SNMP Traps - Understanding Trap Variables

Article Number: 558 | Rating: 2.3/5 from 3 votes | Last Updated: Mon, Oct 24, 2016 at 5:51 AM

Overview

This KB article explains how to identify the SNMP Trap variables that exist in traps and how they are used with SNMPTT.

It is assumed that you have already configured Nagios XI to receive SNMP Traps by following this guide:

How to Integrate SNMP Traps With Nagios XI

Send A Test v2c Trap

The first step is to send the Nagios XI server a test trap to itself.

Execute the following command:

```
snmptrap -v 2c -c public localhost '' 1.3.6.1.4.1.8072.2.3.0.1 1.3.6.1.4.1.8072.2.3.2.1 i 123456
```

Now execute this command to see the trap that was logged in the `/var/log/snmp/`**`snmpttunknown.log`** file:

```
tail /var/log/snmptt/snmpttunknown.log -n 20
```

The output will be something like:

```
Mon Oct 24 16:39:44 2016: Unknown trap (.1.3.6.1.4.1.8072.2.3.0.1) received from localhost at:
Value 0: localhost
Value 1: 127.0.0.1
Value 2: 0:2:09:15.23
Value 3: .1.3.6.1.4.1.8072.2.3.0.1
Value 4: 127.0.0.1
Value 5:
Value 6:
Value 7:
Value 8:
Value 9:
Value 10:
Ent Value 0: .1.3.6.1.4.1.8072.2.3.2.1=123456
```

Looking at the output you can most likely figure out what some of it means, but other information is not so easy to decipher. This is explained in the next section.

Understanding Variables

When a trap is received it comes with lots of data. SNMPTT accesses that data using variables and ultimately sends that data to Nagios XI. A example of an SNMPTT EVENT defined i follows:

```
EVENT netSnmpExampleHeartbeatRate .1.3.6.1.4.1.8072.2.3.0.1 "netSnmpExampleHeartbeatRate" Normal
FORMAT SNMP netSnmpExampleHeartbeatRate
EXEC /usr/local/bin/snmptraphandling.py "$r" "SNMP Traps" "$s" "$@" "" "netSnmpExampleHeartbeatRate"
```

An example of a variable is **`$r`** that you can see in the EXEC line. It equates to the **hostname** in the received trap *(assuming DNS resolution is working)*.

From the trap that was logged in the `/var/log/snmp/`**`snmpttunknown.log`** file, how do you know what the "hostname" field is? It's pretty obvious that it is the **`Value 0`** field but ho

The follow table shows the relationship:

snmpttunknown.log	Object	SNMPTT Variable
Value 0	Hostname	`$R $r`
Value 1	IP Address	`$aR $ar`
Value 2	Uptime	`$T`
Value 3	Trapname / OID	`$O $o`
Value 4	IP address from trap agent	`$aA`
Value 5	Trap community string	`$C`

Value 5	Trap community string	$C
Value 6	Enterprise	$E $e
Value 7	securityEngineID	$Be
Value 8	securityName	$Bu
Value 9	contextEngineID	$BE
Value 10	contextName	$Bn
Ent Value 0+	Passed variables	$1, $2, $n

From the trap that was logged in the /var/log/snmp/**snmpttunknown.log** file, you may have noticed that "Value 5" was empty but from the table above it should be populated wit after the following example *(enabling the embedded handler)*.

Send A Test v3 Trap

To receive SNMP v3 traps, additional configuration lines need to be added to the /etc/snmp/**snmptrapd.conf** file, examples of this can be found in the following KB article:

How To Send A Test Trap

Assuming SNMPTRAPD is correctly configured to receive SNMP v3 traps, the following command will send an SNMP v3 trap to the Nagios XI server itself:

```
snmptrap -v 3 -e 0x090807060504030201 -u the_user_name -a SHA -A the_SHA_string -x AES -X the_AES_string localhost '' 1.3.6.1.4.1.8072.
```

Now execute this command to see the trap that was logged in the /var/log/snmp/**snmpttunknown.log** file:

```
tail /var/log/snmptt/snmpttunknown.log -n 20
```

The output will be something like:

```
Mon Oct 24 19:46:34 2016: Unknown trap (.1.3.6.1.4.1.8072.2.3.0.1) received from localhost at:
Value 0: localhost
Value 1: 127.0.0.1
Value 2: 0:5:55:49.46
Value 3: .1.3.6.1.4.1.8072.2.3.0.1
Value 4: 127.0.0.1
Value 5:
Value 6:
Value 7:
Value 8:
Value 9:
Value 10:
Ent Value 0: .1.3.6.1.4.1.8072.2.3.2.1=123456
```

Looking at the output and looking at the variable table you would expect that **Values 7, 8, 9, 10** would have some SNMP v3 information. The reason why they are empty is that the SI expose the values.

To be able to access these values you need to enable the **embedded handler** for SNMPTRAPD. The steps involved in enabling the embedded handler are covered in the following I

SNMP Traps - Standard Handler vs Embedded Handler

To be very clear, the standard handler works for SNMP v3 traps, it's only required if you want to use the values 7 - 10.

Once you have configured SNMPTRAPD to use the embedded handlder and restarted the SNMPTRAPD service you can send another test trap, the output in the /var/log/snmp/**snm**

```
Mon Oct 24 19:46:51 2016: Unknown trap (.1.3.6.1.4.1.8072.2.3.0.1) received from localhost at:
Value 0: localhost
Value 1: 127.0.0.1
Value 2: (2136662) 5:56:06.62
Value 3: .1.3.6.1.4.1.8072.2.3.0.1
Value 4: 127.0.0.1
Value 5: unknown
Value 6:
Value 7: 0x908070605040302010
Value 8: the_user_name
Value 9: 0x0800f18808763675965302c08500000000
Value 10: unknown
Ent Value 0: .1.3.6.1.4.1.8072.2.3.2.1=123456
```

Here you can see that values 7 - 10 have data. It's also worth mentioning that **Value 5** *(Trap community string)* now shows **unknown**. Previously nothing was logged in here, but now sent an SNMP v3 trap there is no community string involved.

SNMP v2c Trap With Embedded Handler

Assuming that the embedded handler is still configured as per the last section, try sending an SNMP v2c trap again using the following command:

```
snmptrap -v 2c -c public localhost '' 1.3.6.1.4.1.8072.2.3.0.1 1.3.6.1.4.1.8072.2.3.2.1 i 123456
```

Now execute this command to see the trap that was logged in the `/var/log/snmp/`**`snmpttunknown.log`** file:

```
tail /var/log/snmptt/snmpttunknown.log -n 20
```

The output will be something like:

```
Mon Oct 24 19:59:11 2016: Unknown trap (.1.3.6.1.4.1.8072.2.3.0.1) received from localhost at:
Value 0: localhost
Value 1: 127.0.0.1
Value 2: (2210709) 6:08:27.09
Value 3: .1.3.6.1.4.1.8072.2.3.0.1
Value 4: 127.0.0.1
Value 5: public
Value 6:
Value 7: 0x57e6b6e6f677e6
Value 8: unknown
Value 9: 0x57e6b6e6f677e6
Value 10: unknown
Ent Value 0: .1.3.6.1.4.1.8072.2.3.2.1=123456
```

You can see now that **Value 5** *(Trap community string)* now shows the community string of public which was used in the last command. unknown. Previously nothing was logged in her the **embedded handler** is being used. If you wish to use the **$C** variable *(value 5)* then you will need to use the embedded handler.

The Enterprise Variables

In the previous examples, when sending the test traps you will have noticed in the `/var/log/snmp/`**`snmpttunknown.log`** file the last line of output:

```
Ent Value 0: .1.3.6.1.4.1.8072.2.3.2.1=123456
```

This is the data that you send along with the test trap, specifically this bold section in the command:

```
snmptrap -v 2c -c public localhost '' 1.3.6.1.4.1.8072.2.3.0.1 1.3.6.1.4.1.8072.2.3.2.1 i 123456
```

The data is coming in on the OID of **`1.3.6.1.4.1.8072.2.3.2.1`** and it is an **integer** with the value **`123456`**.

`Ent Value 0` is the variable **`$1`** in SNMPTT.

If a second Ent Value was received (**`Ent Value 1`**), it is the variable **`$2`** in SNMPTT.

For example, here is a test trap sending two Ent Values:

```
snmptrap -v 2c -c public localhost '' 1.3.6.1.4.1.8072.2.3.0.1 1.3.6.1.4.1.8072.2.3.2.1 i 123456 1.3.6.1.4.1.8072.2.3.2.2 s "My Test St
```

Now you'll see this in the `/var/log/snmp/`**`snmpttunknown.log`** file:

```
Mon Oct 24 20:26:05 2016: Unknown trap (.1.3.6.1.4.1.8072.2.3.0.1) received from localhost at:
Value 0: localhost
Value 1: 127.0.0.1
Value 2: (2372091) 6:35:20.91
Value 3: .1.3.6.1.4.1.8072.2.3.0.1
Value 4: 127.0.0.1
Value 5: public
Value 6:
Value 7: 0x57e6b6e6f677e6
Value 8: unknown
Value 9: 0x57e6b6e6f677e6
Value 10: unknown
Ent Value 0: .1.3.6.1.4.1.8072.2.3.2.1=123456
Ent Value 1: .1.3.6.1.4.1.8072.2.3.2.2=My Test String
```

Being able to access the Ent Values allows you to build complex SNMPTT configurations and send the right data to Nagios XI.

Example SNMPTT EVENT

Example SNMPTT EVENT

Here is an example event defined in `/etc/snmp/`**`snmptt.conf`** :

```
EVENT netSnmpExampleHeartbeatRate .1.3.6.1.4.1.8072.2.3.0.1 "netSnmpExampleHeartbeatRate" Normal
FORMAT A trap from $r ($aA) had the data 1 = "$1" and 2 = "$2"
EXEC /usr/local/bin/snmptraphandling.py "$r" "SNMP Traps" "$s" "$@" "" "The integer is $1 and the string is $2"
```

Once an EVENT exists in `/etc/snmp/`**`snmptt.conf`**, any incoming trap for that OID will no longer be logged in `/var/log/snmp/`**`snmpttunknown.log`** file, it will now be logged in th

Now send a test trap that matches that OID with two Ent Values:

```
snmptrap -v 2c -c public localhost '' 1.3.6.1.4.1.8072.2.3.0.1 1.3.6.1.4.1.8072.2.3.2.1 i 123456 1.3.6.1.4.1.8072.2.3.2.2 s "My Test St
```

You'll see this in the `/var/log/snmp/`**`snmptt.log`** file:

```
Mon Oct 24 20:43:07 2016 .1.3.6.1.4.1.8072.2.3.0.1 Normal "netSnmpExampleHeartbeatRate" localhost - A trap from localhost (127.0.0.1) h
```

The data you see in the `/var/log/snmp/`**`snmptt.log`** file correlates to the **`FORMAT`** line above.

In Nagios XI, for that SNMP Traps service you should see the following output:

```
The integer is 123456 and the string is My Test String
```

Resources

The SNMPTT documentation was the basis for this KB article and is a very handy for understanding how SNMPTT works.

SNMP Trap Translator

Final Thoughts

For any support related questions please visit the Nagios Support Forums at:

http://support.nagios.com/forum/

Posted by: **tlea** - Mon, Oct 24, 2016 at 1:45 AM. This article has been viewed 11963 times.

Online URL: https://support.nagios.com/kb/article/snmp-traps-understanding-trap-variables-558.html