Probus Tech

Discovery Rule

Checks:

SNMPv2 agent "1.3.6.1.2.1.1.1.0"

SNMPv2 agent "1.3.6.1.2.1.1.5.0"

SNMPv2 agent "1.3.6.1.2.1.2.2.1.2,1.3.6.1.2.1.2.2.1.3"

SNMPv2 agent "1.3.6.1.2.1.47.1.1.1.1.2.1"

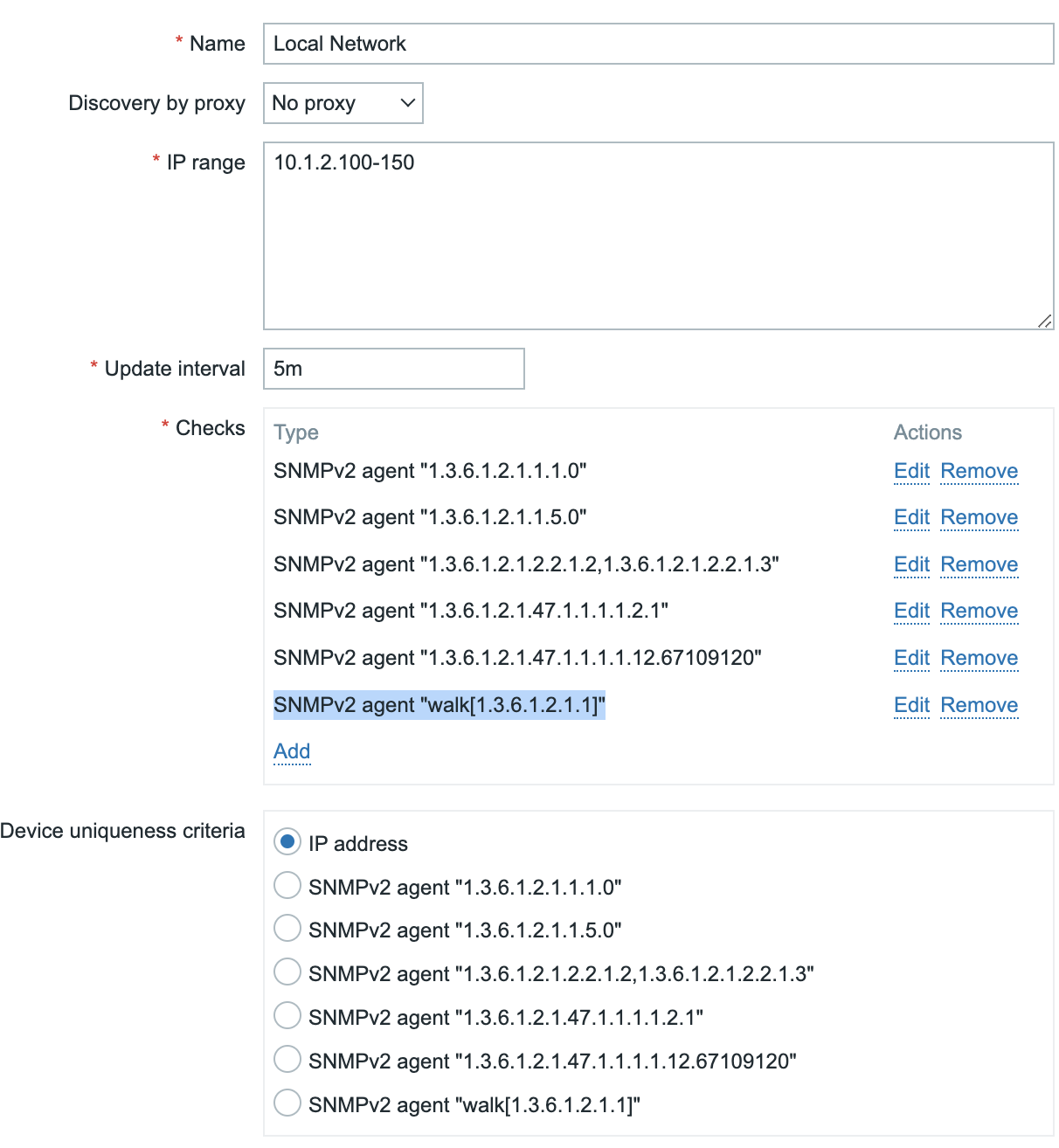
SNMPv2 agent "1.3.6.1.2.1.47.1.1.1.1.12.67109120"

SNMPv2 agent "walk[1.3.6.1.2.1.1]"

Device uniqueness criteria: IP address

Host name: DNS Name

Visible name: HostName



Trigger Actions:

A screenshot of a computer

Description automatically generated

**Discovery actions**

Add Cisco: A screenshot of a computer

Description automatically generated

A screenshot of a web page

Description automatically generated

Add IP Cameras:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Scripts**

Detect Operating System

A screenshot of a computer

Description automatically generated

Ping

A screenshot of a computer

Description automatically generated

Traceroute

A screenshot of a computer

Description automatically generated

**Is it possible to create network map automatically ?**

<https://www.zabbix.com/documentation/current/en/manual/api/reference/map/create>

**Examples**

**Create an empty map**

Create a map with no elements.

[Request](https://www.zabbix.com/documentation/current/en/manual/api#performing-requests):

{ "jsonrpc": "2.0", "method": "map.create", "params": { "name": "Map", "width": 600, "height": 600 }, "id": 1}

Response:

{ "jsonrpc": "2.0", "result": { "sysmapids": [ "8" ] }, "id": 1}

**Create a host map**

Create a map with two host elements and a link between them. Note the use of temporary "selementid1" and "selementid2" values in the map link object to refer to map elements.

[Request](https://www.zabbix.com/documentation/current/en/manual/api#performing-requests):

{ "jsonrpc": "2.0", "method": "map.create", "params": { "name": "Host map", "width": 600, "height": 600, "selements": [ { "selementid": "1", "elements": [ {"hostid": "1033"} ], "elementtype": 0, "iconid\_off": "2" }, { "selementid": "2", "elements": [ {"hostid": "1037"} ], "elementtype": 0, "iconid\_off": "2" } ], "links": [ { "selementid1": "1", "selementid2": "2" } ] }, "id": 1}

Response:

{ "jsonrpc": "2.0", "result": { "sysmapids": [ "9" ] }, "id": 1}

**Create a trigger map**

Create a map with trigger element, which contains two triggers.

[Request](https://www.zabbix.com/documentation/current/en/manual/api#performing-requests):

{ "jsonrpc": "2.0", "method": "map.create", "params": { "name": "Trigger map", "width": 600, "height": 600, "selements": [ { "elements": [ {"triggerid": "12345"}, {"triggerid": "67890"} ], "elementtype": 2, "iconid\_off": "2" } ] }, "id": 1}

Response:

{ "jsonrpc": "2.0", "result": { "sysmapids": [ "10" ] }, "id": 1}

**Map sharing**

Create a map with two types of sharing (user and user group).

[Request](https://www.zabbix.com/documentation/current/en/manual/api#performing-requests):

{ "jsonrpc": "2.0", "method": "map.create", "params": { "name": "Map sharing", "width": 600, "height": 600, "users": [ { "userid": "4", "permission": "3" } ], "userGroups": [ { "usrgrpid": "7", "permission": "2" } ] }, "id": 1}

Response:

{ "jsonrpc": "2.0", "result": { "sysmapids": [ "9" ] }, "id": 1}

**Map shapes**

Create a map with map name title.

[Request](https://www.zabbix.com/documentation/current/en/manual/api#performing-requests):

{ "jsonrpc": "2.0", "method": "map.create", "params": { "name": "Host map", "width": 600, "height": 600, "shapes": [ { "type": 0, "x": 0, "y": 0, "width": 600, "height": 11, "text": "{MAP.NAME}" } ] }, "id": 1}

Response:

{ "jsonrpc": "2.0", "result": { "sysmapids": [ "10" ] }, "id": 1}

**Map lines**

Create a map line.

[Request](https://www.zabbix.com/documentation/current/en/manual/api#performing-requests):

{ "jsonrpc": "2.0", "method": "map.create", "params": { "name": "Map API lines", "width": 500, "height": 500, "lines": [ { "x1": 30, "y1": 10, "x2": 100, "y2": 50, "line\_type": 1, "line\_width": 10, "line\_color": "009900" } ] }, "id": 1}

Response:

{ "jsonrpc": "2.0", "result": { "sysmapids": [ "11" ] }, "id": 1}

<https://www.zabbix.com/documentation/4.2/en/manual/config/visualisation/maps/map>

<https://www.zabbix.com/forum/zabbix-help/399587-is-it-possible-to-create-network-map-automatically>

Searching for "cdp lldp zabbix map" (without the quotes) since CDP and LLDP are the main protocols that should be useful in mapping networks.  
  
[https://www.reddit.com/r/zabbix/comm...mment/gle2lhl/](https://www.reddit.com/r/zabbix/comments/l825y0/comment/gle2lhl/)  
[https://assets.zabbix.com/files/zabs...\_in\_Zabbix.pdf](https://assets.zabbix.com/files/zabsummit2018/Takeshi_Tanaka-Monitoring_and_Visualization_of_LLDP_information_in_Zabbix.pdf)  
<https://github.com/Uninett/nav>  
<https://en.wikipedia.org/wiki/Netdisco>

<https://www.zabbix.com/integrations>

[**zabbix-map**](https://github.com/TiggyWiggler/zabbix-map)

<https://github.com/TiggyWiggler/zabbix-map>

<https://github.com/TiggyWiggler/zabbix-map/wiki/Command-Line-Parameters>

Auto discovery of network topology - Forum

if multiple addresses can be defined per host, the with the help of auto discovery feature, zabbix could discover network topology in a matter of one hour, at least in my case.  
here is how I do it (how i describe below works for nagios, but its not much work to change it to work in zabbix)::  
1) Host list(s) is generated using autodiscovery feature, just IP address enough, hostname or alias a plus  
2) Sample trace script to determine parents, 99% accurate if you choose carefully data to run script on. Sure, its faster to write the whole thing into one C program, it takes me about 70 minutes to map about 650 hosts.

3) Map automatically generated, most probably a mess, that confirms to no specific topology really, just how i like it, spaghetti style :P

So the only step missing is to have zabbix allow multiple address definitions per host, its only logical, i dont understand why its not done this way.

**Tags:** None

Code

#!/usr/bin/perl -w

# (C) Virtual ISP S.A.L.

# Denys Fedoryshchenko

my $debug = 1;

use strict;

my $x;

my @hops;

my @hosts;

my $hop;

my $unknown;

my $lasthost;

my %db;

my $filename = shift(@ARGV);

if (!$filename) {

print "ERROR!\n".$0." hosts.cfg\n";

exit;

}

sub traceroute {

my $dsthost = shift(@\_);

my $host;

undef @hops;

print("Tracing host: $dsthost\n");

my @hops = `/usr/sbin/traceroute -w 2 -n -I $dsthost 2>/dev/null`;

$hop = pop(@hops); #Ignore it is last hop

print($hop." last hop, ignoring\n") if ($debug == 1) ;

$hop = pop(@hops);

if (!$hop) { return("1HOPAWAY"); }

print($hop." this hop is parent\n") if ($debug == 1) ;

#WORKAROUND ON PACKETLOSS (asestriks)

($unknown,$host) = split(' ',$hop);

($unknown,$unknown,$host) = split(' ',$hop) if ($host eq '\*') ;

($unknown,$unknown,$unknown,$host) = split(' ',$hop) if ($host eq '\*') ;

if ($host eq '\*') { return("UNKNOWN"); }

return($host);

}

open(FILE2,'>'.$filename.".new");

open(FILE,'<'.$filename);

print(FILE2 "# host\_name(s) \n#");

while (<FILE>) {

if (/host\_name\s+([A-Z0-9\-\.]+)\s+/i) {

$lasthost = $1;

# print (FILE2 $lasthost.",");

}

if (/address\s+([0-9\.\,]+)\s+$/) {

@hosts = split(/\,/,$1);

while($x = shift(@hosts)) {

print("Adding ".$x." as ".$lasthost."\n") if ($debug == 1);

$db{$x} = $lasthost;

}

}

}

close(FILE);

print "\n\n";

open(FILE,'<'.$filename);

while (<FILE>) {

print FILE2 $\_;

if (/address\s+([0-9]+\.[0-9]+\.[0-9]+\.[0-9]+)/) {

$hop = traceroute($1);

if ($db{$hop}) {

print("Found! $db{$hop}\n");

print(FILE2 "\tparent\t\t".$db{$hop}."\n");

} elsif ($hop ne '1HOPAWAY' && $hop ne 'UNKNOWN') {

print("Not found IP in list - $hop !!! You must have this host in hosts.cfg so program can get name of it and set as parent\n");

} elsif ($hop eq 'UNKNOWN') {

print(FILE2 "#PARENT-NOT-DETECTED\n");

}

}

}

close(FILE);

close(FILE2);