Snmptrap Nodebrain Module

Release 0.9.02

Snmptrap NodeBrain Module
August 2014
NodeBrain Open Source Project

Release 0.9.02

Author: Ed Trettevik

History

2005-10-12 Title: *NodeBrain Tutorial*
Author: Ed Trettevik <eat@nodebrain.org>
Publisher: NodeBrain Open Source Project

2010-12-31 Release 0.8.3

- Updates - still needed

Preface

This manual is intended for users of the Snmptrap NodeBrain Module, a plug-in for statistical anomaly detection. The Baseline module was first introduced in NodeBrain 0.8.3 in September 2010. This module should be treated as a prototype. It has not yet been exercised enough to fully validate the design. We expect it to evolve as we gain experience.

This tutorial is intended for readers seeking an introduction to NodeBrain through a series of simple examples. Other documents are available for readers looking for a more complete reference to the rule language, modules, or API (application programmatic interface).

The intent of the examples in this tutorial is to illustrate individual concepts, not to provide complete working applications or show all related options. We avoid formal syntax descriptions, thinking you are here because you want to figure it out from examples.

Files referenced in this tutorial are included in the tutorial directory of the NodeBrain distribution.

See www.nodebrain.org for more information and the latest update to this document.

Documents

NodeBrain Guide - Information on using `nb`
NodeBrain Tutorial - A gentle introduction to `nb` and the rule language
NodeBrain Language - Rule language syntax and semantics
NodeBrain Library - C API

Document Conventions

Sample code and input/output examples are displayed in a monospace font, indented in HTML and Info, and enclosed in a box in PDF or printed copies. Bold text is used to bring the reader's attention to specific portions of an example. In the following example, the first and last line are associated with the host shell and the lines in between are input or output unique to NodeBrain. The `define` command is highlighted, indicating it is the focus of the example. Lines ending with a backslash \ indicate when a command is continued on the next displayed line. This is supported by the language within source files, but not for other methods of command input. If you copy an example of a command displayed over multiple lines, you must enter it as a single line when used outside the context of a source file.

```
$ nb
> define myFirstRule on(a=1 and b=2) mood="happy";
> assert mood="sad";
> show mood
mood = "sad"
> assert a=1,b=2,c=3,d="This is an example of a long single line that",\
    e="we depict on multiple lines to fit on the documnet page";
2008/06/05 12:09:08 NB000I Rule myFirstRule fired(mood="happy")
> show mood
mood = "happy"
> quit
$
```

Table of Contents

1 Concepts

The Snmptrap module provides a node that monitors SNMP V1 and V2 traps. This node listens on a specified port and interface for SNMP traps. By default, it listens to port 162 on all interfaces.

```
define snmptrap node snmptrap; # default to port 162
define snmptrap node snmptrap(50162); # alternate port
define snmptrap node snmptrap("127.0.0.1"); # interface address
define snmptrap node snmptrap("127.0.0.1:50162"); # both
```

When a trap is received, the node sends an alert command to its own context. Since this module is not MIB aware, the generated alert command references NodeBrain terms that are single quoted OIDs.

```
alert ’<oid>’=<value>,...;
```

You can reference these OID terms in you NodeBrain rules or define aliases as illustrated in the example below.

```
define snmptrap node snmptrap;
snmptrap. define snmpTrap      cell ’1.3.6.1.6.3.1.1.4.1.0’;
snmptrap. define myMsgText     cell ’1.3.6.1.4.1.2789.2005.1’;
snmptrap. define myRestartTrap cell snmpTrap="1.3.6.1.4.1.2789.2005.0.2476317";
snmptrap. define r1 if(myRestartTrap and myMsgText~~"WWW"):$ - echo "$${myMsgText}"
```

To send your own traps to this node, you will need a utility for sending traps. Examples are shown below using the snmptrap in the Net-SNMP package.

```
snmptrap -v 1 -d -c public localhost .1.3.6.1.4.1.2789.2005 localhost 6 2476317 ’’ \
          .1.3.6.1.4.1.2789.2005.1 s "WWW Server Has Been Restarted"

snmptrap -v 2c -d -c public localhost ’’ .1.3.6.1.6.3.1.1.5.3 \
          ifIndex i 2   ifAdminStatus i 1    ifOperStatus i 1
```

2 Tutorial

> *Man is the only kind of varmint who sets his own trap, baits it, then steps on it.* —John Steinbeck (1902–1968)

The Snmptrap node is used to monitor SNMP traps. This is one method of configuring NodeBrain to accept alerts from monitoring tools that are capable of sending SNMP traps. NodeBrain's Snmptrap node is a bit unusual in that it does not use MIBs. Instead, each trap is converted into a NodeBrain alert using single quoted OID terms.

```
alert 'oid'="value",'oid'="value",...;
```

You must then code your rules referencing the *oid* terms. However, I recommend you define aliases for the OIDs of interest to make your rules more readable.

```
#!/usr/local/bin/nb -d
# File: tutorial/Snmptrap/snmptrap.nb
-rm snmptrap.log
set log="snmptrap.log",out=".";
# Node
define snmptrap node snmptrap:trace,dump;
# Aliases
snmptrap. define myProduct cell '1.3.6.1.6.3.1.1.4.3'="1.3.6.1.4.1.1279";
snmptrap. define address cell '1.3.6.1.4.1.1279.4';
snmptrap. define type    cell '1.3.6.1.4.1.1279.5';
# Rules
snmptrap. define r1 if(myProduct and type="hiccup");
```

The example above is only provided to illustrate the syntax for working with single quoted OID terms. You will need to adapt this example to the traps you want to monitor to construct useful rules. However, you can use this example to start collecting traps right away. The traps will show up in your `snmptrap.log` file. Then, you can figure out what you want to monitor.

Remove the `:trace,dump` from your Snmptrap node specification to reduce the amount of information in your log.

```
define snmptrap node snmptrap;
```

You may use a `silent` option to stop logging the alerts generated by the Snmptrap node.

```
define snmptrap node snmptrap:silent;
```

3 Commands

3.1 Define

4 Triggers

This module generates an `alert` for every trap received.

Index