Shinken SNMP Booster Module Documentation

Release 1.0

Thibault Cohen

Jun 16, 2017

Contents

Contents:

CHAPTER 1

SNMP Booster: How does it works

Overview

What is it

The SnmpBooster module allows Shinken Pollers to directly manage SNMP data acquisition. This is an all Python cross-platform SNMP module. It is tightly integrated with the Shinken Poller, Scheduler and Arbiter daemons to provide the best possible user experience.

Why use it

The SnmpBooster module is designed to be efficient and scalable. It has a flexible configuration method to make it easy to use with Shinken Monitoring Packs and the Shinken configuration generator genDevConfig.

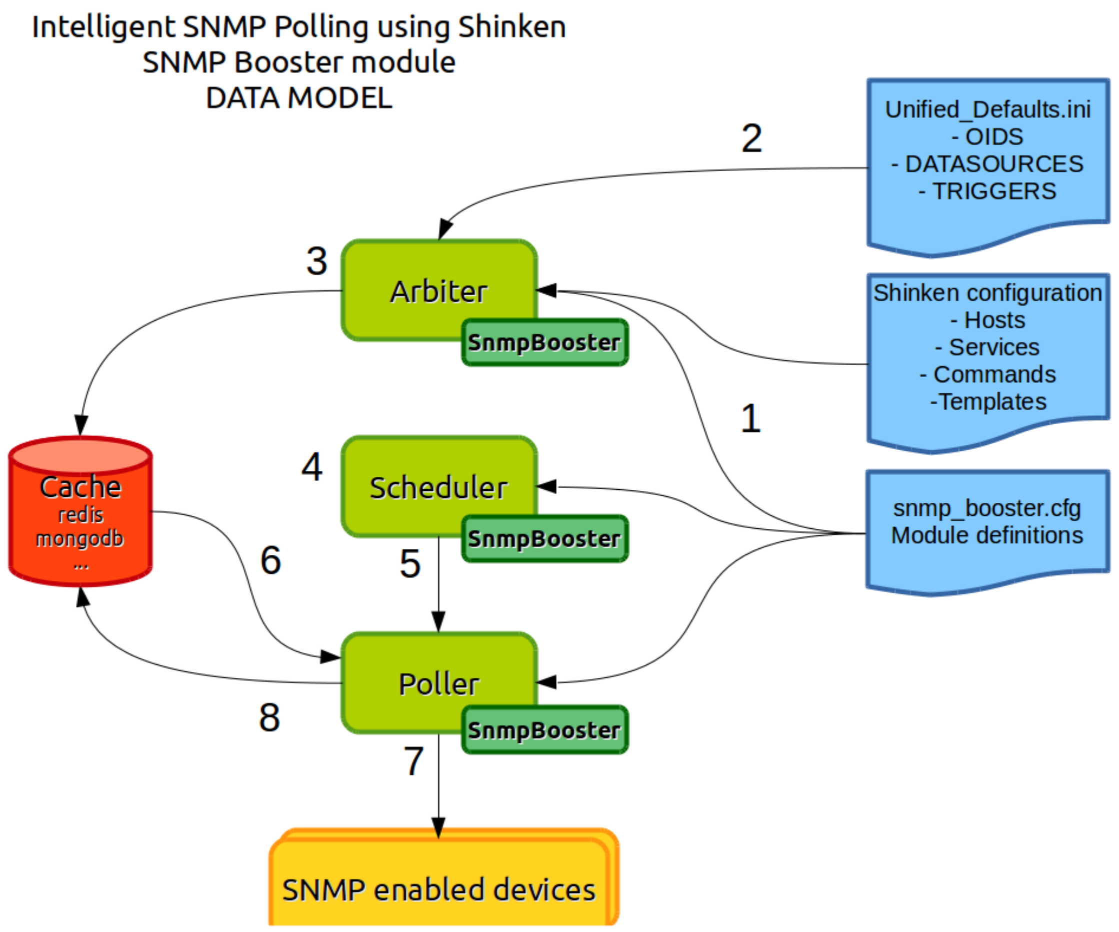
This acquisition module was professionally designed and developed.

It is meant to be used by the discovery engine genDevConfig (**v3.0.5 and newer**) originally developed for the Cricket SNMP monitoring tool and converted for use with Shinken.

It is the only scalable SNMP v2c implementation for Shinken.

How does it work

Shinken Integration



- 1 - The SnmpBooster Arbiter module reads the Shinken SnmpBooster configuration file(s). It reads the check commands and based on the values in the check commands that use the snmp_poller module it will creates a shared configuration cache using Redis. This permits to tie together Shinken Hosts and Services with the SNMP specific configuration. The Scheduler daemon schedules Host and Service checks as it normally does.
- 2 - The SnmpBooster Arbiter module computes Shinken configuration with datasource files (.ini files) and prepare datas for Redis
- 3 - The SnmpBooster Arbiter module stores an entry in Redis for each service defined in Shinken configuration
- 4 - The SnmpBooster Scheduler module determines which services will launch a SNMP requests and which will be a Redis requests
- 5 - Scheduler give tasks to pollers
- 6 - The SnmpBooster Poller module gets datas from Redis:
  - It get the current service if it's a Redis request
  - It get all services from the host of the current service if it's a SNMP request

- 7 - The SnmpBooster Poller module makes SNMP requests
- 8 - The SnmpBooster Poller module computes and stores collected datas from SNMP in Redis

Performance

SnmpBooster uses SNMP v2c getbulk or snmpgetnext for high efficiency. GetBulk and snmp get-next use a single request PDU to ask for multiple OIDs or even entire tables, instead of sending one request PDU per OID.

For example: *A typical 24 port network switch with two uplinks might use 375 OIDS (8 OIDs per interface, plus general OIDs for chassis health, memory, cpu, fans, etc.). SnmpBooster will only require around 4 request PDUs instead of 375 request PDUs. Each PDU is a request packet which takes time to create, send get processed and return. More timeouts to manage, more connections, more impact on the remote device and more latency means much fewer checks per second.*

SNMP Requests are multithreaded and each thread is responsible for gathering all data associated with a device.

SNMP Booster respectes all Shinken processing like retries, downtimes, etc for each monitored service.

A Redis data store is used to store collected data. All services associated to the same device will get their data from the Redis store. When data needs to be refreshed a single thread is chosen to contact the actual device to update the cache.

The SnmpBooster module supports automatic instance mapping for OIDs as well as static instances. (Ex. Based on the interface name it will figure out that the SNMP index(or instance) is 136. This is automatically handled by genDevConfig and SnmpBooster, no user input required. :-)

The generic SNMP configuration information is stored in the Shinken SnmpBooster INI files. There is a Defaults_unified.ini and a series of other Defaults files, one per discovery plugin for genDevConfig.

Important: genDevConfig plugins have all been converted to use the new dynamic instance mapping methods. You are now free to use most if not all Defaults*.ini files included with genDevConfig. 2012-10-28

Limitations

You should have your pollers with SnmpBooster in the same datacenter, as they need to be on the same machine with good connectivity to the active Redis server.

SnmpBooster is not compatible with distributed pollers in multiple datacenters, sorry, the current design of SnmpBooster uses a single centralized Redis instance for storing the timeseries data. For distributed datacenters to be supported, each poller+scheduler+Redis must be realm restrained, which is not the case today.

Design specification

SnmpBooster design specification and current development status.

Data model

The information required to define the data is split in two locations.

The first location is the host and service Shinken configuration (You need to generate or write this)

- Device specific information * IP addresses, host_names, device types, instance keys * A DSTEMPLATE must be referred to in the Service definition * A static SNMP instance could be referred to in the Service definition * An SNMP instance MAP function could be referred to in the Service definition * A TRIGGERGROUP could be refered to in the Service definition * A DS max could be refered to in the Service definition

The second location is SNMP Defaults.* templates. (Here you can create new devices or add new data sources)

- DATASOURCE information * SNMP OID * Type of data and how can it be interpreted (GAUGE, COUNTER, COUNTER64, DERIVE, DERIVE64, TEXT, TIMETICK) * Data format preparation (Scaling the data for example bits to bytes) * Is there an instance to append to the
- Instance MAP function * Mapping the instance dynamically using a function * Data or rules related to the mapping function
- DSTEMPLATEs to associate DATASOURCE to actual device classes * List of DATASOURCES associated with a, for example, Cisco 1900 router. Which in turn can be applied to a Shinken service
- TRIGGER and TRIGGERGROUPS to apply thresholding rules * Define triggers and associate them with a TRIGGERGROUP name that can be applied to a Shinken Service

A final location contains rules to build your Shinken configuration.

- genDevConfig plugins create Shinken configurations

Installation and configuration

SnmpBooster installation

Reference Dictionnary

SnmpBooster reference dictionary

Troubleshooting

SnmpBooster troubleshooting

CHAPTER 2

SNMP Booster: Install and setup

SnmpBooster Download Install and Configure

- *What is the SnmpBooster module*
- *Install and configure the SNMP acquisition module* [You are here]
- *SnmpBooster troubleshooting*
- *SnmpBooster design specification*
- *SnmpBooster configuration dictionnary*

Downloads

The SnmpBooster module and genDevConfig are currently in public beta prior to integration within Shinken. You can consult t

- https://github.com/xkilian/genDevConfig
- https://github.com/savoirfairelinux/mod-booster-snmp (use for_shinken_1.4 branch)
  - Download and copy mod-booster-snmp/shinken/modules/snmp_booster to shinken/modules/

Requirements

The SnmpBooster module requires:

- Python 2.6+
- Shinken 1.2+ < 2.0
- PySNMP 4.2.1+ (Python module and its dependencies)

- ConfigObj (Python module)
- python-redis >= 2.7.2
- Redis package for your operating system (ex. For Ubuntu: apt-get install redis-server)

The genDevConfig profile generator depends on:

- Perl 5.004+
- 4 perl modules available from CPAN and network repositories. genDevConfig/INSTALL has the installation details.

STRONGLY RECOMMENDED: Use the same version of Python and Pyro on all hosts running Shinken processes.

Installation

SnmpBooster:

- Install the dependencies
- Copy the snmp_booster directory from the git repository to your shinken/modules directory.
- Configuration steps are listed in the present web page.

genDevConfig:

- Download and extract the archive to your server.
- See genDevConfig/INSTALL on how to install and configure it.

Configuration

How to define the SnmpBooster module in the Shinken daemons

You need to modify shinken-specific.cfg, which is located in *shinken/etc/shinken-specific.cfg*

Arbiter daemon configuration

Simply declare the module inside arbiter definition:

```
modules SnmpBoosterArbiter
```

Scheduler daemon configuration

Simply declare the module inside scheduler definition:

```
modules SnmpBoosterScheduler
```

Poller daemon configuration

Simply declare the module inside poller definition:

```
modules SnmpBoosterPoller
```

SnmpBooster Module declaration

You have to declare all least 3 modules.

One for the Arbiter:

```
define module {
    module_name          SnmpBoosterArbiter
    module_type          snmp_booster
    datasource           /etc/shinken/snmpbooster_datasource/   ; SET THE DIRECTORY␣
↪FOR YOUR Defaults*.ini FILES provided by genDevConfig
    db_host              192.168.1.2   ; SET THE IP ADDRESS OF YOUR redis SERVER
    loaded_by            arbiter
}
```

One for the Scheduler:

```
define module {
    module_name          SnmpBoosterScheduler
    module_type          snmp_booster
    loaded_by            scheduler
}
```

One for the Poller:

```
define module {
    module_name          SnmpBoosterPoller
    module_type          snmp_booster
    loaded_by            poller
    db_host              192.168.1.2

}
```

If you do not know the IP adress on which your Redis is listening, check under /etc/redis/redis.con. Or do a:

```
netstat -a | grep redis
```

If you are running a test on the local machine you can leave redis on 127.0.0.1 (localhost), but if your poller, scheduler or arbiter is on a different machine, set the redis to listen on a real IP address.

Parameters

module_name Module Name. Example: *SnmpBoosterPoller*

module_type Module type. Must be: *snmp_booster*

datasource Datasource folder. Where all your Defaults*.ini are. Example: */etc/shinken/snmpbooster_datasource/*

db_host Memcached host IP. Default: *127.0.0.1*. Example: *192.168.1.2*

db_port Memcached host port. Default: *27017*. Example: *27017*

loaded_by Which part of Shinken load this module. Must be: *poller*, *arbiter* or *scheduler*. Example: *arbiter*

How to define a Host and Service

Step 1

Create a template for your SNMP enabled devices.

Sample template:

```
cd shinken/etc/packs/network/
mkdir SnmpBooster

vi shinken/etc/packs/network/SnmpBooster/templates.cfg
```

To edit the file

```
define command {
  command_name    check_snmp_booster
  command_line    check_snmp_booster -H $HOSTNAME$ -A $HOSTADDRESS$ -S '$SERVICEDESC$
↪' -C $_HOSTSNMPCOMMUNITYREAD$ -V $_HOSTSNMPCOMMUNITYVERSION$ -t $_SERVICEDSTEMPLATE
↪$ -i $_SERVICEINST$ -n '$_SERVICEINSTNAME$' -T $_SERVICETRIGGERGROUP$ -N $_
↪SERVICEMAPPING$ -b $_HOSTUSEBULK$ -c $_HOSTNOCONCURRENCY$ -d $_
↪SERVICEMAXIMISEDATASOURCE$ -v $_SERVICEMAXIMISEDATASOURCEVALUE$
  module_type     snmp_booster
}

define command {
  command_name    check_snmp_booster_bulk
  command_line    check_snmp_booster -H $HOSTNAME$ -A $HOSTADDRESS$ -S '$SERVICEDESC$
↪' -C $_HOSTSNMPCOMMUNITYREAD$ -V $_HOSTSNMPCOMMUNITYVERSION$ -t $_SERVICEDSTEMPLATE
↪$ -i $_SERVICEINST$ -n '$_SERVICEINSTNAME$' -T $_SERVICETRIGGERGROUP$ -N $_
↪SERVICEMAPPING$ -b 1 -d $_SERVICEMAXIMISEDATASOURCE$ -v $_
↪SERVICEMAXIMISEDATASOURCEVALUE$
  module_type     snmp_booster
}
```

Parameters for check_snmp_booster command

-H, --host-name	server hostname; (**mandatory**)
-A, --host-address	server address; (**mandatory**)
-S, --service	service description; (**mandatory**)
-C, --community	SNMP community; Default: *public*
-P, --port	SNMP port; Default: *161*
-V, --snmp-version	SNMP version; Default: *2c*
-s, --timeout	SNMP request timeout; Default: *5* (seconds)
-e, --retry	SNMP request retry; Default: *1*
-t, --dstemplate	dstemplate name; Example: *standard-interface*; (**mandatory**)

-i, --instance	instance (no mapping need); Example: *1.32.4*
-n, --instance-name	instance name use for mapping; Example: *Intel_Corporation_82579LM_Gigabit_Network_Connection*
-m, --mapping	OID used to do the mapping; Example: *.1.3.6.1.2.1.2.2.1.2*
-N, --mapping-name	name of the OID used to do the mapping; Example: *interface-name*
-T, --triggergroup	name of the trigger group which contains several triggers; Example: *interface-hc*
-b, --use-getbulk	use snmp getbulk requests to do the mapping; Default: *0*
-M, --max-rep-map	max_repetition parameters for snmp getbulk requests; Default: *64*
-g, --request-group-size	max number of asked oids in one SNMP request; Default: *64*
-c, --no-concurrency	Disable concurrent SNMP requests on the same host; Default: *0*
-d, --maximise-datasources	List of datasources you want to set a maximal value for. Each datasources are separated by a comma; Example: *confAvailable,confBusy*
-v, --maximise-datasources-value	List of maximal values for datasources defined with -d options. Each values are separated by a comma and are associated with the datasource in the same position; Example: *2,8*

Template definitions

```
define host{
  name                     SnmpBooster-host
  alias                    SnmpBooster-host template
  check_command            check_host_alive
  max_check_attempts       3
  check_interval           1
  retry_interval           1
  use                      generic-host
  register                 0
  _snmpcommunityread       $SNMPCOMMUNITYREAD$
  _snmpcommunityversion    $SNMPCOMMUNITYVERSION$
  _usebulk                 0
  _noconcurrency           0
}



define service {
  name                     default-snmp-template
  check_command            check_snmp_booster
  _inst                    None
  _triggergroup            None
  _mapping                 None
  _maximisedatasource      ''
  _maximisedatasourcevalue ''
  max_check_attempts       3
  check_interval           1
  retry_interval           1
  register                 0
}
```

Step 2

Define some hosts and services. You would typically use genDevConfig or another configuration generator to create these for you.

Mandatory host arguments related to SNMP polling:

```
_snmpcommunityread    public                   ; which could be set in your resource.cfg␣
↪file
_snmpversion          public                   ; which could be set in your resource.cfg␣
↪file
_usebulk              0                        ; use bulk request to do mapping
_noconcurrency        0                        ; SNMPBooster can make multiple requests␣
↪on the same host at the same time
```

Mandatory service arguments related to SNMP polling:

```
_dstemplate                Cisco-Generic-Router  ; Name of a DSTEMPLATE defined in the␣
↪SnmpBooster config.ini file
```

Optional service arguments related to SNMP polling with default values:

```
_inst                    None   ; Could be numeric: 0, 0.0.1, None
_instname                None   ; Instance name use to do mapping
_triggergroup            None   ; Name of the triggergroup defined in the SnmpBooster␣
↪config.ini file to use for setting warning and critical thresholds
_mapping                 None   ; Mapping name defined in [MAP] section in the␣
↪SnmpBooster ini files
```

Here an example how to configure a service to use instance mapping

```
_instname                FastEthernet0_1
_mapping                 interface-name
```

Sample Shinken host and service configuration:

```
# Generated by genDevConfig 3.0.0
# Args: --showunused -c publicstring 192.168.2.63
# Date: Thu Aug 30 17:47:59 2012

#######################################################################
# Description: Cisco IOS Software, C2960 Software (C2960-LANBASEK9-M), Version 12.
↪2(50)SE4, RELEASE SOFTWARE (fc1) Technical Support: http://www.cisco.com/
↪techsupport Copyright (c) 1986-2010 by Cisco Systems, Inc. Compiled Fri 26-Mar-10␣
↪09:14 by prod_rel_team
#     Contact:
# System Name: SITE1-ASW-Lab04
#    Location:
#######################################################################

define host {
   host_name          192.168.2.63
   display_name               192.168.2.63
   _sys_location
   address            192.168.2.63
   hostgroups
   notes
   parents
   use                        default-snmp-host-template
   register           1
}

define service {
   host_name          192.168.2.63
   service_description        chassis
   display_name               C2960 class chassis
   _dstemplate                Cisco-Generic-Router
   _inst              0
   use                        default-snmp-template
   register           1
}

define service {
   host_name          192.168.2.63
   service_description        chassis.device-traffic
   display_name               Switch fabric statistics - Packets per Second
   _dstemplate                Device-Traffic
   use                        default-snmp-template
   register           1
}

define service {
   host_name          192.168.2.63
   service_description        if.FastEthernet0_1
   display_name               FastEthernet0_1 Description: Link to Router-1 100.0␣
↪MBits/s ethernetCsmacd
   _dstemplate                standard-interface
   _instname          FastEthernet0_1
   _mapping           interface-name
   use                        default-snmp-template
   register           1
}
```

Here is an example configuration of the config.ini file

```
[DATASOURCE]
    OidmyOidDefinition = .1.3.6.1.45.0
    [myOidDefinition] ; Use the same name as the myOidDeiniftion, but omit the␣
↪leading "Oid"
        ds_type = DERIVE
        ds_calc = 8,*  ; RPN expression : Oid, 8, *  Which means Oid * 8 = ds_calc
        ds_oid = $OidmyOidDefinition
[DSTEMPLATE]
    [myCiscoRouter]
        ds = myOidDefinition
[TRIGGER]
    [trigger1]
        warning = RPN expression
        critical = RPN expression
    [trigger2]
        warning = RPN expression
        critical = RPN expression
[TRIGGERGROUP]
    [CiscoRouterTriggers]
        triggers = trigger1, trigger2</code>
```

CHAPTER 3

SnmpBooster Troubleshooting

Check your config

- Have you defined the poller module name?
- Have you defined the correct path to the directory containing your Defaults*.ini files?
- Have you addded the Snmpbooster module to your arbiter, poller, scheduler?
- Have you added copied the genDevConfig templates.cfg in shinken/packs/network/SnmpBooster/
- Have you installed PySNMP, Redis and other dependencies?

Software version consistency

Shinken and SnmpBooster now require **the same Python and Pyro version on all hosts running a Shinken daemon**.

If you cannot use the packaged version of Python and its modules (Pyro, redis, etc.). Use virtualenv to declare a python version to use and install all required modules in that virtualenv.

Software version requirements

Have you verified that the *requirements* are met. Python, PySNMP, Shinken, Pyro, redis, etc.

Validate your check command arguments

Use the check_plugin command and comment out the module to learn what were the exact arguments sent by the poller. This will permit you to validate all the arguments, like snmp community string, inheritance, template application, etc.

Validate connectivity

Take a packet trace using a tool like Wireshark to validate that the remote host is responding.

- Has the host responded
- **Is SnmpBooster repeating the request more often than the polling interval.**
  - If you are seeing repeated requests your device may have a compatibility issues.
  - Save an snmpwalk of the device, get a packet trace using Wireshark, set the poller to debug and save the poller.log file (`/var/log/shinken/pollerd.log`). Send all three to the SnmpBooster developers.

Note: It is normal to see one or more bulkGet requests if you are getting large amounts of data. Ex. a 24 port switch will take 2-3 request packets.

Performance

Make sure you have a low latency connection to your redis from the Poller. Check that redis is running: `netstat -a | grep redis`

Faulty Template

A bad snmp_template file was distributed in the genDevConfig sample-config directory, there were two glaring errors.

This was fixed on 2012-10-16. Make sure you update your template, or use the data from the wiki.

Note that the template should be called: SnmpBooster-template.cfg to make it easier to troubleshoot in the logs. So when you search for SnmpBooster in your logs it will show up as well.

Log files

All warnings and errors generated by the SnmpBooster module start with "[SnmpBooster] error text" and are logged using the standard Shinken logger.

The Arbiter daemon can output initial configuration, loading of host,service keys in redis type error messages. The Scheduler daemon can output scheduling and alert related messages. The Poller daemon can output messages related to instance mapping, acquisition timeouts, invalid community strings, cache failures and more. These are available in the Web interface, as they are placed in the check results for the service.

You can simply do a `grep SnmpBooster *` in your shinken/var directory to see the latest messages related to the SnmpBooster module. You can also sort messages by timestamp to make it easy to find where and when errors occurred.

```
cd shinken/var
grep SnmpBooster *
```

Error codes

Code 0101

Type	INFO
Description	SNMP Booster module starts loading
File	*__init__.py*

Code 0102

Type	ERROR
Description	The attribute **loaded_by** is not set in the Shinken configuration
File	*__init__.py*

Code 0103

Type	ERROR
Description	The attribute **loaded_by** must be *poller*, *scheduler* or *arbiter* and this is not the case
File	*__init__.py*

Code 0201

Type	ERROR
Description	**PySNMP** module can not be loaded. Please checks your installation
File	*libs/checks.py*

Code 0202

Type	ERROR
Description	The current service is not found in the cache (Redis). Maybe you flush it ? Check your Shinken configuration and try to restart the Arbiter to refill the cache (Redio)
File	*libs/checks.py*

Code 0501

Type	WARNING
Description	The Poller didn't found the asked service in the cache (Redis). This error should not appear. Please open an issue on GitHub, if you get it.
File	*libs/results.py*

Code 0502

Type	WARNING
Description	We try to get data from a service which the mapping is not done. We have four possible reasons: • The host is down • The mapping name set in the Shinken service configuration has an error. Please check your configuration • The mapping is not finished yet it will be done in few moments • The instance name set in the Shinken service configuration has an error and it will never found in the mapping SNMP table. Please check your configuration
File	*libs/results.py*

Code 0601

Type	ERROR
Description	**PySNMP** module can not be loaded. Please checks your installation
File	*libs/snmpworker.py*

Code 0602

Type	INFO
Description	The SNMP worker thread is starting
File	*libs/snmpworker.py*

Code 0603

Type	ERROR
Description	We got a SNMP request which is not *get*, *getnext* or *getbulk* Please open an issue on GitHub, if you get it.
File	*libs/snmpworker.py*

Code 0604

Type	INFO
Description	The SNMP worker thread is now stopped
File	*libs/snmpworker.py*

Code 0605

Type	INFO
Description	The SNMP worker thread will be stopped
File	*libs/snmpworker.py*

Code 0606

Type	ERROR
Description	We got a SNMP error. This could be a timeout, a bad response, ...
File	*libs/snmpworker.py*

Code 0701

Type	ERROR
Description	We got a trigger error. It seems that the datasource name use in the trigger doesn't exist. Please check your triggers definitions
File	*libs/trigger.py*

Code 0702

Type	ERROR
Description	We didn't found any collected data in the cache (Redis) to use in the trigger. We have four possible reasons: • The SNMP request is not finished. We have to wait the next check • The oid asked doesn't exists and we never get a value. Please check your Shinken service configuration • The host is down
File	*libs/trigger.py*

Code 0703

Type	ERROR
Description	We didn't found any computed data in the cache (Redis) to use in the trigger. We have two possible reasons: • The current service use a datasource which is a DERIVE, so we need TWO values to compute the derive. • We got an error during the value computation
File	*libs/trigger.py*

Code 0704

Type	ERROR
Description	We got an error during the execution of trigger function. The argument passed to the trigger function has a wrong type or is empty. Please check your trigger configuration
File	*libs/trigger.py*

Code 0705

Type	ERROR
Description	We got an error during the execution of trigger function. The trigger function doesn't exist. Please check your trigger configuration or if it's a new function open an issue on GitHub
File	*libs/trigger.py*

Code 0706

Type	ERROR
Description	We didn't found the asked datasource name defined in the trigger. This could be a typo. Please check your trigger configuration
File	*libs/trigger.py*

Code 0707

Type	ERROR
Description	We got an error during the execution of a trigger. Please check your trigger configuration
File	*libs/trigger.py*

Code 0708

Type	INFO
Description	The trigger triggered. It means the service state will be WARNING or CRITICAL
File	*libs/trigger.py*

Code 0709

Type	ERROR
Description	Unknown trigger error. Maybe it's a good idea to report a bug ?
File	*libs/trigger.py*

Code 0801

Type	WARNING
Description	The parameter **-M** or **–max_rep_map** define in the check command has a bad format. Please check your Shinken configuration
File	*libs/utils.py*

Code 0802

Type	WARNING
Description	The parameter **-g** or **–request_group_size** define in the check command has a bad format. Please check your Shinken configuration
File	*libs/utils.py*

Code 0901

Type	ERROR
Description	**configobj** module can not be loaded. Please checks your installation
File	*snmpbooster_arbiter.py*

Code 0902

Type	INFO
Description	The SNMP Booster module is reading datasource file
File	*snmpbooster_arbiter.py*

Code 0903

Type	INFO
Description	The SNMP Booster module is reading datasource files
File	*snmpbooster_arbiter.py*

Code 0904

Type	ERROR
Description	We got an error merging datasource files. Please check your configuration
File	*snmpbooster_arbiter.py*

Code 0905

Type	ERROR
Description	We got an error merging datasource files. Please check your configuration
File	*snmpbooster_arbiter.py*

Code 0906

Type	ERROR
Description	We got an error during the conversion of the datasource configuration from ini format to python dictionnary format. Please check your configuration
File	*snmpbooster_arbiter.py*

Code 0907

Type	ERROR
Description	We got an error during the serialization of service configuration just before put it in the cache (Redis)
File	*snmpbooster_arbiter.py*

Code 0909

Type	ERROR
Description	We got an error during the update of service configuration in the cache (Redis)
File	*snmpbooster_arbiter.py*

Code 1001

Type	ERROR
Description	We got an error during command line parsing. Please check your check command definition in your Shinken configuration
File	*snmpbooster_poller.py*

Code 1002

Type	ERROR
Description	The SNMP Booster module in the poller can't write check results in the Scheduler queue. You may restart your Poller and/or your Scheduler
File	*snmpbooster_poller.py*

Code 1003

Type	ERROR
Description	The SNMP Booster module in the poller can't write check results in the Scheduler queue. You may restart your Poller and/or your Scheduler
File	*snmpbooster_poller.py*

Code 1004

Type	ERROR
Description	The datasource type is not 'TEXT', 'STRING', 'DERIVE', 'GAUGE', 'COUNTER', 'DERIVE64' or 'COUNTER64'. Please check your Datasource configuration
File	*snmpbooster_poller.py*

Code 1005

Type	WARNING
Description	We get an error while computing service values
File	*snmpbooster_poller.py*

Code 1006

Type	INFO
Description	SNMP Booster Poller module started
File	*snmpbooster_poller.py*

Code 1007

Type	ERROR
Description	The SNMP Booster module in the poller can't read checks results from the Scheduler queue. You may restart your Poller and/or your Scheduler
File	*snmpbooster_poller.py*

Code 1101

Type	INFO
Description	SNMP Booster module loaded
File	*snmpbooster.py*

Code 1102

Type	ERROR
Descrip-tion	The attribute **datasource** is missing in the Shinken module settings. Please check your configuration
File	*snmpbooster.py*

Code 1201

Type	ERROR
Description	**Python Redis** module can not be loaded. Please check your installation
File	*libs/dbclient.py*

Code 1202

Type	ERROR
Description	Can not connect to the Redis server. Please check your configuration
File	*libs/dbclient.py*

Code 1203

Type	ERROR
Description	We got an error while writing in the Redis. The data passed doesn’t seem correct
File	*libs/dbclient.py*

Code 1204

Type	ERROR
Descrip-tion	We got an error while a the upsert in the Redis of a service. This error can only occur on the Arbiter
File	*libs/dbclient.py*

Code 1205

Type	ERROR
Descrip-tion	We got an error updating collected data in the Redis of a service. Thiserror can only occur on the Poller
File	*libs/dbclient.py*

Code 1206

Type	ERROR
Descrip-tion	We got an error updating instance mapping of a service in the Redis. This error can only occur on the Poller
File	*libs/dbclient.py*

Code 1207

Type	ERROR
Description	We got an error getting ONE service in the Redis. This error can only occur on the Poller
File	*libs/dbclient.py*

Code 1208

Type	ERROR
Description	We got an error getting several services in the Redis. This error can only occur on the Poller
File	*libs/dbclient.py*

Code 1301

Type	ERROR
Description	**Python Redis** module can not be loaded. Please check your installation
File	*libs/redisclient.py*

Code 1302

Type	ERROR
Description	Can not connect to the Redis server. Please check your configuration
File	*libs/redisclient.py*

Code 1303

Type	ERROR
Description	We got an error writing service in host:interval list
File	*libs/redisclient.py*

Code 1304

Type	ERROR
Description	We got an error inserting service data in Redis service
File	*libs/redisclient.py*

Code 1305

Type	ERROR
Description	We got an error getting ONE service data in the Redis server
File	*libs/redisclient.py*

Code 1306

Type	ERROR
Description	We got an error getting services list from host:interval key
File	*libs/redisclient.py*

Code 1307

Type	ERROR
Description	We got an error getting ONE service in Redis. This service seems missing
File	*libs/redisclient.py*

Code 1308

Type	ERROR
Description	We got an error getting ONE service in Redis
File	*libs/redisclient.py*

CHAPTER 4

SNMP Booster Cache Manager

SNMP Booster Cache Manager is a tool to perform maintenance tasks for SNMP Booster

```
usage: sbcm.py [-h] [-d DB_NAME] [-b BACKEND] [-r REDIS_ADDRESS]
               [-p REDIS_PORT]
               {search,delete,clear} ...

SNMP Booster Cache Manager

positional arguments:
  {search,delete,clear}
                        sub-command help
    search              search help
    delete              delete help
    clear               clear help

optional arguments:
  -h, --help            show this help message and exit
  -d DB_NAME, --db-name DB_NAME
                        Database name. Default=booster_snmp
  -b BACKEND, --backend BACKEND
                        Backend. Supported : redis. Unsupported: mongodb,
                        memcache
  -r REDIS_ADDRESS, --redis-address REDIS_ADDRESS
                        Redis server address.
  -p REDIS_PORT, --redis-port REDIS_PORT
                        Redis server port.
```

Search commands

```
usage: sbcm.py search [-h] [-H HOST_NAME] [-S SERVICE_NAME] [-t] [-d]

optional arguments:
  -h, --help            show this help message and exit
  -H HOST_NAME, --host-name HOST_NAME
                        Host name
  -S SERVICE_NAME, --service-name SERVICE_NAME
                        Service name
  -t, --show-triggers   Show triggers
  -d, --show-datasource
                        Show datasource
```

Delete commands

```
usage: sbcm.py delete [-h] {host,service} ...

positional arguments:
  {host,service}  delete sub-command help
    host          delete host help
    service       delete service help

optional arguments:
  -h, --help      show this help message and exit
```

Delete services from host

```
usage: sbcm.py delete host [-h] -H HOST_NAME

optional arguments:
  -h, --help            show this help message and exit
  -H HOST_NAME, --host-name HOST_NAME
                        Host name
```

Delete services

```
usage: sbcm.py delete service [-h] -H HOST_NAME -S SERVICE_NAME

optional arguments:
  -h, --help            show this help message and exit
  -H HOST_NAME, --host-name HOST_NAME
                        Host name
  -S SERVICE_NAME, --service-name SERVICE_NAME
                        Service name
```

Clear commands

```
usage: sbcm.py clear [-h] {mapping,cache,old} ...

positional arguments:
  {mapping,cache,old}  clear sub-command help
    mapping            Clear service(s) mapping
    cache                 clear cache help
    old                   clear old help

optional arguments:
  -h, --help            show this help message and exit
```

Clear instance mapping

```
usage: sbcm.py clear mapping [-h] [-H HOST_NAME] [-S SERVICE_NAME]

optional arguments:
  -h, --help            show this help message and exit
  -H HOST_NAME, --host-name HOST_NAME
                        Host name
  -S SERVICE_NAME, --service-name SERVICE_NAME
                        Service name
```

Examples

```
sbcm search -H localhost -S chassis

================================================================================
==   localhost
==   chassis
================================================================================
{'address': u'127.0.0.1',
 'check_interval': 1,
 'check_time': 1414178753.780658,
 'check_time_last': 1414178693.682516,
 'community': 'public',
 'dstemplate': 'Nortel-ERS8600',
 'host': u'localhost',
 'instance_name': '',
 'mapping': None,
 'mapping_name': None,
 'max_rep_map': 64,
 'port': 161,
 'real_check': False,
 'request_group_size': 64,
 'service': u'chassis',
 'timeout': 5,
 'triggergroup': 'chassis_ERS8600',
 'use_getbulk': False,
 'version': '2c'}
```

CHAPTER 5

Design specification

What is it

The SnmpBooster module allows Shinken Pollers to directly manage SNMP data acquisition. This is an all Python cross-platform SNMP module. It is tightly integrated with the Shinken Poller, Scheduler and Arbiter daemons to provide the best possible user experience.

Design specification summary

- STATUS - DESIGN SPEC PERFORMANCE
  - [Done] Functions as an integrated Shinken Poller module
  - [Done] Necessary integration code commited to Shinken official release (Integrated starting at v1.2)
  - [Done] Ability to collect thousands of SNMP metrics per second
  - [Done] Be compatible with distributed data acquisition
  - [Done] Collect data for a host/check_interval tuple via SNMP in a single pass
  - [Done] Use all builtin Shinken scheduler logic for retries, forced checks, timeouts, dependencies, parents
  - [Done] Store collected data for the duration of the check_interval in a Redis
  - [Done] On a restart, after the first collection, be able to pick up where the last check left and calculate derived values
  - [Done] Forced check are not allowed within 30 seconds of last SNMP query to the same host/check_interval, all other requests get data from the cache.
  - [Done] Only a single request to the host/check_interval via SNMP is allowed at a time, all other requests get data from the cache.
- STATUS - DESIGN SPEC USABILITY
  - [Done] Usage documentation

  - [xxxx] Provide sample configuration packs in Shinken
  - [Done] Provide sample config.ini with examples of all types of data
    * SNMP OIDS, DATASOURCES, DSTEMPLATES, TRIGGERS and TRIGGERGROUPS
  - [Done] Directly compatible for use with [[https://github.com/xkilian/genDevConfig|genDevConfig]] Shinken SNMP configuration generator
  - [Done] Provide meaningful feedback for users on errors
  - [Done] Capture all tracebacks and convert them to actual error or warning messages
- STATUS - DESIGN SPEC FEATURES
  - [Done] Return state and performance metrics
  - [Done] Performance metrics can be returned in a Weathermap compatible format
  - [Done] Configuration file format is ConfigObj INI
  - [Done] Load all valid INI configuration files from a directory and merge them
  - [Done] Load a single INI configuration file
  - [xxxx] Load a list of INI configuration files
  - [Done] Configuration file describes all generic acquisition parameters (OID, DATASOURCE, DSTEMPLATE, MAP, TRIGGER, TRIGGERGROUP)
  - [Done] Supports Triggers which are calculation rules to determine states
  - [Done] Triggers support an RPN (Reverse Polish Notation) calculation engine which includes mathematical and logical operators
  - [Done] Each TRIGGER is associated with a severity level, WARNING or CRITICAL
  - [Done] Multiple TRIGGERS can be associated with a TRIGGERGROUP
  - [Done] Use builtin Python Operators
  - [Done] Support DERIVE, TEXT, GAUGE and COUNTER data types
  - [xxxx] Support TIMETICKS data type
  - [Done] Support applying RPN based calculations to received metric for scaling or conversion purposes
  - [Done] Use a Python SNMP library which supports asynchronous acquisition PySNMP
  - [Done] Datasources can use rule based runtime instance mapping
  - [Done] Set Snmp version as a check runtime option
  - [Done] Set Snmp DSTEMPLATE as a check runtime option
  - [Done] Set Snmp TRIGGERGROUP as a check runtime option
  - [Done] Set Snmp COMMUNITY as a check runtime option
  - [Done] Set SNmp DS Max as a check runtime option
  - [Done] Use Snmp version 2c GetBulk
  - [Done] Support Snmp version 2c GetNext if GetBulk is not supported
  - [Done] Support Snmp version 1 GetNext
  - [xxxx] Set Snmp Timeout as a check runtime option, instead of a hardcoded value at 5 seconds
- STATUS - DESIGN SPEC MAINTAINABILITY

- [xxxx] Functions documented in the source code
- [Done] Critical functions documented in the source code
- [Done] Locking sections identified in the code
- [xxxx] Unit tests with at least 80% coverage
- [xxxx] Unit tests integrated with Shinken test suite
- [Done] Code hosted on github
- [Done] configuration validity and integrity checking of all INI files
- [xxxx] Pep8 compliant
- [xxxx] Pylint pass

genDevConfig Plugins - Compatibility status with SnmpBooster

- STATUS - genDevConfig maintained Plugins
  - [Done] Avaya ES switches
  - [Done] Avaya ERS routing switches
  - [Done] Accedian performance probes
  - [Done] Alcatel OS64xx
  - [Done] Alcatel OS68xx, OS69xx
  - [Done] Alcatel OXE
  - [Done] Cisco 29x0 switches
  - [Done] Cisco PIX/ASA
  - [Done] Cisco IOS routers
  - [Done] Geist RS-Mini environmental sensors
  - [Done] IBM IMM and IMM2 modules
  - [Done] IP Forward
  - [InProgress] PaloAlto
  - [Done] JUNOS devices
  - [Done] MIB-II Interfaces
  - [Done] Spectracom SecureSync NTP server
  - [Done] TrippLite NET and NET2 modules
  - [Done] NetSNMP unix hosts ** Validation required**
  - [Done] Packeteer devices ** Validation required**
  - [Done] Sensatronics devices ** Validation required**
  - [Done] Foundry devices ** Validation required**
  - [Done] Packeteer devices ** Validation required**
  - [Done] Cisco CSS ** Validation required**

- STATUS -

Tip:

- [xxxx] Denotes a specification that is planned but not implemented
- [InProgress] Denotes a specification that is under development
- [Done] Denotes a specification that is implemented

CHAPTER 6

SnmpBooster reference dictionary

SnmpBooster.ini dictionary

There are five dictionaries:

- *DATASOURCE*
- *DSTEMPLATE*
- *MAP*
- *TRIGGER*
- *TRIGGERGROUP*

DATASOURCE DICTIONARY

OidVariableName refers to an actual OID that can be queries using SNMP against the device on the network.

[VariableName] refers to a Datasource and all the information required to gather and prepare the data using SNMP ds_type refers to how the data should be prepared ds_calc refers to any scaling manipulations to make the data more understandable. This is an RPN expression, where the first variable is omitted, as it is always the $OidVariable ds_oid refers to the actual $OidVariable name. An instance identifier can be appended to the name to signify that an instance is provided by the Shinken service definition. This information is passed when the check is called. ...

DSTEMPLATE DICTIONNARY

[DsTemplateName] refers to the name of the DSTEMPLATE that will be referred to in the Shinken service definitions. ds refers to the list of DATASOURCES to be collected. If an instance is expected for the list of DATASOURCES, it MUST be the same instance for all Oids. If a different instance is required, use a second DSTEMPLATE.

TRIGGER DICTIONNARY

...

TRIGGERGROUP DICTIONNARY

...

MAP DICTIONNARY

...

SnmpBooster.ini example configuration

This example definition will be used to explain each section.

```
[DATASOURCE]
    OidmyOidDefinition = .1.3.6.1.45.0
    [myOidDefinition] ; Use the same name as the myOidDeiniftion, but omit the␣
↪leading "Oid"
        ds_type = DERIVE
        ds_calc = 8,*  ; RPN expression : Oid, 8, *  Which means Oid * 8 = Total
        ds_oid = $OidmyOidDefinition
[DSTEMPLATE]
    [myCiscoRouter]
        ds = myOidDefinition
[TRIGGER]
    [trigger1]
        warning = RPN expression
        critical = RPN expression
    [trigger2]
        warning = RPN expression
        critical = RPN expression
[TRIGGERGROUP]
    [CiscoRouterTriggers]
        triggers = trigger1, trigger2
```

Note: You cannot use operator characters in a variable name : "-+*/". You will get error in poller log after because it not able to get a variable name.

SnmpBooster.ini configuring SNMP Datasources

The first location is generic data related to SNMP parameters.

- DATASOURCE information * SNMP OID * Type of data and how can it be interpreted (GAUGE, COUNTER, COUNTER64, DERIVE, DERIVE64, TEXT, TIMETICK) * Data format preparation (Scaling the data for example bits to bytes) * Is there an instance to append to the
- Instance MAP function * Mapping the instance dynamically using a function * Data or rules related to the mapping function

SnmpBooster.ini configuring SNMP DSTEMPLATES

- DSTEMPLATEs to associate DATASOURCE to actual device classes * List of DATASOURCES associated with a, for example, Cisco 1900 router. Which in turn can be applied to a Shinken service

SnmpBooster.ini setting triggers/thresholds

- TRIGGER and TRIGGERGROUPS to apply thresholding rules * Define triggers and associate them with a TRIGGERGROUP name that can be applied to a Shinken Service

CHAPTER 7

SNMP Booster Developer Reference

SNMP Booster root class

This module contains the SnmpBoosterArbiter class which is the part of SNMP Booster loaded in the Arbiter

class `module.snmpbooster.`**`SnmpBooster`**(*mod_conf*)

Bases: `BaseModule`

SNMP Poller module class Improve SNMP checks

`init`()

Called by poller to say 'let's prepare yourself guy'

SNMP Booster classes

This module contains the SnmpBoosterArbiter class which is the part of SNMP Booster loaded in the Arbiter

class `module.snmpbooster_arbiter.`**`SnmpBoosterArbiter`**(*mod_conf*)

Bases: *`module.snmpbooster.SnmpBooster`*

SNMP Poller module class Improve SNMP checks

`hook_late_configuration`(*arb*)

Read config and fill database

This module contains the SnmpBoosterScheduler class which is the part of SNMP Booster loaded in the Scheduler

class `module.snmpbooster_scheduler.`**`SnmpBoosterScheduler`**(*mod_conf*)

Bases: *`module.snmpbooster.SnmpBooster`*

SNMP Poller module class Improve SNMP checks

static **`get_frequence`**(*chk*)

return check_interval if state type is HARD else retry_interval if state type is SOFT

hook_get_new_actions(*sche*)
Set if is a SNMP or Cache check

static **set_true_check**(*check*, *real=False*)
Add -r option to the command line

This module contains the SnmpBoosterPoller class which is the part of SNMP Booster loaded in the Poller

class `module.snmpbooster_poller.`**SnmpBoosterPoller**(*mod_conf*)
Bases: `module.snmpbooster.SnmpBooster`

SNMP Poller module class Improve SNMP checks

get_new_checks()
Get new checks if less than nb_checks_max If no new checks got and no check in queue, sleep for 1 sec REF: doc/shinken-action-queues.png (3)

launch_new_checks()
Launch checks that are in status REF: doc/shinken-action-queues.png (4)

manage_finished_checks()
This function handles finished check It gets output and exit_code and Add check to the return queue

save_results()
Save results to database

work(*master_slave_queue*, *returns_queue*, *control_queue*)
Main loop of SNMP Booster

SNMP Booster libs

This module contains two functions: * check_cache: Get data from cache * check_snmp: Get data from SNMP request

`module.libs.checks.`**check_cache**(*check*, *arguments*, *db_client*)
Get data from database

`module.libs.checks.`**check_snmp**(*check*, *arguments*, *db_client*, *task_queue*, *result_queue*)
Prepare snmp requests

This module contains database/cache abstraction class

class `module.libs.dbclient.`**DBClient**(*db_host*, *db_port*, *db_name*)
Bases: `object`

Class used to abstract the use of the database/cache

connect()
This function inits the connection to the database

disconnect()
This function kills the connection to the database

get_service(*host*, *service*)
This function gets one service from the database Return :query_result: dict

get_services(*host*, *check_interval*)
This function Gets all services with the same host and check_interval Return :query_result: list of dicts

static **handle_error**(*result*, *context=''*)
This function handles mongodb errors

update_service (*host*, *service*, *data*)
This function updates/inserts a service It used by arbiter in hook_late_configuration to put the configuration in the database Return * query_result: None * error: bool

update_service_init (*host*, *service*, *data*)

update_service_instance (*host*, *instance_name*, *instance*)
This function update a instance from SNMP mapping requests Return * query_result: None * error: bool

This module contains a set of functions to format the plugin output which is shown on the UI

`module.libs.output.`**format_output** (*service*, *ds_name*)
Format value for derive type

`module.libs.output.`**get_output** (*service*)
Prepare service output

`module.libs.output.`**prepare_format** (*value*, *ds_data*)
Prepare a dict to put in string formatting

This module contains a function to retrieve output and compute trigger

`module.libs.result.`**set_output_and_status** (*check_result*)
get output, compute exit_code an return it

This module contains a class to create a Thread which make SNMP requests and handle answers with callbacks

class `module.libs.snmpworker.`**SNMPWorker** (*mapping_queue*, *max_prepared_tasks*)
Bases: `threading.Thread`

Thread which execute all SNMP tasks/requests

append_task_to_dispatcher (*snmp_task*)

real_run ()
Process SNMP tasks SNMP task is a dict: - For a bulk request

```
{"authData": cmdgen.CommunityData('public')
 "transportTarget": cmdgen.UdpTransportTarget((transportTarget, 161))
 "nonRepeaters": 0
 "maxRepetitions": 64
 "varNames": ['1.3.6.1.2.1.2.2.1.2.0', '...']
 "cbInfo:: (cbFun, (arg1, arg2, ...))
 }
```

•For a next request

```
{"authData": cmdgen.CommunityData('public')
 "transportTarget": cmdgen.UdpTransportTarget((transportTarget, 161))
 "varNames": ['1.3.6.1.2.1.2.2.1.2.0', '...']
 "cbInfo:: (cbFun, (arg1, arg2, ...))
}
```

•For a get request

```
{"authData": cmdgen.CommunityData('public)
 "transportTarget": cmdgen.UdpTransportTarget((transportTarget, 161))
 "varNames": ['1.3.6.1.2.1.2.2.1.2.0', '...']
 "cbInfo:: (cbFun, (arg1, arg2, ...))
}
```

run()

stop_worker()
Stop SNMP worker thread

module.libs.snmpworker.**callback_get**(*send_request_handle*, *error_indication*, *error_status*, *error_index*, *var_binds*, *cb_ctx*)
Callback function for GET SNMP requests

module.libs.snmpworker.**callback_mapping_bulk**(*send_request_handle*, *error_indication*, *error_status*, *error_index*, *var_binds*, *cb_ctx*)
Callback function for BULK SNMP requests

module.libs.snmpworker.**callback_mapping_next**(*send_request_handle*, *error_indication*, *error_status*, *error_index*, *var_binds*, *cb_ctx*)
Callback function for GENEXT SNMP requests

module.libs.snmpworker.**handle_snmp_error**(*error_indication*, *cb_ctx*, *request_type*)
Handle SNMP errors

This module contains the function which compute triggers and return the exit code of a service

module.libs.trigger.**get_trigger_result**(*service*)
Get return code from trigger calculator return error_message, exit_code :error_message: is None if there no error :exit_code: 0, 1, 2 or 3

Usefull functions used everywhere in snmp booster

module.libs.utils.**calculation**(*value*, *ds_calc*)
Get result from calc

```
>>> calculation(1, [2, "add"])
3.0
```

module.libs.utils.**compute_value**(*result*)
Get a computed value from raw_value, ds_type and calculation result argument must have this form

```
{'value_last': u'0',
 'calc': None,
 'check_time': 1410456115.376102,
 'key': {'host': u'myhost1',
         'ds_names': [u'ifOutErrors'],
         'service': u'if.lo',
         'oid_type': 'ds_oid'},
 'check_time_last': 1410456100.722268,
 'value_last_computed': u'0',
 'type': u'TEXT',
 'value': Counter32(0),
}
```

```
>>> data = {}
>>> data['value_last'] = u'0'
>>> data['calc'] = None
>>> data['check_time'] = 1410456115.376102
>>> data['key'] = {}
>>> data['key']['host'] = u'myhost1'
>>> data['key']['ds_names'] = [u'ifOutErrors']
>>> data['key']['service'] = u'if.lo'
>>> data['key']['oid_type'] = 'ds_oid'
>>> data['check_time_last'] = 1410456100.722268
>>> data['value_last_computed'] = u'Text collected from SNMP'
>>> data['type'] = u'TEXT'
>>> data['value'] = "Text collected from SNMP"
>>> compute_value(data)
'Text collected from SNMP'
```

`module.libs.utils.`**`derive`**(*value*, *value_last*, *check_time*, *check_time_last*, *limit=4294967295*)
Get a derive

```
>>> derive(20, 10, 1412776670, 1412776660)
1.0
>>> derive(15, 4294967290, 1412776670, 1412776660)
2.0
>>> derive(20, 2**64 - 11, 1412776670, 1412776660, 2**64 - 1)
3.0
```

`module.libs.utils.`**`dict_serialize`**(*serv*, *mac_resol*, *datasource*)
Get serv, datasource And return the service serialized

`module.libs.utils.`**`flatten_dict`**(*tree_dict*)
Convert unlimited tree dictionnary to a flat dictionnary

```
>>> flatten_dict({'a': 1, 'b': {'c': {'d': 2, 'e': 4}}})
{'a': 1, 'b.c.d': 2, 'b.c.e': 4}
>>> flatten_dict("bad_input")
```

`module.libs.utils.`**`format_counter64_value`**(*result*)
Format value for counter64 type

`module.libs.utils.`**`format_counter_value`**(*result*, *limit=4294967295*)
Format value for counter type

`module.libs.utils.`**`format_derive64_value`**(*result*)
Format value for derive64 type

`module.libs.utils.`**`format_derive_value`**(*result*, *limit=4294967295*)
Format value for derive type

`module.libs.utils.`**`format_gauge_value`**(*result*)
Format value for gauge type

`module.libs.utils.`**`format_text_value`**(*result*)
Format value for text type

`module.libs.utils.`**`merge_dicts`**(*old_dict*, *new_dict*)
Convert unlimited tree dictionnary to a flat dictionnary

```
>>> flatten_dict({'a': 1, 'b': {'c': {'d': 2, 'e': 4}}})
{'a': 1, 'b.c.d': 2, 'b.c.e': 4}
>>> flatten_dict("bad_input")
```

`module.libs.utils.`**`parse_args`**(*cmd_args*)
Parse service command line and return a dict

`module.libs.utils.`**`rpn_calculator`**(*rpn_list*)
Reverse Polish notation calculator

```
>>> rpn_calculator([4, 5, "add"])
9.0
>>> rpn_calculator([1, 2, "eq"])
False
```

```
>>> rpn_calculator([3, 2, "gt", 1, 1, "eq", "and_"])
True
```

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index