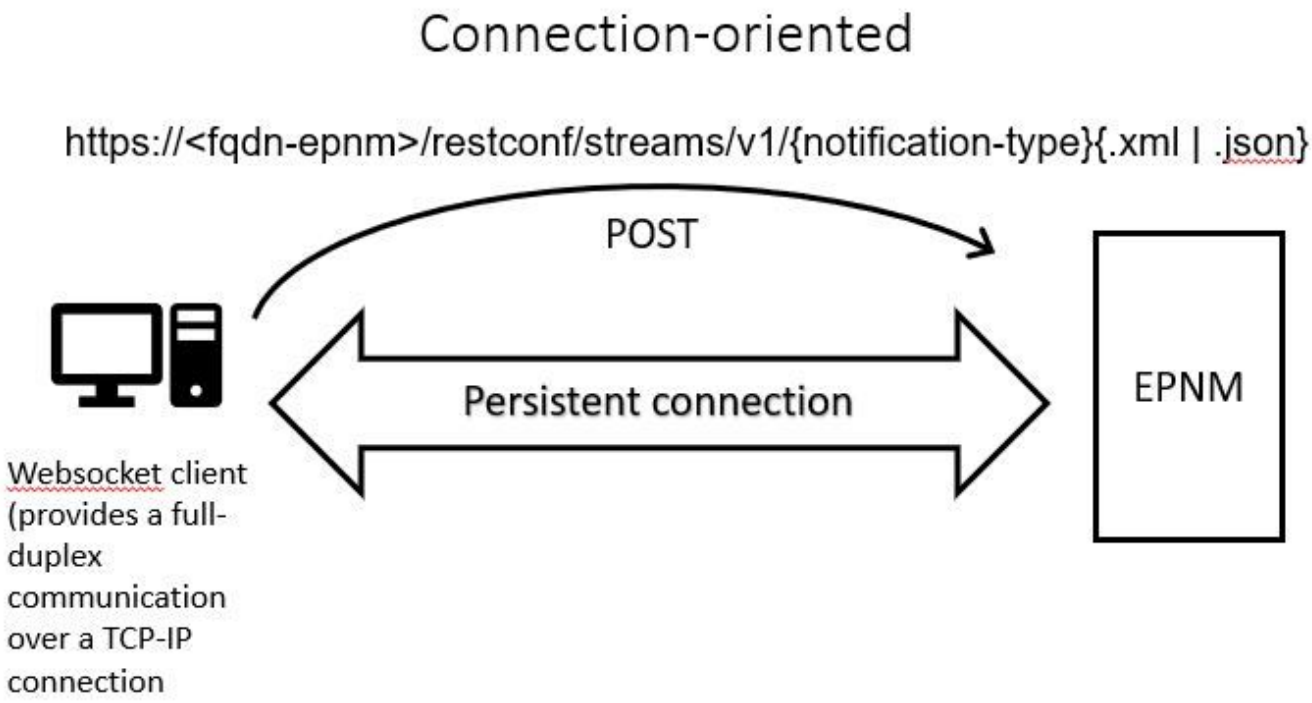
默认情况下，警报和库存通知处于禁用状态。要启用它们，请更改 restconf-config.properties 文件（无需重新启动EPNM应用）：

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

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

面向连接的通知

在图片中，客户端运行WebSocket并通过预定义的URL、基本身份验证和安全HTTPS通道订阅EPNM。



运行WebSockets Python客户端

Python中的WebSocket-client库可用于在客户端机器中创建WebSocket。

```
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})
```

订用面向连接的客户端

此代码设置一个在以下位置订用EPNM的WebSocket客户端：**wss://10.122.28.3/restconf/streams/v1/inventory.json**. 它使用Python WebSocket库，以便建立连接并处理传入和传出消息。订用也可以（根据要订用的通知类型）：

- /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}

此 on_message, on_error 和 on_close 函数是回调函数，在WebSocket连接分别收到消息、遇到错误或关闭时调用这些函数。此 **on_open** 函数是当WebSocket连接已建立并准备使用时调用的回调。

此 username 和 password 变量设置为访问远程服务器所需的登录凭证。然后，这些凭证将使用 base64 模块并添加到WebSocket请求的报头。

此 run_forever 对WebSocket对象调用方法，以启动连接，使其无限期打开，并侦听来自服务器的消息。此 sslopt 参数用于配置连接的SSL/TLS选项。 此 CERT_NONE flag禁用证书验证。

运行代码以使WebSocket准备好接收通知：

```
(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: YYYYYYYYYYY
Sec-WebSocket-Version: 13
Connection: Upgrade
Authorization: Basic XXXXXXXXXXXX


-----------------------
--- 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!
```

您可以使用以下数据库查询检查服务器的通知订阅：

```
ade # ./sql_execution.sh "SELECT * from RstcnfNtfctnsSbscrptnMngr WHERE CONNECTIONTYPE = 'connection-or
```

为了更好地将 conn-oriented.txt 文件（是数据库查询的结果），您可以使用类似工具将其转换为 HTML aha （此处在Ubuntu机器中说明了其用途）：

```
devasc@labvm:~/tmp$ sudo apt-get install aha
devasc@labvm:~/tmp$ cat conn-oriented.txt | aha > conn-oriented.html
```

然后打开 conn-oriented.html 浏览器中的文件：

ID	INSTANCE_VERSION	CLASSNAME	CONNECTIONTYPE	ENDPOINTURL	SUBSCRIBEDUSER	SUBSCRIPTIONCREATIONTIME	SUBSCRIPTIONID	SUBSCRIPTIONTOPIC	SUBSCRIPTIONUPDATETIME
2361930571	0	cnfNtfctnsSbscrptnMngr3	connection-oriented	de4d9082-c05c-439b-a7e7-65016623fee1	root	Mon Aug 28 16:13:04 BRT 2023	3648313822269611499	inventory	Mon Aug 28 16:13:04 BRT 2023

根据EPNM在线文档，一旦建立，相同的连接将在应用的整个生命周期内保持活动状态：

- 直到客户端与服务器断开连接
- 直到服务器因维护或故障转移而停机

如果由于某种原因，您需要删除特定订用，您可以发送 HTTP DELETE 请求 SUBSCRIPTIONID 在URL中指定 https://
.例如：

```
devasc@labvm:~/tmp$ curl --location --insecure --request DELETE 'https://10.122.28.3/restconf/data/v1/c
```

验证消息、DEBUG条目 show log，使用的文件名， SQL输出

为了排除使用面向连接的机制的客户端无法正确接收通知的原因，您可以运行指示的数据库查询，并检查订阅是否存在。如果不存在此订用，请要求客户端所有者确保发出此订用。

同时，您可以在以下位置启用调试级别 com.cisco.nms.nbi.epnm.restconf.notifications.handler.NotificationsHandlerAdapter 以便您在每次发送订用时都捕获它：

```
ade # sudo /opt/CSCOlumos/bin/setLogLevel.sh com.cisco.nms.nbi.epnm.restconf.notifications.handler.Noti
```

发送订用后，您可以检查中是否显示包含WebSocket客户端IP地址的条目 localhost_access_log.txt:

```
ade # zgrep -h '"GET /restconf/streams/.* HTTP/1.1" 101' $(ls -1t /opt/CSCOlumos/logs/localhost_access_
```

最后，再次检查数据库（注意时间戳与中的条目匹配） localhost_access_log.txt).

H	CONNECTIONTYPE	ENDPOINTURL	ISENDPOINTREACHABLE	NOTIFICATIONFORMAT	SUBSCRIBEDUSER	SUBSCRIPTIONCREATIONTIME	SUBSCRIPTIONID	SUBSCRIPTIOI
	connection-oriented	852a674a-e3d0-4ecc-8ea0-787af30f1305	0	json	root	Mon Aug 28 22:17:06 BRT 2023	8743327441517764088	inventory

下一个日志显示何时发送订阅的POST请求：

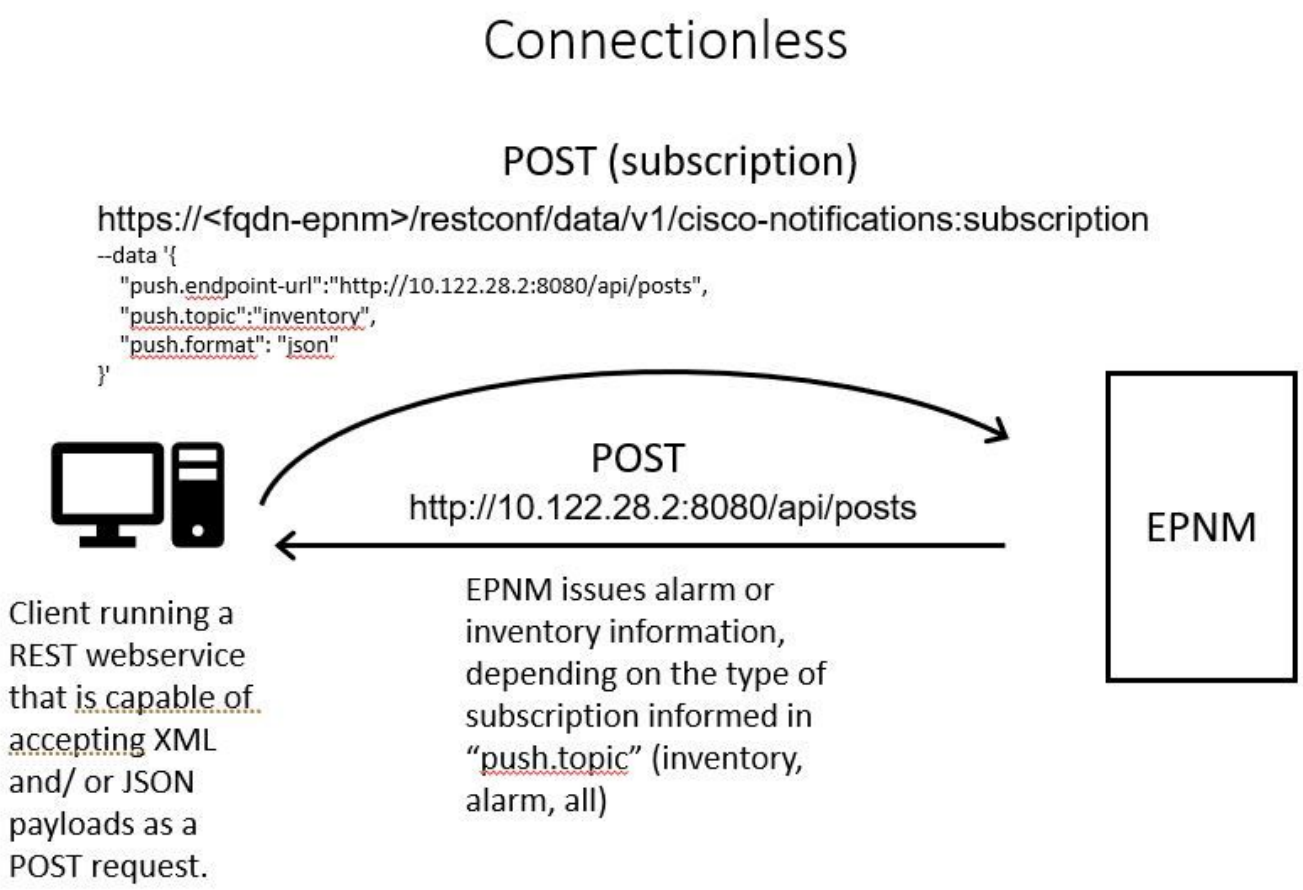
```
ade # grep -Eh 'DEBUG com.cisco.nms.nbi.epnm.restconf.notifications.handler.NotificationsHandlerAdapter
```

只要连接保持活动状态，EPN-M服务器就会向订用通知的所有客户端发送类型推送更改更新的通知。此示例显示当NCS2k的主机名更改时EPNM发送的一个通知：

```
{ "push.push-change-update":{ "push.notification-id":2052931975556780123, "push.topic":"inventory", "pu
```

无连接通知

下一个是工作流程， connectionless 通知：



运行REST Webservice Python客户端

用户应具有能够接受XML和/或JSON负载作为POST请求的REST Web服务。此REST服务是 思科EPNMrestconf notifications framework发布通知。这是 要在远程计算机上安装的REST Web服务示例：

```
from flask import Flask, request, jsonify app = Flask(__name__) @ app.route('/api/posts', methods=['POS
```

这是定义单个终端的Python Flask Web应用 /api/posts 接受 **HTTP POST** 请求。此 create_post() 每当调用时， **HTTP POST** 请求发送到 /api/posts. 内部 create_post() 函数，使用来检索来自传入请求的数据 **request.get_json()**，返回JSON负载的词典。然后，有效负载会打印为 **print(post_data)** 用于调试目的。然后，使用密钥创建响应消息 message 和价值 **Post created successfully** （字典格式）。此响应消息随后会返回到HTTP状态代码为201（已创建）的客户端。

此 if __name__ == '__main__': block是标准Python构造，检查脚本是否作为主程序运行，而不是作为模块导入。如果脚本作为主程序运行，它会启动Flask应用程序并在指定的IP地址和端口上运行该脚本。此 **debug=True** 参数启用调试模式，该模式在对代码进行更改时提供详细的错误消息和自动重新加载服务器。

运行程序以启动 REST web 服务:

```
(venv) [apinelli@centos8_cxlabs_spo app]$ python connectionless.py * Serving Flask app 'connectionless'
```

无连接客户端的订用

用户订用通知： REST服务终端与主题一起发送以订购。在本例中，主题是 all.

```
[apinelli@centos8_cxlabs_spo ~]$ curl --location -X POST --insecure 'https://10.122.28.3/restconf/data/
```

预期响应为201响应，以及响应正文中订阅的详情：

```
{ "push.notification-subscription": { "push.subscription-id": 7969974728822328535, "push.subscribed-use
```

可以通过GET请求获取用户订阅的通知列表：

```
curl --location --insecure 'https://10.122.28.3/restconf/data/v1/cisco-notifications:subscription' \ --
```

答复如下：

```
{ "com.response-message": { "com.header": { "com.firstIndex": 0, "com.lastIndex": 1 }, "com.data": { "p
```

验证消息、DEBUG条目 show log, 使用的文件名， SQL输出

从回应中注意到，有两个订用：一个用于 all (**"push.topic": "all"**) 一个用于库存 (**"push.topic": "inventory"**).您可以通过查询数据库来确认它(请注意，订用的类型为“无连接”，并且 SUBSCRIPTIONID 字段与 GET 命令以黄色突出显示):

```
ade # ./sql_execution.sh "SELECT * from RstcnfNtfctnsSbscrptnMngr WHERE CONNECTIONTYPE = 'connection-le
```

ID	INSTANCE_VERSION	CLASSNAME	CONNECTIONTYPE	ENDPOINTURL	SUBSCRIBEDUSER	SUBSCRIPTIONCREATIONTIME	SUBSCRIPTIONID	SUBSCRIPTIONTOPIC
2361930573	0	cnfNtfctnsSbscrptnMngr3	connection-less	http://10.122.28.2:8080/api/posts	root	Tue Aug 29 10:02:05 BRT 2023	7969974728822328535	all
337897630	0	cnfNtfctnsSbscrptnMngr3	connection-less	http://10.122.28.2:8080/api/posts	root	Fri Mar 31 17:45:47 BRT 2023	2985507860170167151	inventory

如果需要删除无连接订用，可以发送 HTTP DELETE 请求，包含要删除的订阅ID。假设您要删除 **subscription-id 2985507860170167151**:

```
curl --location --insecure --request DELETE 'https://10.122.28.3/restconf/data/v1/cisco-notifications:s
```

现在，如果再次查询数据库，则只能看到带有 SUBSCRIPTIONID 等于 7969974728822328535.

当库存发生变化时，客户端会打印通知（与以下内容类型相同） connection-oriented 在部分中看到的通知关于 connected-oriented 客户端)，然后是201响应：

```
(venv) [apinelli@centos8_cxlabs_spo app]$ python connectionless.py * Serving Flask app 'connectionless'
```

结论

在本文档中，两种基于API的通知可以在EPNM(connectionless和 connection-oriented)进行了说明，并给出了可用作模拟基础的相应客户端的示例。

相关信息

- https://www.cisco.com/c/dam/en/us/td/docs/net_mgmt/epn_manager/RESTConf/Cisco_Evolved_Programmable_Network_Manager_5_1_2_F
- 技术支持和文档 - Cisco Systems

关于此翻译
思科采用人工翻译与机器翻译相结合的方式将此文档翻译成不同语言，希望全球的用户都能通过各自的语言得到支持性的内容。

请注意：即使是最好的机器翻译，其准确度也不及专业翻译人员的水平。

Cisco Systems, Inc. 对于翻译的准确性不承担任何责任，并建议您总是参考英文原始文档（已提供链接）。