
Shinken SNMP Booster Module Documentation

Release 1.0

Thibault Cohen

November 20, 2015

1	SNMP Booster: How does it works	3
1.1	Overview	3
1.2	How does it work	4
1.3	Limitations	5
1.4	Design specification	5
1.5	Data model	5
1.6	Installation and configuration	6
1.7	Reference Dictionary	6
1.8	Troubleshooting	6
1.9	Graph templates	6
2	SNMP Booster: Install and setup	7
2.1	SnmpBooster Download Install and Configure	7
2.2	Downloads	7
2.3	Requirements	7
2.4	Installation	8
2.5	Configuration	8
3	SnmpBooster Troubleshooting	15
3.1	Check your config	15
3.2	Software version consistency	15
3.3	Software version requirements	15
3.4	Validate your check command arguments	15
3.5	Validate connectivity	15
3.6	Performance	16
3.7	Faulty Template	16
3.8	Log files	16
3.9	Error codes	16
4	SNMP Booster Cache Manager	23
4.1	Search commands	23
4.2	Delete commands	24
4.3	Clear commands	24
5	Design specification	27
5.1	What is it	27
5.2	Design specification summary	27
5.3	genDevConfig Plugins - Compatibility status with SnmpBooster	29

6	SnmpBooster reference dictionary	31
6.1	SnmpBooster.ini dictionary	31
6.2	SnmpBooster.ini example configuration	32
6.3	SnmpBooster.ini configuring SNMP Datasources	32
6.4	SnmpBooster.ini configuring SNMP DSTEMPLATES	32
6.5	SnmpBooster.ini setting triggers/thresholds	32
7	SNMP Booster Developer Reference	33
7.1	SNMP Booster root class	33
7.2	SNMP Booster classes	33
7.3	SNMP Booster libs	34
8	Indices and tables	39
	Python Module Index	41

Contents:

SNMP Booster: How does it works

1.1 Overview

1.1.1 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.

1.1.2 Why use it

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

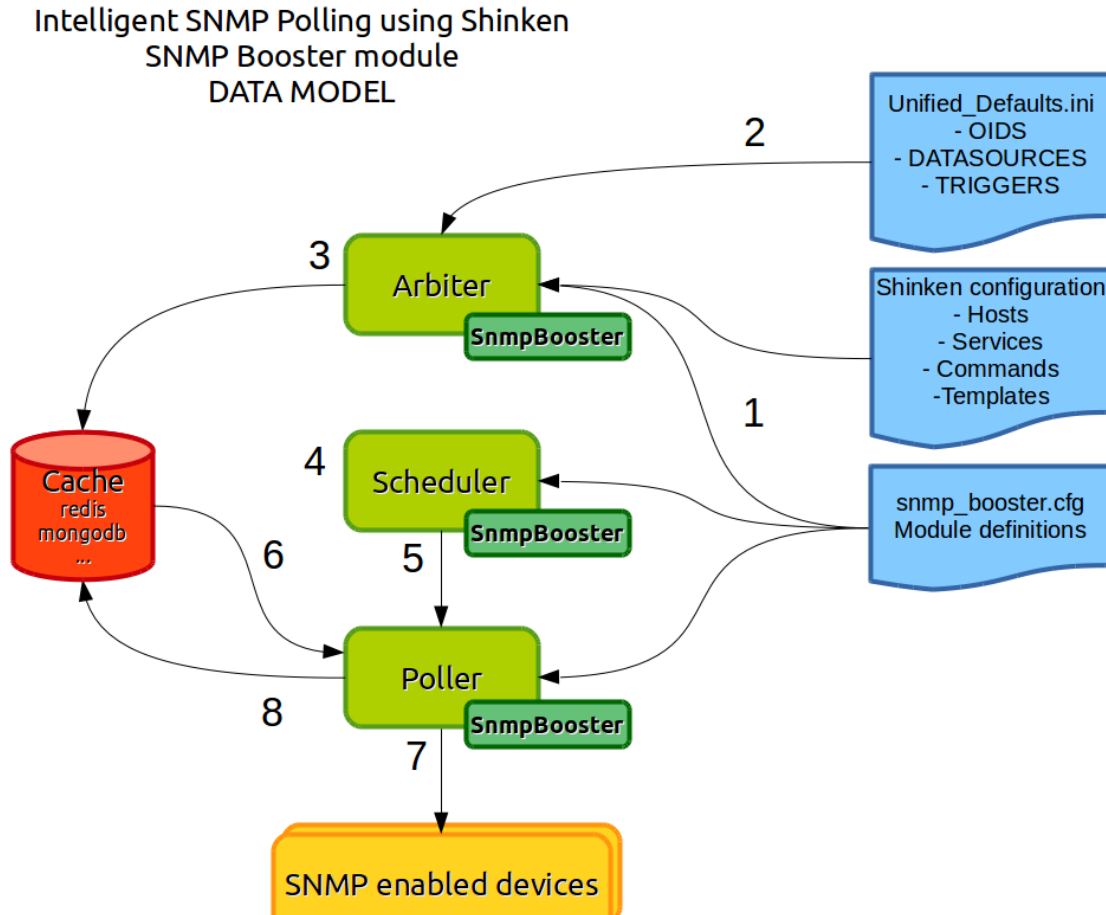
This acquisition module was professionally designed and developed.

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

It is one of the very few serious SNMP v2c implementation making use of SnmpGetBulk type requests.

1.2 How does it work

1.2.1 Shinken Integration



- 1 - The SnpmpBooster Arbiter module reads the Shinken SnpmpBooster 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 create 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 SnpmpBooster Arbiter module computes Shinken configuration with datasource files (.ini files) and prepare datas for Redis
- 3 - The SnpmpBooster Arbiter module stores an entry in Redis for each service defined in Shinken configuration
- 4 - The SnpmpBooster 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 SnpmpBooster 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

1.2.2 Performance

SnmpBooster uses SNMP v2c getbulk for high efficiency, unless forced to use SNMP v2c get-next or SNMPv1 get-next. GetBulk uses 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.*

The SnmpBooster module supports automatic instance mapping for OIDs. (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

1.3 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.

1.4 Design specification

SnmpBooster design specification and current development status.

1.5 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 referred 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
- DTEMPLATES 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

1.6 Installation and configuration

SnmpBooster installation

1.7 Reference Dictionary

SnmpBooster reference dictionary

1.8 Troubleshooting

SnmpBooster troubleshooting

1.9 Graph templates

These are .graph files defined in your Shinken configuration directory. Refer to the Shinken graphite templates(Not yet created) or PNP4Nagios how-to documentation. The graph templates are independent from SnmpBooster and provide templates for any collected data from Shinken.

SNMP Booster: Install and setup

2.1 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*

2.2 Downloads

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

- <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/

2.3 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.

2.4 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.

2.5 Configuration

2.5.1 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
    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.conf`. 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

2.5.2 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 $_HOSTSNMP
    module_type     snmp_booster
}

define command {
    command_name    check_snmp_booster_bulk
    command_line    check_snmp_booster -H $HOSTNAME$ -A $HOSTADDRESS$ -S '$SERVICEDESC$' -C $_HOSTSNMP
    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: <i>public</i> |
| -P, --port | SNMP port; Default: <i>161</i> |
| -V, --snmp-version | SNMP version; Default: <i>2c</i> |
| -s, --timeout | SNMP request timeout; Default: <i>5</i> (seconds) |
| -t, --dstemplate | dstemplate name; Example: <i>standard-interface</i> ; (mandatory) |
| -i, --instance | instance (no mapping need); Example: <i>1.32.4</i> |
| -n, --instance-name | instance name use for mapping; Example: <i>Intel_Corporation_82579LM_Gigabit_Network_Connection</i> |
| -m, --mapping | OID used to do the mapping; Example: <i>1.3.6.1.2.1.2.2.1.2</i> |
| -N, --mapping-name | name of the OID used to do the mapping; Example: <i>interface-name</i> |
| -T, --triggergroup | name of the trigger group which contains several triggers; Example: <i>interface-hc</i> |
| -b, --use-getbulk | use snmp getbulk requests to do the mapping; Default: <i>0</i> |
| -M, --max-rep-map | max_repetition parameters for snmp getbulk requests; Default: <i>64</i> |
| -g, --request-group-size | max number of asked oids in one SNMP request; Default: <i>64</i> |
| -c, --no-concurrency | Disable concurrent SNMP requests on the same host; Default: <i>0</i> |

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
    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:

<code>_snmpcommunityread</code>	<code>public</code>	<code>; which could be set in your resource.cfg file</code>
<code>_snmpversion</code>	<code>public</code>	<code>; which could be set in your resource.cfg file</code>
<code>_usebulk</code>	<code>0</code>	<code>; use bulk request to do mapping</code>
<code>_noconcurrency</code>	<code>0</code>	<code>; SNMPBooster can make multiple requests on the same host</code>

Mandatory service arguments related to SNMP polling:

<code>_dstemplate</code>	<code>Cisco-Generic-Router</code>	<code>; Name of a DSTEMPLATE defined in the SnmpBooster config.ini file</code>
--------------------------	-----------------------------------	--

Optional service arguments related to SNMP polling with default values:

<code>_inst</code>	<code>None</code>	<code>; Could be numeric: 0, 0.0.1, None</code>
<code>_instname</code>	<code>None</code>	<code>; Instance name use to do mapping</code>
<code>_triggergroup</code>	<code>None</code>	<code>; Name of the triggergroup defined in the SnmpBooster config.ini file</code>
<code>_mapping</code>	<code>None</code>	<code>; Mapping name defined in [MAP] section in the SnmpBooster ini files</code>

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

<code>_instname</code>	<code>FastEthernet0_1</code>
<code>_mapping</code>	<code>interface-name</code>

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 S
#   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 ethernetCsr
    _dstemplate         standard-interface
    _instname           FastEthernet0_1
    _mapping             interface-name
    use                 default-snmp-template
    register           1
}
```

2.5.3 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>
```

SnmpBooster Troubleshooting

3.1 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 added 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?

3.2 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.

3.3 Software version requirements

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

3.4 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.

3.5 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 Snmp-Booster 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.

3.6 Performance

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

3.7 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.

3.8 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 *
```

3.9 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 <i>poller</i> , <i>scheduler</i> or <i>arbiter</i> and this is not the case
	File	<i>__init__.py</i>

Code 0201	Type	ERROR
	Description	PySNMP module can not be loaded. Please checks your installation
	File	<i>libs/checks.py</i>

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	<i>libs/checks.py</i>

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	<i>libs/results.py</i>

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	<i>libs/results.py</i>

Code 0601	Type	ERROR
	Description	PySNMP module can not be loaded. Please checks your installation
	File	<i>libs/snmpworker.py</i>

Code 0602	Type	INFO
	Description	The SNMP worker thread is starting
	File	<i>libs/snmpworker.py</i>

Code 0603	Type	ERROR
	Description	We got a SNMP request which is not <i>get</i> , <i>getnext</i> or <i>getbulk</i> Please open an issue on GitHub, if you get it.
	File	<i>libs/snmpworker.py</i>

Code 0604	Type	INFO
	Description	The SNMP worker thread is now stopped
	File	<i>libs/snmpworker.py</i>

Code 0605	Type	INFO
	Description	The SNMP worker thread will be stopped
	File	<i>libs/snmpworker.py</i>

Code 0606	Type	ERROR
	Description	We got a SNMP error. This could be a timeout, a bad response, ...
	File	<i>libs/snmpworker.py</i>

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	<i>libs/trigger.py</i>

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	<i>libs/trigger.py</i>

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	<i>libs/trigger.py</i>

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	<i>libs/trigger.py</i>

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	<i>libs/trigger.py</i>

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	<i>libs/trigger.py</i>

Code 0707	Type	ERROR
	Description	We got an error during the execution of a trigger. Please check your trigger configuration
	File	<i>libs/trigger.py</i>

Code 0708	Type	INFO
	Description	The trigger triggered. It means the service state will be WARNING or CRITICAL
	File	<i>libs/trigger.py</i>
Code 0709	Type	ERROR
	Description	Unknown trigger error. Maybe it's a good idea to report a bug ?
	File	<i>libs/trigger.py</i>
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	<i>libs/utlis.py</i>
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	<i>libs/utlis.py</i>
Code 0901	Type	ERROR
	Description	configobj module can not be loaded. Please checks your installation
	File	<i>snmpbooster_arbiter.py</i>
Code 0902	Type	INFO
	Description	The SNMP Booster module is reading datasource file
	File	<i>snmpbooster_arbiter.py</i>
Code 0903	Type	INFO
	Description	The SNMP Booster module is reading datasource files
	File	<i>snmpbooster_arbiter.py</i>
Code 0904	Type	ERROR
	Description	We got an error merging datasource files. Please check your configuration
	File	<i>snmpbooster_arbiter.py</i>
Code 0905	Type	ERROR
	Description	We got an error merging datasource files. Please check your configuration
	File	<i>snmpbooster_arbiter.py</i>
Code 0906	Type	ERROR
	Description	We got an error during the conversion of the datasource configuration from ini format to python dictionary format. Please check your configuration
	File	<i>snmpbooster_arbiter.py</i>
Code 0907	Type	ERROR
	Description	We got an error during the serialization of service configuration just before put it in the cache (Redis)
	File	<i>snmpbooster_arbiter.py</i>
Code 1001	Type	ERROR
	Description	We got an error during command line parsing. Please check your check command definition in your Shinken configuration
	File	<i>snmpbooster_poller.py</i>

Code 1002	Type	ERROR
	De-scription	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	<i>snmpbooster_poller.py</i>
Code 1003	Type	ERROR
	De-scription	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	<i>snmpbooster_poller.py</i>
Code 1004	Type	ERROR
	De-scription	The datasource type is not 'TEXT', 'STRING', 'DERIVE', 'GAUGE', 'COUNTER', 'DERIVE64' or 'COUNTER64'. Please check your Datasource configuration
	File	<i>snmpbooster_poller.py</i>
Code 1005	Type	WARNING
	Description	We get an error while computing service values
	File	<i>snmpbooster_poller.py</i>
Code 1006	Type	INFO
	Description	SNMP Booster Poller module started
	File	<i>snmpbooster_poller.py</i>
Code 1007	Type	ERROR
	De-scription	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	<i>snmpbooster_poller.py</i>
Code 1101	Type	INFO
	Description	SNMP Booster module loaded
	File	<i>snmpbooster.py</i>
Code 1102	Type	ERROR
	Description	The attribute datasource is missing in the Shinken module settings. Please check your configuration
	File	<i>snmpbooster.py</i>
Code 1201	Type	ERROR
	Description	Python Redis module can not be loaded. Please check your installation
	File	<i>libs/dbclient.py</i>
Code 1202	Type	ERROR
	Description	Can not connect to the Redis server. Please check your configuration
	File	<i>libs/dbclient.py</i>
Code 1203	Type	ERROR
	Description	We got an error while writing in the Redis. The data passed doesn't seem correct
	File	<i>libs/dbclient.py</i>
Code 1204	Type	ERROR
	Description	We got an error while a the upsert in the Redis of a service. This error can only occur on the Arbiter
	File	<i>libs/dbclient.py</i>

Code 1205	Type	ERROR
	Description	We got an error updating collected data in the Redis of a service. This error can only occur on the Poller
	File	<i>libs/dbclient.py</i>
Code 1206	Type	ERROR
	Description	We got an error updating instance mapping of a service in the Redis. This error can only occur on the Poller
	File	<i>libs/dbclient.py</i>
Code 1207	Type	ERROR
	Description	We got an error getting ONE service in the Redis. This error can only occur on the Poller
	File	<i>libs/dbclient.py</i>
Code 1208	Type	ERROR
	Description	We got an error getting several services in the Redis. This error can only occur on the Poller
	File	<i>libs/dbclient.py</i>
Code 1301	Type	ERROR
	Description	Python Redis module can not be loaded. Please check your installation
	File	<i>libs/redisclient.py</i>
Code 1302	Type	ERROR
	Description	Can not connect to the Redis server. Please check your configuration
	File	<i>libs/redisclient.py</i>
Code 1303	Type	ERROR
	Description	We got an error writing service in host:interval list
	File	<i>libs/redisclient.py</i>
Code 1304	Type	ERROR
	Description	We got an error inserting service data in Redis service
	File	<i>libs/redisclient.py</i>
Code 1305	Type	ERROR
	Description	We got an error getting ONE service data in the Redis server
	File	<i>libs/redisclient.py</i>
Code 1306	Type	ERROR
	Description	We got an error getting services list from host:interval key
	File	<i>libs/redisclient.py</i>
Code 1307	Type	ERROR
	Description	We got an error getting ONE service in Redis. This service seems missing
	File	<i>libs/redisclient.py</i>
Code 1308	Type	ERROR
	Description	We got an error getting ONE service in Redis
	File	<i>libs/redisclient.py</i>

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 help
                        delete help
                        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.
```

4.1 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
```

4.2 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
```

4.2.1 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
```

4.2.2 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
```

4.3 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
```

4.3.1 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
```

Design specification

5.1 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.

5.2 Design specification summary

- STATUS - DESIGN SPEC PERFORMANCE
 - [Done] Functions as an integrated Shinken Poller module
 - [Done] Necessary integration code committed 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] SkonfUI and Discovery 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 [\[\[https://github.com/xkilian/genDevConfig|genDevConfig\]\]](https://github.com/xkilian/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, DSTEM-
PLATE, 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 mathemati-
cal 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] Use Snmp version 2c GetBulk
 - [xxxx] Support Snmp version 2c GetNext if GetBulk is not supported
 - [xxxx] 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
- [xxxx] configuration validity and integrity checking of all INI files
- [xxxx] Pep8 compliant
- [xxxx] Pylint pass

5.3 genDevConfig Plugins - Compatibility status with SnmpBooster

- STATUS - genDevConfig maintained Plugins
 - [Done] Avaya ES switches
 - [Done] Avaya ERS routing switches
 - [Done] Cisco 29x0 switches
 - [Done] MIB-II Interfaces
 - [Done] Cisco PIX/ASA
 - [Done] JUNOS devices
 - [Done] Cisco IOS routers
 - [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**
 - [InProgress] New Cisco Access points
- STATUS - Other maintained Plugins

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
-

SnmpBooster reference dictionary

6.1 SnmpBooster.ini dictionary

There are five dictionaries:

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

6.1.1 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. ...

6.1.2 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.

6.1.3 TRIGGER DICTIONNARY

...

6.1.4 TRIGGERGROUP DICTIONNARY

...

6.1.5 MAP DICTIONNARY

...

6.2 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
```

6.3 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

6.4 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

6.5 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

SNMP Booster Developer Reference

7.1 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'
```

7.2 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
```

7.3 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 init's 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 is 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 and 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*)

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, ...))
}
```

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 dictionary to a flat dictionary

```
>>> 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 dictionary to a flat dictionary

```
>>> 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
```

Indices and tables

- `genindex`
- `modindex`
- `search`

m

- `module`, 33
 - `module.libs`, 34
 - `module.libs.checks`, 34
 - `module.libs.dbclient`, 34
 - `module.libs.output`, 35
 - `module.libs.result`, 35
 - `module.libs.snmpworker`, 35
 - `module.libs.trigger`, 36
 - `module.libs.utils`, 36
 - `module.snmpbooster`, 33
 - `module.snmpbooster_arbiter`, 33
 - `module.snmpbooster_poller`, 33
 - `module.snmpbooster_scheduler`, 33

A

append_task_to_dispatcher() (module.
 module.libs.snmpworker.SNMPWorker method),
 35

C

calculation() (in module module.libs.utils), 36
 callback_get() (in module module.libs.snmpworker), 35
 callback_mapping_bulk() (in module module.
 module.libs.snmpworker), 35
 callback_mapping_next() (in module module.
 module.libs.snmpworker), 36
 check_cache() (in module module.libs.checks), 34
 check_snmp() (in module module.libs.checks), 34
 compute_value() (in module module.libs.utils), 36
 connect() (module.libs.dbclient.DBClient method), 34

D

DBClient (class in module.libs.dbclient), 34
 derive() (in module module.libs.utils), 36
 dict_serialize() (in module module.libs.utils), 37
 disconnect() (module.libs.dbclient.DBClient method), 34

F

flatten_dict() (in module module.libs.utils), 37
 format_counter64_value() (in module module.libs.utils),
 37
 format_counter_value() (in module module.libs.utils), 37
 format_derive64_value() (in module module.libs.utils),
 37
 format_derive_value() (in module module.libs.utils), 37
 format_gauge_value() (in module module.libs.utils), 37
 format_output() (in module module.libs.output), 35
 format_text_value() (in module module.libs.utils), 37

G

get_frequence() (module.snmpbooster_scheduler.SnmpBoosterScheduler
 static method), 33
 get_new_checks() (module.
 module.snmpbooster_poller.SnmpBoosterPoller
 method), 34

get_output() (in module module.libs.output), 35
 get_service() (module.libs.dbclient.DBClient method), 34
 get_services() (module.libs.dbclient.DBClient method),
 34
 get_trigger_result() (in module module.libs.trigger), 36

H

handle_error() (module.libs.dbclient.DBClient static
 method), 34
 handle_snmp_error() (in module module.
 module.libs.snmpworker), 36
 hook_get_new_actions() (module.
 module.snmpbooster_scheduler.SnmpBoosterScheduler
 method), 33
 hook_late_configuration() (module.
 module.snmpbooster_arbiter.SnmpBoosterArbiter
 method), 33

I

init() (module.snmpbooster.SnmpBooster method), 33

L

launch_new_checks() (module.
 module.snmpbooster_poller.SnmpBoosterPoller
 method), 34

M

manage_finished_checks() (module.
 module.snmpbooster_poller.SnmpBoosterPoller
 method), 34
 merge_dicts() (in module module.libs.utils), 37
 module (module), 33
 module.libs (module), 34
 module.libs.checks (module), 34
 module.libs.dbclient (module), 34
 module.libs.output (module), 35
 module.libs.result (module), 35
 module.libs.snmpworker (module), 35
 module.libs.trigger (module), 36
 module.libs.utils (module), 36

`module.snmpbooster` (module), [33](#)
`module.snmpbooster_arbiter` (module), [33](#)
`module.snmpbooster_poller` (module), [33](#)
`module.snmpbooster_scheduler` (module), [33](#)

P

`parse_args()` (in module `module.libs.utils`), [37](#)
`prepare_format()` (in module `module.libs.output`), [35](#)

R

`rpn_calculator()` (in module `module.libs.utils`), [37](#)
`run()` (module `module.libs.snmpworker.SNMPWorker` method),
[35](#)

S

`save_results()` (module `module.snmpbooster_poller.SnmpBoosterPoller`
method), [34](#)
`set_output_and_status()` (in module `module.libs.result`),
[35](#)
`set_true_check()` (module `module.snmpbooster_scheduler.SnmpBoosterScheduler`
static method), [33](#)
`SnmpBooster` (class in module `module.snmpbooster`), [33](#)
`SnmpBoosterArbiter` (class in module `module.snmpbooster_arbiter`), [33](#)
`SnmpBoosterPoller` (class in module `module.snmpbooster_poller`), [33](#)
`SnmpBoosterScheduler` (class in module `module.snmpbooster_scheduler`), [33](#)
`SNMPWorker` (class in module `module.libs.snmpworker`), [35](#)
`stop_worker()` (module `module.libs.snmpworker.SNMPWorker`
method), [35](#)

U

`update_service()` (module `module.libs.dbclient.DBClient`
method), [34](#)
`update_service_init()` (module `module.libs.dbclient.DBClient`
method), [34](#)
`update_service_instance()` (module `module.libs.dbclient.DBClient` method), [34](#)

W

`work()` (module `module.snmpbooster_poller.SnmpBoosterPoller`
method), [34](#)