slapos.core Documentation

Release 1.6.10

Vifib

May 03, 2021

Contents

Contents:

CHAPTER 1

SlapOS command line usage

1.1 Notes

- Default SlapOS Master is https://slap.vifib.com. It can be changed by altering configuration files or with the `--master-url` argument for commands that support it.
- Most commands take a configuration file parameter, provided as `--cfg /path/to/file.cfg`.

  If no such argument is provided:

  - "node" commands read configuration from `/etc/opt/slapos/slapos.cfg`, or the file referenced by the `SLAPOS_CONFIGURATION` environment variable.
  - likewise, "client" commands (request, supply...) use `~/.slapos/slapos.cfg`, or the `SLAPOS_CLIENT_CONFIGURATION` variable.

1.2 Common options

Without arguments, the `slapos` program lists all the available commands and common options.

```
usage: slapos [--version] [-v | -q] [--debug] [--log-file LOG_FILE]
              [--log-color] [--log-time] [-h]

SlapOS client 1.6.10

optional arguments:
  --version             show program's version number and exit
  -v, --verbose         Increase verbosity of output. Can be repeated.
  -q, --quiet           Suppress output except warnings and errors.
  --debug               Show tracebacks on errors.
  --log-file LOG_FILE, --logfile LOG_FILE, --log_file LOG_FILE
                        Specify a file to log output (default: console only)
  --log-color           Colored log output in console (stripped if redirected)
```

(continues on next page)

(continued from previous page)

```
  --log-time            Include timestamp in console log
  -h, --help            show this help message and exit

client commands:
  computer info  get information of an computer
  computer list  request an instance and get status and parameters of instance
  computer token  get token for setup a computer
  configure client  configure slapos client with an existing account
  console        open python console with slap library imported
  remove         remove a Software from a node
  request        request an instance and get status and parameters of instance
  service info   get status, software_release and parameters of an instance
  service list   request an instance and get status and parameters of instance
  supply         supply a Software to a node

node commands:
  node bang      request update on all partitions
  node boot      Test network and invoke simple format and bang (Use on Linux startup)
  node collect   Collect system consumption and data and store.
  node format    create users, partitions and network configuration
  node instance  run instance deployment
  node promise   run only promises to test the parition state
  node prune     Clean up unused shared slapos.recipe.cmmi parts.
  node register  Register a new computer on SlapOS Master.
  node report    run instance reports and garbage collection
  node restart   alias for 'node supervisorctl restart'
  node software  run software installation/deletion
  node start     alias for 'node supervisorctl start'
  node status    alias for 'node supervisorctl status'
  node stop      alias for 'node supervisorctl stop'
  node supervisorctl  open supervisor console, for process management
  node supervisord  launch, if not already running, supervisor daemon
  node tail      alias for 'node supervisorctl tail'

other commands:
  cache lookup   perform a query to the networkcache
  cache source   perform a query to the networkcache.
  complete       Generate shell completions.
  configure local  Configure a slapos node, from scratch to ready-to-use, using␣
↪slapproxy.
  help           print detailed help for another command
  proxy show     display proxy instances and parameters
  proxy start    minimalist, stand-alone SlapOS Master
```

The `-q` and `-v` options control the verbosity of console output (`-v`: DEBUG, default: INFO, `-q`: WARNING).

Output to a logfile is not affected, and is the same as `-v`.

1.3 SlapOS Client commands

These commands are used by clients (as human beings or programs) to manage their own instances.

1.3.1 configure client

configure slapos client with an existing account

```
usage: slapos configure client [-h] [--cfg CFG] [--master-url MASTER_URL]
                               [--master-url-web MASTER_URL_WEB]
                               [--token TOKEN]

configure slapos client with an existing account

optional arguments:
  -h, --help            show this help message and exit
  --cfg CFG             SlapOS configuration file (default:
                        $SLAPOS_CLIENT_CONFIGURATION or ~/.slapos/slapos-
                        client.cfg)
  --master-url MASTER_URL
                        URL of SlapOS Master REST API (default:
                        https://slap.vifib.com)
  --master-url-web MASTER_URL_WEB
                        URL of SlapOS Master webservice to register
                        certificates (default: https://panel.rapid.space)
  --token TOKEN         SlapOS 'credential security' authentication token (if
                        this parameter is omitted, the token will be prompted
                        during configuration)
```

1.3.2 configure local

configure slapos node, slapos proxy and slapos client to work in a self-contained, autonomous way, without any external slapos master.

```
usage: slapos configure local [-h] [--cfg CFG]
                              [--interface-name INTERFACE_NAME]
                              [--partition-number PARTITION_NUMBER]
                              [--ipv4-local-network IPV4_LOCAL_NETWORK]
                              [--daemon-listen-ip DAEMON_LISTEN_IP]
                              [--daemon-listen-port DAEMON_LISTEN_PORT]
                              [--slapos-instance-root SLAPOS_INSTANCE_ROOT]
                              [--slapos-software-root SLAPOS_SOFTWARE_ROOT]
                              [--slapos-buildout-directory SLAPOS_BUILDOUT_DIRECTORY]
                              [--slapos-configuration-directory SLAPOS_CONFIGURATION_
↪DIRECTORY]

Configure a slapos node, from scratch to ready-to-use, using slapproxy.

optional arguments:
  -h, --help            show this help message and exit
  --cfg CFG             SlapOS configuration file (default:
                        $SLAPOS_CONFIGURATION or /etc/opt/slapos/slapos.cfg)
  --interface-name INTERFACE_NAME
                        Primary network interface. IP of Partitions will be
                        added to it (default: lo)
  --partition-number PARTITION_NUMBER
                        Number of partitions to create in the SlapOS Node
                        (default: 20)
  --ipv4-local-network IPV4_LOCAL_NETWORK
                        Subnetwork used to assign local IPv4 addresses. It
```

(continues on next page)

(continued from previous page)

```
                        should be a not used network in order to avoid
                        conflicts (default: 10.0.0.0/16)
  --daemon-listen-ip DAEMON_LISTEN_IP
                        Listening IP of the "slapproxy" daemon (default:
                        127.0.0.1)
  --daemon-listen-port DAEMON_LISTEN_PORT
                        Listening port of the "slapproxy" daemon (default:
                        8080)
  --slapos-instance-root SLAPOS_INSTANCE_ROOT
                        Target location of the SlapOS configuration directory
                        (default: /srv/slapgrid)
  --slapos-software-root SLAPOS_SOFTWARE_ROOT
                        Target location of the SlapOS configuration directory
                        (default: /opt/slapgrid)
  --slapos-buildout-directory SLAPOS_BUILDOUT_DIRECTORY
                        Target location of the SlapOS configuration directory
                        (default: /opt/slapos)
  --slapos-configuration-directory SLAPOS_CONFIGURATION_DIRECTORY
                        Target location of the SlapOS configuration directory
                        (default: /etc/opt/slapos)
```

1.3.3 request

```
usage: slapos request [-h] [--cfg CFG] [--node NODE [NODE ...]] [--type TYPE]
                      [--state STATE] [--slave]
                      [--parameters PARAMETERS [PARAMETERS ...]]
                      reference software_url

request an instance and get status and parameters of instance

positional arguments:
  reference             Your instance reference
  software_url          Your software url

optional arguments:
  -h, --help            show this help message and exit
  --cfg CFG             SlapOS configuration file (default:
                        $SLAPOS_CLIENT_CONFIGURATION or ~/.slapos/slapos-
                        client.cfg)
  --node NODE [NODE ...]
                        Node request option 'option1=value1 option2=value2'
                        (i.e. computer_guid=COMP-1234)
  --type TYPE           Software type to be requested
  --state STATE         State of the requested instance
  --slave               Ask for a slave instance
  --parameters PARAMETERS [PARAMETERS ...]
                        Give your configuration 'option1=value1
                        option2=value2'
```

Request the deployment of a service (instance).

Examples

- Request a wordpress instance named “mybeautifulinstance” on any available machine (either owned by user, public, or shared by other user to current user):

```
$ slapos request mybeautifulinstance wordpress
```

- Request a wordpress instance named “My Beautiful Instance” on Node named “COMP-12345”:

```
$ slapos request "My Beautiful Instance" wordpress --node computer_guid=COMP-12345
```

- Request a kvm instance named “mykvm” on Node named “COMP-12345”, specifying nbd-host and nbd-ip parameters:

```
$ slapos request mykvm kvm --node computer_guid=COMP-12345 --parameters \
    nbd-host=debian.nbd.vifib.net nbd-port=1024
```

- Request a kvm instance specifying the full URL, with default settings:

```
$ slapos request mykvm \
    https://lab.node.vifib.com/nexedi/slapos/raw/1.0.56/software/kvm/software.cfg
```

- Request a kvm instance specifying an alias from SlapOS Master, with default settings:

```
$ slapos request mykvm product.kvm
```

In these examples, `wordpress` and `kvm` are aliases for the full URL, and are defined in `slapos-client.cfg`.

1.3.4 info

```
2021-05-03 16:58:31 cliff.commandmanager[187] CRITICAL slapos: the command 'info'␣
↪does not exist or is not yet implemented.

Available commands: 'cache lookup', 'cache source', 'complete', 'computer info',
↪'computer list', 'computer token', 'configure client', 'configure local', 'console',
↪ 'help', 'node bang', 'node boot', 'node collect', 'node format', 'node instance',
↪'node promise', 'node prune', 'node register', 'node report', 'node restart', 'node␣
↪software', 'node start', 'node status', 'node stop', 'node supervisorctl', 'node␣
↪supervisord', 'node tail', 'proxy show', 'proxy start', 'remove', 'request',
↪'service info', 'service list', 'supply'

Please find documentation and forum at http://community.slapos.org
Please also make sure that the SlapOS Node package is up to date.
```

Get informations of specified instance, like connection parameters, Software Release. Return an error if instance does not exist for the current user.

Examples:

- Ask informations about an instance named “My Service”

  $ slapos info “My Service”

1.3.5 list

```
2021-05-03 16:58:31 cliff.commandmanager[188] CRITICAL slapos: the command 'list'␣
↪does not exist or is not yet implemented.

Available commands: 'cache lookup', 'cache source', 'complete', 'computer info',
↪'computer list', 'computer token', 'configure client', 'configure local', 'console',
↪ 'help', 'node bang', 'node boot', 'node collect', 'node format', 'node instance',
↪'node promise', 'node prune', 'node register', 'node report', 'node restart', 'node␣
↪software', 'node start', 'node status', 'node stop', 'node supervisorctl', 'node␣
↪supervisord', 'node tail', 'proxy show', 'proxy start', 'remove', 'request',
↪'service info', 'service list', 'supply'
```

(continues on next page)

(continued from previous page)

```
Please find documentation and forum at http://community.slapos.org
Please also make sure that the SlapOS Node package is up to date.
```

List all deployed services owned by current user. From SlapOS Master point of view, it should return the list of all non-destroyed Hosting Subscriptions.

1.3.6 supply

```
usage: slapos supply [-h] [--cfg CFG] software_url node

supply a Software to a node

positional arguments:
  software_url  Your software url
  node          Target node

optional arguments:
  -h, --help    show this help message and exit
  --cfg CFG     SlapOS configuration file (default:
                $SLAPOS_CLIENT_CONFIGURATION or ~/.slapos/slapos-client.cfg)
```

Ask installation of a software on a specific node or group of nodes. Nodes will then be ready to accept instances of specified software.

Examples

- Ask installation of kvm Software Release on COMP-12345:

```
$ slapos supply kvm COMP-12345
```

- Ask installation of kvm Software Relase on COMP-12345, using alias from SlapOS Master:

```
$ slapos supply product.kvm COMP-12345
```

In this example, `kvm` is an alias for the full URL, and is defined in `slapos-client.cfg`.

1.3.7 remove

```
usage: slapos remove [-h] [--cfg CFG] software_url node

remove a Software from a node

positional arguments:
  software_url  Your software url
  node          Target node

optional arguments:
  -h, --help    show this help message and exit
  --cfg CFG     SlapOS configuration file (default:
                $SLAPOS_CLIENT_CONFIGURATION or ~/.slapos/slapos-client.cfg)
```

Ask removal of a software from a specific node or group of nodes. Existing instances won't work anymore.

Examples

- Ask removal of kvm Software Release on COMP-12345:

```
$ slapos remove kvm COMP-12345
```

In this example, `kvm` is an alias for the full URL, and is defined in `slapos-client.cfg`.

1.3.8 console

```
usage: slapos console [-h] [--cfg CFG] [-u MASTER_URL] [-k KEY_FILE]
                      [-c CERT_FILE] [-i] [-b] [-p]
                      [script_file]

open python console with slap library imported

You can play with the global "slap" object and
with the global "request" method.

examples :
>>> # Request instance
>>> request(kvm, "myuniquekvm")
>>> # Request software installation on owned computer
>>> supply(kvm, "mycomputer")
>>> # Fetch instance informations on already launched instance
>>> request(kvm, "myuniquekvm").getConnectionParameter("url")

positional arguments:
  script_file           Script to run

optional arguments:
  -h, --help            show this help message and exit
  --cfg CFG             SlapOS configuration file (default:
                        $SLAPOS_CLIENT_CONFIGURATION or ~/.slapos/slapos-
                        client.cfg)
  -u MASTER_URL, --master_url MASTER_URL
                        Url of SlapOS Master to use
  -k KEY_FILE, --key_file KEY_FILE
                        SSL Authorisation key file
  -c CERT_FILE, --cert_file CERT_FILE
                        SSL Authorisation certificate file
  -i, --ipython         Use IPython shell if available (default)
  -b, --bpython         Use BPython shell if available
  -p, --python          Use plain Python shell
```

1.4 SlapOS Node commands

This group of commands is used to control the current SlapOS Node. They are only useful to Node administrators.

1.4.1 node, node status

These are both aliases for `node supervisorctl status`. It displays the status of the node, also running the supervisor daemon if needed.

```
usage: slapos node supervisorctl [-h] [--cfg CFG] ...

open supervisor console, for process management

positional arguments:
  supervisor_args  parameters passed to supervisorctl

optional arguments:
  -h, --help       show this help message and exit
  --cfg CFG        SlapOS configuration file (default: $SLAPOS_CONFIGURATION
                   or /etc/opt/slapos/slapos.cfg)
```

1.4.2 node register

```
usage: slapos node register [-h] [--interface-name INTERFACE_NAME]
                            [--master-url MASTER_URL]
                            [--master-url-web MASTER_URL_WEB]
                            [--partition-number PARTITION_NUMBER]
                            [--ipv4-local-network IPV4_LOCAL_NETWORK]
                            [--ipv6-interface IPV6_INTERFACE] [--login-auth]
                            [--login LOGIN] [--password PASSWORD]
                            [--token TOKEN] [--create-tap] [--dry-run]
                            node_name

Register a new computer on SlapOS Master.

This command will generate everything you need for run your slapos node,
The files at /etc/opt/slapos (by default):

  - /etc/opt/slapos/slapos.cfg: The configuration of your SlapOS Node
  - /etc/opt/slapos/ssl/certificate : Your server SSL Cetificate
  - /etc/opt/slapos/ssl/key: Your server SSL Private Key

positional arguments:
  node_name             Chosen title for the node

optional arguments:
  -h, --help            show this help message and exit
  --interface-name INTERFACE_NAME
                        Primary network interface. IP of Partitions will be
                        added to it (default: eth0)
  --master-url MASTER_URL
                        URL of SlapOS Master REST API (default:
                        https://slap.vifib.com)
  --master-url-web MASTER_URL_WEB
                        URL of SlapOS Master webservice to register
                        certificates (default: https://panel.rapid.space)
  --partition-number PARTITION_NUMBER
                        Number of partitions to create in the SlapOS Node
                        (default: 10)
  --ipv4-local-network IPV4_LOCAL_NETWORK
                        Subnetwork used to assign local IPv4 addresses. It
                        should be a not used network in order to avoid
                        conflicts (default: 10.0.0.0/16)
  --ipv6-interface IPV6_INTERFACE
```

(continues on next page)

(continued from previous page)

```
                        Interface name to get ipv6
  --login-auth          Force login and password authentication
  --login LOGIN         Your SlapOS Master login. Asks it interactively, then
                        password.
  --password PASSWORD   Your SlapOS Master password. If not provided, asks it
                        interactively. NOTE: giving password as parameter
                        should be avoided for security reasons.
  --token TOKEN         SlapOS 'computer security' authentication token
  --create-tap, -t      Will trigger creation of one virtual "tap" interface
                        per Partition and attach it to primary interface.
                        Requires primary interface to be a bridge. Needed to
                        host virtual machines (default: False)
  --dry-run, -n         Simulate the execution steps (default: False)
```

This will register the current node, and generate the SlapOS configuration file.

The command requires an authentication token, either provided as an argument, or given at the interactive prompt. Go to the SlapOS Master web page, click `Servers` and them `Token`. A token is valid for a single `node register` command and will expire after one day.

If the Node is already registered (`slapos.cfg` and certificate are already present), the command issues a warning, backups the original configuration and creates a new one.

Notes:

- "IPv6 interface" and "create tap" won't be put at all in the SlapOS Node configuration file if not explicitly written.

Examples

- Register computer named "mycomputer" to SlapOS Master:

```
$ slapos node register mycomputer
```

- Register computer named "mycomputer" to SlapOS Master using br0 as primary interface, tap0 as IPv6 interface and different local ipv4 subnet:

```
$ slapos node register mycomputer --interface-name br0 --ipv6-interface tap0 \
    --ipv4-local-network 11.0.0.0/16
```

- Register computer named "mycomputer" to another SlapOS master accessible via https://www.myownslaposmaster.com, and SLAP webservice accessible via https://slap.myownslaposmaster.com (note that this address should be the "slap" webservice URL, not web URL):

```
$ slapos node register mycomputer --master-url https://slap.myownslaposmaster.com␣
↪\
    --master-url-web https://www.myownslaposmaster.com
```

- Register computer named "mycomputer" to SlapOS Master, and ask to create tap interface to be able to host KVMs:

```
$ slapos node register mycomputer --create-tap
```

1.4.3 node software

```
usage: slapos node software [-h] [--cfg CFG] [--instance-root INSTANCE_ROOT]
                            [--software-root SOFTWARE_ROOT]
                            [--master-url MASTER_URL]
                            [--computer-id COMPUTER_ID]
                            [--supervisord-socket SUPERVISORD_SOCKET]
                            [--supervisord-configuration-path SUPERVISORD_
↪CONFIGURATION_PATH]
                            [--buildout BUILDOUT] [--pidfile PIDFILE]
                            [--key_file KEY_FILE] [--cert_file CERT_FILE]
                            [--signature_private_key_file SIGNATURE_PRIVATE_KEY_FILE]
                            [--master_ca_file MASTER_CA_FILE]
                            [--certificate_repository_path CERTIFICATE_REPOSITORY_
↪PATH]
                            [--maximum-periodicity MAXIMUM_PERIODICITY]
                            [--promise-timeout PROMISE_TIMEOUT] [--now]
                            [--maximal_delay MAXIMAL_DELAY] [--buildout-debug]
                            [--all | --only-sr ONLY_SR]

run software installation/deletion

optional arguments:
  -h, --help            show this help message and exit
  --cfg CFG             SlapOS configuration file (default:
                        $SLAPOS_CONFIGURATION or /etc/opt/slapos/slapos.cfg)
  --instance-root INSTANCE_ROOT
                        The instance root directory location.
  --software-root SOFTWARE_ROOT
                        The software_root directory location.
  --master-url MASTER_URL
                        The master server URL. Mandatory.
  --computer-id COMPUTER_ID
                        The computer id defined in the server.
  --supervisord-socket SUPERVISORD_SOCKET
                        The socket supervisor will use.
  --supervisord-configuration-path SUPERVISORD_CONFIGURATION_PATH
                        The location where supervisord configuration will be
                        stored.
  --buildout BUILDOUT   Location of buildout binary.
  --pidfile PIDFILE     The location where pidfile will be created. Can be
                        provided by configuration file as option
                        `pidfile_software` in slapos section, otherwise
                        defaults to /opt/slapos/slapgrid-sr.pid
  --key_file KEY_FILE   SSL Authorisation key file.
  --cert_file CERT_FILE
                        SSL Authorisation certificate file.
  --signature_private_key_file SIGNATURE_PRIVATE_KEY_FILE
                        Signature private key file.
  --master_ca_file MASTER_CA_FILE
                        Root certificate of SlapOS master key.
  --certificate_repository_path CERTIFICATE_REPOSITORY_PATH
                        Path to directory where downloaded certificates would
                        be stored.
  --maximum-periodicity MAXIMUM_PERIODICITY
                        Periodicity at which buildout should be run in
                        instance.
```

(continues on next page)

(continued from previous page)

```
  --promise-timeout PROMISE_TIMEOUT
                        Promise timeout in seconds (default: 20)
  --now                 Launch slapgrid without delay. Default behavior.
  --maximal_delay MAXIMAL_DELAY
                        Deprecated. Will only work from configuration file in
                        the future.
  --buildout-debug      Run buildout in debug mode (with -D command line
                        switch)
  --all                 Process all Software Releases, even if already
                        installed.
  --only-sr ONLY_SR, --only ONLY_SR
                        Force the update of a single software release (can be
                        full URL or MD5 hash), even if is already installed.
                        This option will make all other sofware releases be
                        ignored.
```

Return values:

(among other standard Python return values)

- 0 Everything went fine.
- 1 At least one software was not correctly installed.

1.4.4 node instance

```
usage: slapos node instance [-h] [--cfg CFG] [--instance-root INSTANCE_ROOT]
                            [--software-root SOFTWARE_ROOT]
                            [--master-url MASTER_URL]
                            [--computer-id COMPUTER_ID]
                            [--supervisord-socket SUPERVISORD_SOCKET]
                            [--supervisord-configuration-path SUPERVISORD_
↪CONFIGURATION_PATH]
                            [--buildout BUILDOUT] [--pidfile PIDFILE]
                            [--key_file KEY_FILE] [--cert_file CERT_FILE]
                            [--signature_private_key_file SIGNATURE_PRIVATE_KEY_FILE]
                            [--master_ca_file MASTER_CA_FILE]
                            [--certificate_repository_path CERTIFICATE_REPOSITORY_
↪PATH]
                            [--maximum-periodicity MAXIMUM_PERIODICITY]
                            [--promise-timeout PROMISE_TIMEOUT] [--now]
                            [--maximal_delay MAXIMAL_DELAY] [--buildout-debug]
                            [--all | --only-cp ONLY_CP]

run instance deployment

optional arguments:
  -h, --help            show this help message and exit
  --cfg CFG             SlapOS configuration file (default:
                        $SLAPOS_CONFIGURATION or /etc/opt/slapos/slapos.cfg)
  --instance-root INSTANCE_ROOT
                        The instance root directory location.
  --software-root SOFTWARE_ROOT
                        The software_root directory location.
```

(continues on next page)

(continued from previous page)

```
  --master-url MASTER_URL
                        The master server URL. Mandatory.
  --computer-id COMPUTER_ID
                        The computer id defined in the server.
  --supervisord-socket SUPERVISORD_SOCKET
                        The socket supervisor will use.
  --supervisord-configuration-path SUPERVISORD_CONFIGURATION_PATH
                        The location where supervisord configuration will be
                        stored.
  --buildout BUILDOUT   Location of buildout binary.
  --pidfile PIDFILE     The location where pidfile will be created. Can be
                        provided by configuration file as option
                        `pidfile_instance` in slapos section, otherwise
                        defaults to /opt/slapos/slapgrid-cp.pid
  --key_file KEY_FILE   SSL Authorisation key file.
  --cert_file CERT_FILE
                        SSL Authorisation certificate file.
  --signature_private_key_file SIGNATURE_PRIVATE_KEY_FILE
                        Signature private key file.
  --master_ca_file MASTER_CA_FILE
                        Root certificate of SlapOS master key.
  --certificate_repository_path CERTIFICATE_REPOSITORY_PATH
                        Path to directory where downloaded certificates would
                        be stored.
  --maximum-periodicity MAXIMUM_PERIODICITY
                        Periodicity at which buildout should be run in
                        instance.
  --promise-timeout PROMISE_TIMEOUT
                        Promise timeout in seconds (default: 20)
  --now                 Launch slapgrid without delay. Default behavior.
  --maximal_delay MAXIMAL_DELAY
                        Deprecated. Will only work from configuration file in
                        the future.
  --buildout-debug      Run buildout in debug mode (with -D command line
                        switch)
  --all                 Process all Computer Partitions.
  --only-cp ONLY_CP, --only ONLY_CP
                        Update a single or a list of computer partitions
                        (ie.:slappartX, slappartY), this option will make all
                        other computer partitions be ignored.
```

Return values:

(among other standard Python return values)

- 0 Everything went fine.
- 1 At least one instance was not correctly processed.
- 2 At least one promise has failed.

1.4.5 node collect

```
usage: slapos node collect [-h] [--cfg CFG]
```

(continues on next page)

(continued from previous page)

```
Collect system consumption and data and store.

optional arguments:
  -h, --help  show this help message and exit
  --cfg CFG   SlapOS configuration file (default: $SLAPOS_CONFIGURATION or
              /etc/opt/slapos/slapos.cfg)
```

Return values:

- 0 Everything went fine.
- 1 Fail to collect computer information.

1.4.6 node report

```
usage: slapos node report [-h] [--cfg CFG] [--instance-root INSTANCE_ROOT]
                          [--software-root SOFTWARE_ROOT]
                          [--master-url MASTER_URL]
                          [--computer-id COMPUTER_ID]
                          [--supervisord-socket SUPERVISORD_SOCKET]
                          [--supervisord-configuration-path SUPERVISORD_CONFIGURATION_
↪PATH]
                          [--buildout BUILDOUT] [--pidfile PIDFILE]
                          [--key_file KEY_FILE] [--cert_file CERT_FILE]
                          [--signature_private_key_file SIGNATURE_PRIVATE_KEY_FILE]
                          [--master_ca_file MASTER_CA_FILE]
                          [--certificate_repository_path CERTIFICATE_REPOSITORY_PATH]
                          [--maximum-periodicity MAXIMUM_PERIODICITY]
                          [--promise-timeout PROMISE_TIMEOUT] [--now]
                          [--maximal_delay MAXIMAL_DELAY]

run instance reports and garbage collection

optional arguments:
  -h, --help            show this help message and exit
  --cfg CFG             SlapOS configuration file (default:
                        $SLAPOS_CONFIGURATION or /etc/opt/slapos/slapos.cfg)
  --instance-root INSTANCE_ROOT
                        The instance root directory location.
  --software-root SOFTWARE_ROOT
                        The software_root directory location.
  --master-url MASTER_URL
                        The master server URL. Mandatory.
  --computer-id COMPUTER_ID
                        The computer id defined in the server.
  --supervisord-socket SUPERVISORD_SOCKET
                        The socket supervisor will use.
  --supervisord-configuration-path SUPERVISORD_CONFIGURATION_PATH
                        The location where supervisord configuration will be
                        stored.
  --buildout BUILDOUT   Location of buildout binary.
  --pidfile PIDFILE     The location where pidfile will be created. Can be
                        provided by configuration file as option
                        `pidfile_report` in slapos section, otherwise defaults
```

(continues on next page)

(continued from previous page)

```
                        to /opt/slapos/slapgrid-ur.pid
  --key_file KEY_FILE   SSL Authorisation key file.
  --cert_file CERT_FILE
                        SSL Authorisation certificate file.
  --signature_private_key_file SIGNATURE_PRIVATE_KEY_FILE
                        Signature private key file.
  --master_ca_file MASTER_CA_FILE
                        Root certificate of SlapOS master key.
  --certificate_repository_path CERTIFICATE_REPOSITORY_PATH
                        Path to directory where downloaded certificates would
                        be stored.
  --maximum-periodicity MAXIMUM_PERIODICITY
                        Periodicity at which buildout should be run in
                        instance.
  --promise-timeout PROMISE_TIMEOUT
                        Promise timeout in seconds (default: 20)
  --now                 Launch slapgrid without delay. Default behavior.
  --maximal_delay MAXIMAL_DELAY
                        Deprecated. Will only work from configuration file in
                        the future.
```

Return values:

(among other standard Python return values)

- 0 Everything went fine.
- 1 At least one instance hasn't correctly been processed.

1.4.7 node start|stop|restart|tail|status

```
usage: slapos node <start|stop|restart|tail|status> [-h] [--cfg CFG] <instance>
↪:[process]

Start/Stop/Restart/Show stdout/stderr of instance and/or process.

optional arguments:
 -h, --help       show this help message and exit
 --cfg CFG        SlapOS configuration file (default: $SLAPOS_CONFIGURATION
                  or /etc/opt/slapos/slapos.cfg)
```

Examples

- Start all processes of slappart3:

```
$ slapos node start slappart3:
```

- Stop only apache in slappart1:

```
$ slapos node stop slappart1:apache
```

- Show stdout/stderr of mysqld in slappart2:

```
$ slapos node tail slappart2:mysqld
```

1.4.8 node supervisorctl

```
usage: slapos node supervisorctl [-h] [--cfg CFG] ...

open supervisor console, for process management

positional arguments:
  supervisor_args  parameters passed to supervisorctl

optional arguments:
  -h, --help       show this help message and exit
  --cfg CFG        SlapOS configuration file (default: $SLAPOS_CONFIGURATION
                   or /etc/opt/slapos/slapos.cfg)
```

1.4.9 node supervisord

```
usage: slapos node supervisord [-h] [--cfg CFG] [-n]

launch, if not already running, supervisor daemon

optional arguments:
  -h, --help      show this help message and exit
  --cfg CFG       SlapOS configuration file (default: $SLAPOS_CONFIGURATION or
                  /etc/opt/slapos/slapos.cfg)
  -n, --nodaemon  Do not daemonize supervisord
```

1.5 SlapOS Miscellaneous commands

1.5.1 configure client

```
usage: slapos configure client [-h] [--cfg CFG] [--master-url MASTER_URL]
                               [--master-url-web MASTER_URL_WEB]
                               [--token TOKEN]

configure slapos client with an existing account

optional arguments:
  -h, --help            show this help message and exit
  --cfg CFG             SlapOS configuration file (default:
                        $SLAPOS_CLIENT_CONFIGURATION or ~/.slapos/slapos-
                        client.cfg)
  --master-url MASTER_URL
                        URL of SlapOS Master REST API (default:
                        https://slap.vifib.com)
  --master-url-web MASTER_URL_WEB
                        URL of SlapOS Master webservice to register
                        certificates (default: https://panel.rapid.space)
  --token TOKEN         SlapOS 'credential security' authentication token (if
                        this parameter is omitted, the token will be prompted
                        during configuration)
```

This creates a client configuration file, and downloads a certificate + key pair from the SlapOS Master. They will be used for all the "slapos client" commands.

The command requires an authentication token, either provided as an argument, or given at the interactive prompt.

Go to the SlapOS Master web page, click `Account`, then `Token`. A token is valid for a single `configure client` command and will expire after one day.

1.5.2 cache lookup

```
usage: slapos cache lookup [-h] [--cfg CFG] software_url

perform a query to the networkcache
You can provide either a complete URL to the software release,
or a corresponding MD5 hash value.

The command will report which OS distribution/version have a binary
cache of the software release, and which ones are compatible
with the OS you are currently running.

positional arguments:
  software_url  Your software url or MD5 hash

optional arguments:
  -h, --help    show this help message and exit
  --cfg CFG     SlapOS configuration file (default: $SLAPOS_CONFIGURATION or
                /etc/opt/slapos/slapos.cfg)
```

Examples

- See if the wordpress Software Release is available in precompiled format for our distribution:

```
$ slapos cache lookup http://git.erp5.org/gitweb/slapos.git/blob_plain/refs/tags/
↪slapos-0.156:/software/kvm/software.cfg
Software URL: http://git.erp5.org/gitweb/slapos.git/blob_plain/refs/tags/slapos-0.
↪156:/software/kvm/software.cfg
MD5:          4410088e11f370503e9d78db4cfa4ec4
-------------
Available for:
distribution     |   version    |       id       | compatible?
-----------------+--------------+----------------+-------------
CentOS           |          6.3 |     Final      | no
Fedora           |           17 | Beefy Miracle  | no
Ubuntu           |        12.04 |    precise     | yes
debian           |        6.0.6 |                | no
debian           |          7.0 |                | no
```

You can also use the corresponding hash value in place of the URL.

CHAPTER 2

slap interface documentation

interface slapos.slap.interface.slap.**IException**
Classes which implement IException are used to report errors.

interface slapos.slap.interface.slap.**INotFoundError**
Extends: *slapos.slap.interface.slap.IException*

Classes which implement INotFoundError are used to report missing information on the slap server.

interface slapos.slap.interface.slap.**IRequester**
Classes which implement IRequester can request software instance to the slapgrid server.

getInformation(*partition_reference*)
Get information about an existing instance. If it is called from a Computer Partition, get information about Software Instance of the instance tree.

partition_reference – local reference of the instance used by the recipe to identify the instances.

request(*software_release*, *software_type*, *partition_reference*, *shared=False*, *partition_parameter_kw=None*, *filter_kw=None*)
Request software release instantiation to slapgrid server.

Returns a new computer partition document, where this software release will be installed.

software_release – URI of the software release which has to be instantiated

software_type – type of component provided by software_release

partition_reference – local reference of the instance used by the recipe to identify the instances.

shared – boolean to use a shared service

partition_parameter_kw – dictionary of parameter used to fill the parameter dictionary of newly created partition.

filter_kw – dictionary of filtering parameter to select the requested

computer partition.

computer_guid - computer of the requested partition partition_type - virtio, slave, full, limited port - port provided by the requested partition

Example: request('http://example.com/foo/bar', 'typeA', 'mysql_1')

interface slapos.slap.interface.slap.**ISoftwareRelease**
Software release interface specification

available()
Notify (to the slapgrid server) that the software release is available.

building()
Notify (to the slapgrid server) that the software release is not available and under creation.

getState()
Returns a string representing the expected state of the software installation.

The result can be: available, destroyed

getURI()
Returns a string representing the URI of the software release.

getComputerId()
Returns a string representing the identifier of the computer where the SR is installed.

destroyed()
Notify (to the slapgrid server) that the software installation has been correctly destroyed.

error(*error_log*)
Notify (to the slapgrid server) that the software installation is not available and reports an error.

error_log – a text describing the error It can be a traceback for example.

interface slapos.slap.interface.slap.**IComputerPartition**
Extends: *slapos.slap.interface.slap.IRequester*

Computer Partition interface specification

Classes which implement IComputerPartition can propagate the computer partition state to the SLAPGRID server and request new computer partition creation.

rename(*partition_reference*, *slave_reference=None*)
Change the partition reference of a partition

partition_reference – new local reference of the instance used by the recipe to identify the instances.

slave_reference – current reference of the slave instance to modify

setComputerPartitionRelatedInstanceList(*instance_reference_list*)
Set relation between this Instance and all his children.

instance_reference_list – list of instances requested by this Computer Partition.

getInstanceGuid()
Returns a string representing the unique identifier of the instance inside the slapgrid server.

getState()
Returns a string representing the expected state of the computer partition.

The result can be: started, stopped, destroyed

setUsage(*usage_log*)
Associate a usage log to the computer partition. This method does not report the usage to the slapgrid server. See IComputer.report.

usage_log – a text describing the computer partition usage. It can be an XML for example.

error(*error_log*)
Notify (to the slapgrid server) that the software instance is not available and reports an error.

error_log – a text describing the error It can be a traceback for example.

getInstanceParameterDict()
Returns a dictionary of instance parameters.

The contained values can be used to fill the software instantiation profile.

getStatus()
Returns a dictionary containing the latest status of the computer partition. The dictionary keys are:

user – user who reported the latest status created_at – date of the status text – message log of the status

getId()
Returns a string representing the identifier of the computer partition inside the slapgrid server.

destroyed()
Notify (to the slapgrid server) that the software instance has been correctly destroyed.

getAccessStatus()
Get latest computer partition Access message state

getConnectionParameterDict()
Returns a dictionary of connection parameters.

The contained values are connection parameters of a compute partition.

started()
Notify (to the slapgrid server) that the software instance is available and started.

getType()
Returns the Software Type of the instance.

getCertificate()
Returns a dictionary containing the authentication certificates associated to the computer partition. The dictionary keys are:

key – value is a SSL key certificate – value is a SSL certificate

Raise an INotFoundError if no software release is associated.

bang(*log*)
Report a problem detected on a computer partition. This will trigger the re-instantiation of all partitions in the instance tree.

log – a text explaining why the method was called

getInstanceParameter(*key*)
Returns the instance parameter associated to the key.

Raise an INotFoundError if no key is defined.

key – a string name of the parameter

setConnectionDict(*connection_dict*, *slave_reference=None*)
Store the connection parameters associated to a partition.

connection_dict – dictionary of parameter used to fill the connection dictionary of the partition.

slave_reference – current reference of the slave instance to modify

getSoftwareRelease()
Returns the software release associate to the computer partition.

Raise an INotFoundError if no software release is associated.

stopped()
Notify (to the slapgrid server) that the software instance is available and stopped.

getConnectionParameter(*key*)
Return the connection parameter associate to the key.

Raise an INotFoundError if no key is defined.

key – a string name of the parameter

getFullHostingIpAddressList()
Returns a dictionary containing the latest status of the computer partition.

interface slapos.slap.interface.slap.**IComputer**
Computer interface specification

Classes which implement IComputer can fetch information from the slapgrid server to know which Software Releases and Software Instances have to be installed.

getInformation()
Get information about current computer. If it is called from a Computer, get information about itself.

updateConfiguration(*configuration_xml*)
Report the current computer configuration.

configuration_xml – computer XML description generated by slapformat

getStatus()
Returns a dictionary containing the latest status of the computer. The dictionary keys are:

user – user who reported the latest status created_at – date of the status text – message log of the status

reportUsage(*computer_partition_list*)
Report the computer usage to the slapgrid server. IComputerPartition.setUsage has to be called on each computer partition to define each usage.

computer_partition_list – a list of computer partition for which the usage needs to be reported.

bang(*log*)
Report a problem detected on a computer. This will trigger IComputerPartition.bang on all instances hosted by the Computer.

log – a text explaining why the method was called

getSoftwareReleaseList()
Returns the list of software release which has to be supplied by the computer.

Raise an INotFoundError if computer_guid doesn't exist.

generateCertificate()
Returns a dictionary containing the new certificate files for the computer. The dictionary keys are:

key – key file certificate – certificate file

Raise ValueError is another certificate is already valid.

revokeCertificate()
Revoke current computer certificate.

Raise ValueError is there is not valid certificate.

getComputerPartitionList()
Returns the list of configured computer partitions associated to this computer.

Raise an INotFoundError if computer_guid doesn't exist.

interface slapos.slap.interface.slap.**IOpenOrder**
Extends: *slapos.slap.interface.slap.IRequester*

Open Order interface specification

Classes which implement Open Order describe which kind of software instances is requested by a given client.

requestComputer(*computer_reference*)
Request a computer to slapgrid server.

Returns a new computer document.

computer_reference – local reference of the computer

interface slapos.slap.interface.slap.**ISupply**
Supply interface specification

Classes which implement Supply describe which kind of software releases a given client is ready to supply.

supply(*software_release*, *computer_guid=None*)
Tell that given client is ready to supply given software release

software_release – URI of the software release which has to be instantiated

computer_guid – the identifier of the computer inside the slapgrid server.

interface slapos.slap.interface.slap.**slap**
Initialize slap connection to the slapgrid server

Slapgrid server URL is defined during the slap library installation, as recipes should not use another server.

initializeConnection(*slapgrid_uri*, *authentification_key=None*)
Initialize the connection parameters to the slapgrid servers.

slapgrid_uri – URI the slapgrid server connector

authentification_key – string the authenticate the agent. (Yes, there's a typo in the argument name)

Example: https://slapos.server/slap_interface

registerSoftwareRelease(*software_release*)
Instantiate a software release in the slap library.

software_release – URI of the software release definition

registerToken()
Instantiate an token in the slap library.

registerSupply()
Instantiate a supply in the slap library.

getComputerDict()
Get the list of existing computer for the current user.

registerComputer(*computer_guid*)
Instantiate a computer in the slap library.

computer_guid – the identifier of the computer inside the slapgrid server.

registerComputerPartition(*computer_guid*, *partition_id*)
Instantiate a computer partition in the slap library.

computer_guid – the identifier of the computer inside the slapgrid server.

partition_id – the identifier of the computer partition inside the slapgrid server.

Raise an INotFoundError if computer_guid doesn't exist.

getOpenOrderDict()
Get the list of existing open orders (services) for the current user.

getSoftwareReleaseListFromSoftwareProduct(*software_product_reference*, *software_release_url*)
Get the list of Software Releases from a product or from another related Software Release, from a Software Product point of view.

registerOpenOrder()
Instantiate an open order in the slap library.

CHAPTER 3

Tio Format

3.1 What is TioFormat?

TIO is a data format used to provide informations about consumption, invoicing and state history.

This innformation is used by SlapOS Master to generated invoices and/or utilisation reports.

3.2 XSD

TioFormat XSD:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <!-- Define the XML Schema of a transaction -->
  <xs:element name="journal">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="transaction" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="title" type="xs:string" minOccurs="0"/>
              <xs:element name="start_date" type="xs:string"/>
              <xs:element name="stop_date" type="xs:string"/>
              <xs:element name="reference" type="xs:string"/>
              <xs:element name="currency" type="xs:string"/>
              <xs:element name="payment_mode" type="xs:string"/>
              <xs:element name="category" type="xs:string" minOccurs="0" maxOccurs=
↪"unbounded"/>
              <xs:element name="arrow" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
                  <xs:element name="source" type="xs:string" minOccurs="0"/>
```

(continues on next page)

(continued from previous page)

```
                <xs:element name="destination" type="xs:string" minOccurs="0"/>
                </xs:sequence>
                <xs:attribute name="type" use="required"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="movement" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                <xs:element name="resource" type="xs:string"/>
                <xs:element name="title" type="xs:string" minOccurs="0"/>
                <xs:element name="reference" type="xs:string" minOccurs="0"/>
                <xs:element name="quantity" type="xs:float"/>
                <xs:element name="price" type="xs:float"/>
                <xs:element name="VAT" type="xs:string"/>
                <xs:element name="category" type="xs:string" minOccurs="0"␣
↪maxOccurs="unbounded"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
          <xs:attribute name="type" use="required"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

3.3 Schema Components

3.3.1 Element: `journal`

Name	journal
Type	journal
Documentation	journal is the root element in the XML file

Schema Component Representation:

```
<xs:element name="journal">
```

3.3.2 Complex Type: `journal`

Name	journal
Documentation	Tio document contains transactions

Schema Component Representation:

```
<xs:complexType>
  <xs:sequence>
    <xs:element name="transaction" maxOccurs="unbounded">
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

3.3.3 Complex Type: `transaction`

Name	transaction
Documentation	transaction contains a title, a start_date, a stop_date, a reference, a currency, a payment mode, some categories, some arrows and a list of movement.

Schema Component Representation:

```
<xs:complexType>
  <xs:sequence>
    <xs:element name="title" type="xs:string" minOccurs="0"/>
    <xs:element name="start_date" type="xs:string"/>
    <xs:element name="stop_date" type="xs:string"/>
    <xs:element name="reference" type="xs:string"/>
    <xs:element name="currency" type="xs:string"/>
    <xs:element name="payment_mode" type="xs:string"/>
    <xs:element name="category" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="arrow" maxOccurs="unbounded">
    </xs:element>
    <xs:element name="movement" maxOccurs="unbounded">
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="type" use="required"/>
</xs:complexType>
```

3.3.4 Element: `title`

Name	title
Type	string
Documentation	title is the name of the transaction

Schema Component Representation:

```
<xs:element name="title" type="xs:string" minOccurs="0"/>
```

3.3.5 Element: `start_date`

Name	start_date
Type	string
Documentation	the date at which a service started

Schema Component Representation:

```
<xs:element name="start_date" type="xs:string"/>
```

3.3.6 Element: stop_date

Name	stop_date
Type	string
Documentation	the date at which a service was completed

Schema Component Representation:

```
<xs:element name="stop_date" type="xs:string"/>
```

3.3.7 Element: reference

Name	reference
Type	string
Documentation	absolute reference of the transaction

Schema Component Representation:

```
<xs:element name="reference" type="xs:string"/>
```

3.3.8 Element: currency

Name	currency
Type	string
Documentation	currency used in the transaction

Schema Component Representation:

```
<xs:element name="currency" type="xs:string"/>
```

3.3.9 Element: payment_mode

Name	payment_mode
Type	string
Documentation	payment mode of the transaction

Schema Component Representation:

```
<xs:element name="payment_mode" type="xs:string"/>
```

3.3.10 Element: `category`

Name	category
Type	string
Documentation	To add your own category section in the transaction

Schema Component Representation:

```
<xs:element name="category" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
```

3.3.11 Element: `arrow`

Name	arrow
Type	arrow
Documentation	represents who provided a service to somebody else

Schema Component Representation:

```
<xs:element name="arrow" maxOccurs="unbounded">
```

3.3.12 Complex Type: `arrow`

Name	arrow
Documentation	contains a source and a destination.

Schema Component Representation:

```
<xs:complexType>
  <xs:sequence>
  <xs:element name="source" type="xs:string" minOccurs="0"/>
  <xs:element name="destination" type="xs:string" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="type" use="required"/>
</xs:complexType>
```

3.3.13 Element: `source`

Name	source
Type	string
Documentation	who provided the service

Schema Component Representation:

```
<xs:element name="source" type="xs:string" minOccurs="0"/>
```

3.3.14 Element: destination

Name	destination
Type	string
Documentation	who received the service

Schema Component Representation:

```
<xs:element name="destination" type="xs:string" minOccurs="0"/>
```

3.3.15 Element: movement

Name	movement
Type	movement
Documentation	represents how much service exchanged in the transaction

Schema Component Representation:

```
<xs:element name="movement" maxOccurs="unbounded">
```

3.3.16 Complex Type: movement

Name	movement
Documentation	contains a resource, a title, a reference, a quantity, a price, a VAT and some categories

Schema Component Representation:

```
<xs:complexType>
  <xs:sequence>
  <xs:element name="resource" type="xs:string"/>
  <xs:element name="title" type="xs:string" minOccurs="0"/>
  <xs:element name="reference" type="xs:string" minOccurs="0"/>
  <xs:element name="quantity" type="xs:float"/>
  <xs:element name="price" type="xs:float"/>
  <xs:element name="VAT" type="xs:string"/>
  <xs:element name="category" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

3.3.17 Element: resource

Name	resource
Type	string
Documentation	represents the kind of service provided

Schema Component Representation:

```
<xs:element name="resource" type="xs:string"/>
```

3.3.18 Element: `quantity`

Name	quantity
Type	float
Documentation	represents the amount of service exchanged

Schema Component Representation:

```
<xs:element name="quantity" type="xs:string"/>
```

3.3.19 Element: `price`

Name	price
Type	float
Documentation	represents the price of service exchanged

Schema Component Representation:

```
<xs:element name="price" type="xs:string"/>
```

3.3.20 Element: `VAT`

Name	VAT
Type	string
Documentation	represents the VAT of service exchanged

Schema Component Representation:

```
<xs:element name="VAT" type="xs:string"/>
```

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Index

I

R

S

U