Zabbix-CLI

Version-1.7.0

Rafael Martinez Guerrero (University of Oslo)
E-mail: rafael@postgresql.org.es

Source: https://github.com/usit-gd/zabbix-cli

Contents

Introduction

Zabbix-cli is a terminal client for managing some Zabbix administration tasks via the zabbix-API.

The zabbix-cli code is distributed under the GNU General Public License 3 and it is written in Python. It has been developed and tested by members of the Department for IT Infrastructure at the Center for Information Technology at the University of Oslo, Norway.

Main features

- Terminal client
- Two execution modes available: Zabbix-CLI shell and commandline.
- 54 zabbix-CLI commands available.
- Multilevel configuration system.
- Possibility to define Bulk updates. Several performance improvements are used when running in bulk modus.
- Authentication-token, authentication-file and environment variables support for autologin.
- Support for plain, CSV and JSON output.
- Online help
- Written in Python.

Installation

System requirements

- Linux/Unix
- Python 2.6 or 2.7
- Python modules: request ipaddr

Before you install Zabbix-CLI you have to install the software needed by this tool

In systems using `yum`, e.g. Centos, RHEL, Fedora:

```
yum install python-requests python-ipaddr
```

In system using `apt-get`, e.g. Debian, Ubuntu:

```
apt-get install python-requests python-ipaddr
```

If you are going to install from source, you need to install also these packages: `python-dev(el), python-setuptools, git, make, python-docutils`

In systems using `yum`:

```
yum install python-devel python-setuptools git make python-docutils
```

In system using `apt-get`:

```
apt-get install python-dev python-setuptools git make python-docutils
```

Installing from source

The easiest way to install zabbix-cli from source is to download the latest stable release from GitHub https://github.com/usit-gd/zabbix-cli/releases in tar.gz or zip format.

You can also clone the official GitHub GIT repository and get the latest code from the master branch.

```
[root@server]# cd
[root@server]# git clone https://github.com/usit-gd/zabbix-cli.git

[root@server]# cd zabbix-cli
[root@server]# ./setup.py install
.....
```

NOTE: The code in the master branch can be unstable and with bugs

between releases. Use it at your own risk.

For stable code to be used in production use the source code distributed via the release section: https://github.com/usit-gd/zabbix-cli/releases

Installing via RPM packages

The University of Oslo will make available in the near future an official repository that can be used to install RPM packages via yum .

In the meantime download the latest RPM package for your distribution from https://github.com/usit-gd/zabbix-cli/releases and run this command:

```
# yum install <rpm_file>
```

Installing via Deb packages

Zabbix-CLI has been accepted into the official Debian package repository (unstable). It is available for Debian and Ubuntu systems. Check https://packages.qa.debian.org/z/zabbix-cli.html for details.

You can also download the latest DEB package from https://github.com/usit-gd/zabbix-cli/releases and install it with:

```
# dpkg -i <debian_package>
```

Configuration

Configuration file

Zabbix-CLI needs a configuration file to work. Until version 1.5.4 we supported a **singlelevel configuration system** with three possible locations for our configuration file:

1. Config file defined with `--config` or `-c` parameter when starting `zabbix-cli`
2. `$HOME/.zabbix-cli/zabbix-cli.conf`
3. `/etc/zabbix-cli/zabbix-cli.conf`

With the **singlelevel configuration system**, Zabbix-cli checked for a configuration file in these locations and in this order and used the first one that existed. This means that you could always override: 3) with 2) or 1), and 2) with 1).

From version 1.6.0, Zabbix-cli has started to use a **multilevel configuration system.**

This means thet we do not override entire configuration files but we merge all the defined configuration files in our system and use the parameter values defined in the configuration file with higher priority if a parameter is defined in more than one file.

The ordered list with the files with higher on top:

1. `/usr/share/zabbix-cli/zabbix-cli.fixed.conf`

2. `/etc/zabbix-cli/zabbix-cli.fixed.conf`
3. Configuration file defined with the parameter `-c` / `--config` when executing zabbix-cli
4. `$HOME/.zabbix-cli/zabbix-cli.conf`
5. `/etc/zabbix-cli/zabbix-cli.conf`
6. `/usr/share/zabbix-cli/zabbix-cli.conf`

With this implementation:

- Local configuration will be kept during upgrades.
- The local configuration is separate from the package defaults.
- Several actors will be allow to have their own files.
- It is possible to provide package, host and user defaults, as well as locking down features on a host, package level.
- Always well known where the admin made his changes

A default configuration file can be found in `/usr/share/zabbix-cli/zabbix-cli.conf` or `etc/zabbix-cli.conf` in the source code.

The easiest way to configurate your client will be running this command to create your own `$HOME/.zabbix-cli/zabbix-cli.conf` file.:

```
# zabbix-cli-init <zabbix API url>
```

The parameter `zabbix_api_url` must be defined in the configuration file. Without this parameter, `zabbix-cli` will not know where to connect. This parameter will be defined automatically if you have run the command `zabbix-cli-init`.

Remember to activate logging with `logging=ON` if you want to activate logging. The user running `zabbix-cli` must have read/write access to the log file defined with `log_file`. This parameter will be defined automatically with an OFF value if you have run the command `zabbix-cli-init`.

From version 1.6.0 we have a new zabbix-cli command that can be used to see all the active configuration files in your system and the configuration parameters that zabbix-cli is using:

```
[zabbix-cli rafael@zabbix-ID]$ show_zabbixcli_config

+-----------------------------------------------+
| Active configuration files                    |
+-----------------------------------------------+
| */usr/share/zabbix-cli/zabbix-cli.fixed.conf |
| */etc/zabbix-cli/zabbix-cli.fixed.conf       |
| */root/.zabbix-cli/zabbix-cli.conf           |
| */etc/zabbix-cli/zabbix-cli.conf             |
| */usr/share/zabbix-cli/zabbix-cli.conf       |
+-----------------------------------------------+

+-------------------------------------+----------------------------------------+
|              Configuration parameter | Value                                  |
+-------------------------------------+----------------------------------------+
|                       zabbix_api_url | https://zabbix.example.org         |
|                            system_id | zabbix-ID                              |
|                    default_hostgroup | All-hosts                              |
|               default_admin_usergroup | Zabbix-admin                           |
|        default_create_user_usergroup | All-users                              |
| default_notification_users_usergroup | All-notification-users                 |
```

```
|            default_directory_exports | /home/user/zabbix_exports             |
|                default_export_format | XML                                   |
|    include_timestamp_export_filename | ON                                    |
|                           use_colors | ON                                    |
|                 use_auth_token_file | ON                                    |
|                             logging | ON                                    |
|                           log_level | INFO                                  |
|                            log_file | /home/user/.zabbix-cli/zabbix-cli.log |
+-------------------------------------+---------------------------------------+
```

Environment Authentication

You can define the `ZABBIX_USERNAME` and `ZABBIX_PASSWORD` environment variables to pass authentication credentials to `zabbix-cli`.

For example:

```
export ZABBIX_USERNAME=zbxuser
read -srp "Zabbix Password: " ZABBIX_PASSWORD; export ZABBIX_PASSWORD;
zabbix-cli
```

NOTE: It is important to remember that this method will save the

password in clear text in a environment variable. This value will be available to other processes running in the same session.

Authentication file

You can define the file `$HOME/.zabbix-cli_auth` if you want to avoid to write your username and password everytime you use `zabbix-cli`. This can be useful if you are running `zabbix-cli` in non-interactive modus from scripts or automated jobs.

The format of this file is a line with this information:

```
USERNAME::PASSWORD
```

NOTE: The password will be saved in clear text so be carefull with the information saved here and restrict access to this file only to your user. `chmod 400 ~/.zabbix-cli_auth` will be defined by `zabbix-cli` on this file the first time it uses it.

Authentication token file

The file `$HOME/.zabbix-cli_auth_token` will be created with information about the API-auth-token from the last login if the parameter `use_auth_token_file=ON` is defined in the configuration file.

The information in this file will be used, if we can, to avoid having to write the username and password everytime you use `zabbix-cli`. This can be useful if you are running `zabbix-cli` in non-interactive modus from scripts or automated jobs.

This authentication method will work as long as the API-auth-token saved is active in Zabbix. The `Auto-logout` attribute of the user will define how long the API-auth-token will be active.

If the API-auth-token is not valid, `zabbix-cli` will delete the file `$HOME/.zabbix-cli_auth_token` and you will have to login again with a valid username and password.

Zabbix-CLI shell

The Zabbix-CLI interactive shell can be started by running the program `/usr/bin/zabbix-cli`

```
[user@host]# zabbix-cli

##############################################################
Welcome to the Zabbix command-line interface (v.1.7.0)
##############################################################
Type help or \? to list commands.

[zabbix-cli rafael@zabbix-ID]$ help

Documented commands (type help <topic>):
========================================
EOF                              shell
acknowledge_event                show_alarms
acknowledge_trigger_last_event   show_global_macros
add_host_to_hostgroup            show_history
add_user_to_usergroup            show_host
add_usergroup_permissions        show_host_inventory
clear                            show_host_usermacros
create_host                      show_hostgroup
create_host_interface            show_hostgroups
create_hostgroup                 show_hosts
create_maintenance_definition    show_items
create_notification_user         show_maintenance_definitions
create_user                      show_maintenance_periods
create_usergroup                 show_template
define_global_macro              show_templates
define_host_monitoring_status    show_trigger_events
define_host_usermacro            show_triggers
export_configuration             show_usergroup
import_configuration             show_usergroups
link_template_to_host            show_usermacro_host_list
load_balance_proxy_hosts         show_usermacro_template_list
move_proxy_hosts                 show_users
quit                             show_zabbixcli_config
remove_host                      unlink_template_from_host
remove_host_from_hostgroup       update_host_inventory
remove_maintenance_definition    update_host_proxy
remove_user                      update_usergroup_permissions
remove_user_from_usergroup

Miscellaneous help topics:
==========================
shortcuts  support

Undocumented commands:
======================
help
```

NOTE: It is possible to use Zabbix-CLI in a non-interactive modus by running `/usr/bin/zabbix-cli` with the parameter `--command <zabbix_command>` or `-C <zabbix_command>` in the OS shell. This can be used to run `zabbix-cli` commands from shell scripts or other programs .e.g.

```
[user@host]# zabbix-cli -C "show_usergroups"

+---------+--------------------------+--------------------+-------------+
```

```
| GroupID | Name                        |     GUI access     |    Status   |
+---------+-----------------------------+--------------------+-------------+
|      13 | DBA                         | System default (0) |  Enable (0) |
|       9 | Disabled                    | System default (0) | Disable (1) |
|      11 | Enabled debug mode          | System default (0) |  Enable (0) |
|       8 | Guests                      |    Disable (2)     | Disable (1) |
|      12 | No access to the frontend   |    Disable (2)     |  Enable (0) |
|      49 | testgroup                   | System default (0) |  Enable (0) |
|      15 | Test users                  | System default (0) |  Enable (0) |
|      16 | Test users intern           |    Internal (1)    |  Enable (0) |
|       7 | Zabbix administrators       |    Internal (1)    |  Enable (0) |
|      14 | Zabbix core                 | System default (0) |  Enable (0) |
+---------+-----------------------------+--------------------+-------------+
```

From version 1.5.4 it is possible to use the parameter `--file <zabbix_command_file>` or `-f <zabbix_command_file>` to define a file with multiple `zabbix-cli` commands.

Some performance improvements get activated when executing `zabbix-cli` in this way. The perfomance gain when running multiple commands via an input file can be as high as 70% when creating new hosts in Zabbix.

```
[user@host]# cat zabbix_input_file.txt

# This a comment.
# Creating hosts.

create_host test000001.example.net All-manual-hosts .+ 1
create_host test000002.example.net All-manual-hosts .+ 1
create_host test000003.example.net All-manual-hosts .+ 1

# Deleting hosts

remove_host test000001.example.net
remove_host test000002.example.net
remove_host test000003.example.net

[user@host]# zabbix-cli -f zabbix_input_file.txt

[OK] File [/home/user/zabbix_input_file.txt] exists. Bulk execution of commands defined

[Done]: Host (test000001.example.net) with ID: 14213 created
[Done]: Host (test000002.example.net) with ID: 14214 created
[Done]: Host (test000003.example.net) with ID: 14215 created
[Done]: Hosts (test000001.example.net) with IDs: 14213 removed
[Done]: Hosts (test000002.example.net) with IDs: 14214 removed
[Done]: Hosts (test000003.example.net) with IDs: 14215 removed
```

One can also use the parameters `--output csv` or `--output json` when running `zabbix-cli` in non-interactive modus to generate an output in CSV or JSON format.

```
[user@host ~]# zabbix-cli --output csv show_usergroups

"13","DBA","System default (0)","Enable (0)"
"9","Disabled","System default (0)","Disable (1)"
"11","Enabled debug mode","System default (0)","Enable (0)"
```

```
"8","Guests","Disable (2)","Disable (1)"
"12","No access to the frontend","Disable (2)","Enable (0)"
"49","testgroup","System default (0)","Enable (0)"
"15","Test users","System default (0)","Enable (0)"
"16","Test users intern","Internal (1)","Enable (0)"
"7","Zabbix administrators","Internal (1)","Enable (0)"
"14","Zabbix core","System default (0)","Enable (0)"
```

Remember that you have to use "" and escape some characters if running commands in non-interactive modus with parameters that have spaces or special characters for the shell.e.g.

```
[user@host ~]# zabbix-cli -C "show_host * \"'available':'2','maintenance_status':'1'\" "

+--------+----------------------+--------------------------+-----------------------------
| HostID | Name                 | Hostgroups               | Templates
+--------+----------------------+--------------------------+-----------------------------
|  10110 | test01.uio.no        | [8] Database servers     | [10102] Template App SSH Ser
|        |                      |                          | [10104] Template ICMP Ping
|        |                      |                          | [10001] Template OS Linux
|        |                      |                          |
|        |                      |                          |
|        |                      |                          |
|        |                      |                          |
|        |                      |                          |
|        |                      |                          |
|        |                      |                          |
|        |                      |                          |
|        |                      |                          |
+--------+----------------------+--------------------------+-----------------------------
|  10484 | test02.uio.no        | [12] Web servers         | [10094] Template App HTTP Se
|        |                      | [13] PostgreSQL servers  | [10073] Template App MySQL
|        |                      | [17] MySQL servers       | [10102] Template App SSH Ser
|        |                      | [21] ssh servers         | [10104] Template ICMP Ping
|        |                      | [5] Discovered hosts     |
|        |                      | [8] Database servers     |
+--------+----------------------+--------------------------+-----------------------------
|  10427 | test03.uio.no        | [12] Web servers         | [10094] Template App HTTP Se
|        |                      | [17] MySQL servers       | [10073] Template App MySQL
|        |                      | [21] ssh servers         | [10102] Template App SSH Ser
|        |                      | [5] Discovered hosts     | [10104] Template ICMP Ping
|        |                      | [8] Database servers     |
+--------+----------------------+--------------------------+-----------------------------
```

acknowledge_event

This command acknowledges an event

```
acknowledge_events [eventIDs]
                   [message]
```

Parameters:

- **[eventIDs]: IDs of the events to acknowledge. One can define** several values in a comma separated list.
- **[message]:** Text of the acknowledgement message.

acknowledge_trigger_last_event

This command acknowledges the last event of a trigger.

```
acknowledge_trigger_last_event [triggerIDs]
                               [message]
```

Parameters:

- **[triggerIDs]:** IDs of the triggers to acknowledge. One can define several values in a comma separated list.
- **[message]:** Text of the acknowledgement message.

add_host_to_hostgroup

This command adds one/several hosts to one/several hostgroups

```
add_host_to_hostgroup [hostnames]
                      [hostgroups]
```

Parameters:

- **[hostnames]:** Hostname or zabbix-hostID. One can define several values in a comma separated list.
- **[hostgroups]:** Hostgroup name or zabbix-hostgroupID. One can define several values in a comma separated list.

add_user_to_usergroup

This command adds one/several users to one/several usergroups

```
add_user_to_hostgroup [usernames]
                      [usergroups]
```

Parameters:

- **[usernames]:** Username or zabbix-userID. One can define several values in a comma separated list.
- **[usergroups]:** Usergroup name or zabbix-usergroupID. One can define several values in a comma separated list.

add_usergroup_permissions

This command adds a permission for an usergroup on a hostgroup.

If the usergroup already have permissions on the hostgroup, nothing will be changed.

```
define_usergroup_permissions [usergroup]
                             [hostgroups]
                             [permission code]
```

Parameters:

- **usergroup:** Usergroup that will get a permission on a hostgroup
- **hostgroups:** Hostgroup names where the permission will apply. One can define several values in a comma separated list.

- **permission:**
  - **deny**: Deny [usergroup] all access to [hostgroups]
  - **ro**: Give [usergroup] read access to [hostgroups]
  - **rw**: Give [usergroup] read and write access to [hostgroups]

clear

This command clears the screen and shows the welcome banner

```
clear
```

create_host

This command creates a host.

```
create_host [hostname|IP]
            [hostgroups]
            [proxy]
            [status]
```

Parameters:

- **[Hostname|IP]:** Hostname or IPaddress
- **[hostgroups]:** Hostgroup name or zabbix-hostgroupID to add the host to. One can define several values in a comma separated list.

  Remember that the host will get added per default to all hostgroups defined with the parameter `default_hostgroup` in the zabbix-cli configuration file.

  This command will fail if both `default_hostgroup` and [hostgroups] are empty.
- **[proxy]:** Proxy server used to monitor this host. One can use regular expressions to define a group of proxy servers from where the system will choose a random proxy.

  If this parameter is not defined, the system will assign a random proxy from the list of all available proxies.

  If the system does not have proxy servers defined, the new host will be monitor by the Zabbix-server.

  e.g. Some regular expressions that can be used:
  - *proxy-(prod|test)+d.example.org*

    e.g. proxy-prod1.example.org and proxy-test8.example.org will match this expression.
  - .+

    All proxies will match this expression.
- **[status]:** Status of the host. If this parameter is not defined, the system will use the default.
  - 0 - (default) monitored host
  - 1 - unmonitored host

All host created with this function will get assigned a default interface of type 'Agent' using the port 10050.

create_host_interface

This command creates a hostinterface

```
create_host_interface [hostname]
                      [interface connection]
                      [interface type]
                      [interface port]
                      [interface IP]
                      [interface DNS]
                      [default interface]
```

Parameters:

- **[hostname]**: Hostname
- **[interface connection]**: Type of connection. Possible values:
  - 0 - Connect using host DNS name (Default) or interface DNS if provided
  - 1 - Connect using host IP address
- **[interface type]**: Type of interface. Possible values:
  - 1 - Zabbix agent
  - 2 - SNMP (Default)
  - 3 - IPMI
  - 4 - JMX
- **[interface port]**: Interface port (Default: 161)
- **[interface IP]**: IP address if interface connection is 1
- **[interface DNS]**: DNS if interface connection is 0: (hostname by default)
- **[default interface]**: Define this interface som default. Possible values:
  - 0 - Not default interface
  - 1 - Default interface (Default)

The default value for a parameter is shown between brackets []. If the user does not define any value or a wrong value, the default value will be used. This command can be run with or without parameters. e.g.:

create_hostgroup

This command creates a hostgroup

```
create_hostgroup [group name]
```

Parameters:

- **[group name]:** Name of the hostgroup

create_maintenance_definition

This command creates a 'one time only' maintenance definition for a defined period of time. Use the zabbix dashboard for more advance definitions.

```
create_maintenance_definition [name]
                              [description]
                              [host/hostgroup]
                              [time period]
                              [maintenance type]
```

Parameters:

- **[name]**: Maintenance definition name.
- **[description]**: Maintenance definition description
- **[host/hostgroup]**: Host/s and/or hostgroup/s the that will undergo maintenance.

  One can define more than one value in a comma separated list and mix host and hostgroup values.
- **[time period]** Time period when the maintenance must come into effect.

  One can define an interval between to timestamps in ISO format or a time period in minutes, hours or days from the moment the definition is created.

  e.g. From 22:00 until 23:00 on 2016-11-21 -> '2016-11-21T22:00 to 2016-11-21T23:00'

  2 hours from the moment we create the maintenance -> '2 hours'
- **[maintenance type]** Maintenance type.

  Type values:

  - 0 - (default) With data collection
  - 1 - Without data collection

create_notification_user

This command creates a notification user. These users are used to send notifications when a zabbix event happens.

Sometimes we need to send a notification to a place not owned by any user in particular, e.g. an email list or jabber channel but Zabbix has not the possibility of defining media for a usergroup.

This is the reason we use *notification users*. They are users nobody owns, but that can be used by other users to send notifications to the media defined in the notification user profile.

All notification users will have an 'Alias' value that starts with *notification-user-*

Check the parameter **default_notification_users_usergroup** in your zabbix-cli configuration file. The usergroup defined here has to exists if you want this command to work.

```
create_notification_user [sendto]
                         [mediatype]
                         [remarks]
```

Parameters:

- **[sendto]**: E-mail address, SMS number, jabber address, ...
- **[mediatype]**: One of the media types names defined in your Zabbix installation, e.g. Email, SMS, jabber, ...
- **[remarks]**: Comments about this user. e.g. Operations email. Max lenght is 20 characters.

create_user

This command creates a user.

```
create_user [alias]
            [name]
            [surname]
            [passwd]
            [type]
            [autologin]
```

```
            [autologout]
            [groups]
```

Parameters:

- **[alias]:** User alias (account name)
- **[name]:** Name of the user
- **[surname]:** Surname of the user
- **[passwd]:** Password
- **[type]:** Type of the user. Possible values:
    - 1 - (default) Zabbix user;
    - 2 - Zabbix admin;
    - 3 - Zabbix super admin.
- **[autologin]:** Whether to enable auto-login. Possible values:
    - 0 - (default) auto-login disabled;
    - 1 - auto-login enabled.
- **[autologout]:** User session life time in seconds. If set to 0, the session will never expire. Default: 86400
- **[groups]:** User groups to add the user to.

  Remember that the user will get added per default to all usergroups defined with the parameter `default_usergroup` in the zabbix-cli configuration file.

  This command will fail if both `default_usergroup` and [groups] are empty.

create_usergroup

This command creates an usergroup

```
create_usergroup [group name]
                 [GUI access]
                 [Status]
```

Parameters:

- **[group name]:** Name of the usergroup
- **[GUI access]:** Frontend authentication method of the users in the group. Possible values:
    - 0 - (default) use the system default authentication method;
    - 1 - use internal authentication;
    - 2 - disable access to the frontend.
- **[status]:** Whether the user group is enabled or disabled. Possible values are:
    - 0 - (default) enabled;
    - 1 - disabled.

define_global_macro

This command defines a global macro

```
define_global_macro [macro name]
                    [macro value]
```

Parameters:

- **macro name:** Name of the zabbix macro. The system will format this value to use the macro format definition needed by Zabbix. e.g. site_url will be converted to ${SITE_URL}
- **macro value:** Default value of the macro

define_host_usermacro

This command defines a host usermacro.

```
defines_host_usermacro [hostname]
                       [macro name]
                       [macro value]
```

Parameters:

- **hostname:** Hostname that will get the macro locally defined.
- **macro name:** Name of the zabbix macro. The system will format this value to use the macro format definition needed by Zabbix. e.g. site_url will be converted to ${SITE_URL}
- **macro value:** Default value of the macro

define_host_monitoring_status

This command defines the monitoring status of a host. A monitor status of 'Not monitored (off)' will stop all monitoring of the host and a 'Monitored (on)' value will start monitoring.

```
defines_host_monitoring_status [hostname]
                               [on/off]
```

Parameteres:

- **hostname:** Hostname that will get the monitoring status updated.

export_configuration

This command exports the configuration of different Zabbix components to a JSON or XML file. This files can be used to import or restore these objects in a Zabbix system. Several parameters in the zabbix-cli.conf configuration file can be used to control some export options.

```
export_configuration [export_directory]
                     [object type]
                     [object name]
```

Parameters:

- **[export directory]:** Directory where the export files will be saved.
- **[object type]:** Possible values: `groups`, `hosts`, `images`, `maps`, `screens`, `templates` One can use the special value `#all#` to export all object type groups.
- **[object name]:** Object name or Zabbix-ID. One can define several values in a comma separated list.

  One can use the special value #all# to export all objects in a object type group. This parameter will be defined automatically as #all# if [object type] == #all#

import_configuration

This command imports the configuration of a Zabbix component.

We use the options `createMissing=True` and `updateExisting=True` when importing data. This means that new objects will be created if they do not exists and that existing objects will be updated if they exist.

```
import_configuration [import file]
                     [dry run]
```

Parameters:

- **[import file]:** File with the JSON or XML code to import. This command will use the file extension (.json or .xml) to find out the import format.

  This command finds all the pathnames matching a specified pattern according to the rules used by the Unix shell. Tilde expansion ~, *, ?, and character ranges expressed with [ ] will be correctly matched. For a literal match, wrap the meta-characters in brackets. For example, '[?]' matches the character '?'.

- **[dry run]:** If this parameter is used, the command will only show the files that would be imported without running the import process.

  - 0 - Dry run deactivated
  - 1 (default) - Dry run activated

link_template_to_host

This command links one/several templates to one/several hosts

```
link_template_to_host [templates]
                      [hostnames]
```

Parameters:

- **[templates]:** Template or zabbix-templateID. One can define several values in a comma separated list.
- **[hostnames]:** Hostname or zabbix-hostID. One can define several values in a comma separated list.

load_balance_proxy_hosts

This command will spread hosts evenly along a serie of proxies.

```
load_balance_proxy_hosts [proxy list]
```

Parameters:

- **proxy list:** Comma delimited list with the proxies that will share the monitoring task for a group of hosts.

  The group of hosts is obtained from the hosts assigned to the proxies in [proxy list]

e.g. If proxy-1 is monitoring 1500 hosts and proxy-2 is monitoring 500 hosts, we can run this command to redistribute the 2000 hosts between the two proxies. Every proxy will get assigned automatically ca 1000 hosts from the list of 2000 host

```
load_balance_proxy_host proxy-1,proxy-2
```

move_proxy_hosts

This command moves all hosts monitored by a proxy (src) to another proxy (dst).

```
move_proxy_hosts [proxy_src]
                 [proxy_dst]
```

Parameters:

- **proxy_src:** Source proxy server.
- **proxy_dst:** Destination proxy server.

quit

This command quits/terminates the zabbix-CLI shell.

```
quit
```

A shortcut to this command is `\q`.

remove_host

This command removes a hosts

```
remove_host  [hostname]
```

Parameters:

- **[hostname]:** Hostname or zabbix-hostID.

remove_host_from_hostgroup

This command removes one/several hosts from one/several hostgroups

```
remove_host_from_hostgroup [hostnames]
                           [hostgroups]
```

Parameters:

- **[hostnames]:** Hostname or zabbix-hostID. One can define several values in a comma separated list.
- **[hostgroups]:** Hostgroup name or zabbix-hostgroupID. One can define several values in a comma separated list.

remove_maintenance_definition

This command removes one or several maintenance definitions

```
remove_maintenance_definitions [definitionID]
```

Parameters:

- **[definitionID]**: Definition ID.

  One can define more than one value in a comma separated list.

remove_user

This command removes an user.

```
remove_user [username]
```

Parameters:

- **username:** Username to remove.

remove_user_from_usergroup

This command removes an user from one/several usergroups

```
remove_user_to_usergroup [username]
                         [usergroups]
```

Parameters:

- **username:** Username to remove
- **usergroups:** Usergroup names from where the username will be removed. One can define several values in a comma separated list.

shell

This command runs a command in the operative system.

```
shell [command]
```

Parameters:

- **[command]:** Any command that can be run in the operative system.

It exists a shortcut `[!]` for this command that can be used insteed of `shell`. This command can be run only with parameters. e.g.:

```
[pgbackman]$ ! ls -l
total 88
-rw-rw-r--. 1 vagrant vagrant   135 May 30 10:04 AUTHORS
drwxrwxr-x. 2 vagrant vagrant  4096 May 30 10:03 bin
drwxrwxr-x. 4 vagrant vagrant  4096 May 30 10:03 docs
drwxrwxr-x. 2 vagrant vagrant  4096 May 30 10:03 etc
-rw-rw-r--. 1 vagrant vagrant     0 May 30 10:04 INSTALL
-rw-rw-r--. 1 vagrant vagrant 35121 May 30 10:04 LICENSE
drwxrwxr-x. 4 vagrant vagrant  4096 May 30 10:03 vagrant
```

show_alarms

This command shows all active alarms with the last event unacknowledged.

```
show_alarms [description]
            [filters]
            [hostgroups]
            [Last event unacknowledged]
```

Parameters:

- **description:** Type of alarm description to search for. Leave this parameter empty to search for all descriptions. One can also use wildcards.
- **filters:** One can filter the result by host and priority. No wildcards can be used.

  Priority values:

  - 0 - (default) not classified;
  - 1 - information;
  - 2 - warning;
  - 3 - average;
  - 4 - high;
  - 5 - disaster.
- **hostgroups:** One can filter the result to get alarms from a particular hostgroup or group og hostgroups. One can define several values in a comma separated list.
- **Last event unacknowledged:** One can filter the result after the acknowledged value of the last event of an alarm.

  Values:

  - true - (default) Show only active alarms with last event unacknowledged.
  - false - Show all active alarms, also those with the last event acknowledged.

e.g.: Get all alarms with priority 'High' that contain the word 'disk' in the description from all hostgroups in the system and the last event unacknowledged:

```
show_alarms *disk* "'priority':'4'" * true
```

show_global_macros

This command shows all global macros

```
show_global_macros
```

show_history

Show the list of commands that have been entered during the zabbix-cli shell session.

```
show_history
```

A shortcut to this command is `\s`. One can also use the *Emacs Line-Edit Mode Command History Searching* to get previous commands containing a string. Hit `[CTRL]+[r]` in the zabbix-CLI shell followed by the search string you are trying to find in the history.

show_host

This command shows hosts information

```
show_host [HostID / Hostname]
          [Filter]
```

Parameters:

- **HostID / Hostname:** One can search by HostID or by Hostname. One can use wildcards if we search by Hostname
- **Filter:**
    - Zabbix agent: 'available': (0=Unknown, 1=Available, 2=Unavailable)
    - Maintenance: 'maintenance_status': (0:No maintenance, 1:In progress)
    - Status: 'status': (0:Monitored,1: Not monitored)

e.g.: Show all hosts with Zabbix agent: Available AND Status: Monitored:

```
show_host * "'available':'1','status':'0'"
```

show_host_inventory

This command shows hosts inventory

```
show_host_inventory [Hostname]
```

Parameters:

- **Hostname:** Hostname.

This command will return all inventory information in json format when running zabbix-cli in non-interactive modus.

If zabbix-cli is running in interactive modus, only a few attributes will be shown (hostname, vendor,chassis,gateway,contact address)

show_host_usermacros

This command shows all usermacros for a host

```
show_host_usermacros [hostname]
```

Parameters:

- **Hostname:** Hostname.

show_hostgroup

This command show hostgroups information

```
show_hostgroup [hostgroup]
```

Parameters:

- **hostgroup:** Hostgroup name. One can use wildcards.

show_hostgroups

This command shows all hostgroups defined in the system.

```
show_hostgroups
```

show_hosts

This command shows all hosts defined in the system.

```
show_hosts
```

show_items

This command shows items that belong to a template.

```
show_items [template]
```

Parameters:

- **[templates]:** Template or zabbix-templateID.

show_maintenance_definitions

This command shows maintenance definitions global information. The logical operator AND will be used if one defines more than one parameter.

```
show_maintenance_definitions [definitionID]
                             [hostgroup]
                             [host]
```

Parameters:

- **[definitionID]**: Definition ID. One can define more than one value.
- **[hostgroup]**: Hostgroup name. One can define more than one value.
- **[host]**: Hostname. One can define more than one value.

show_maintenance_periods

This command shows maintenance periods global information.

```
show_maintenance_periods [definitionID]
```

Parameters:

- **[definitionID]**: Definition ID. One can define more than one value.

show_template

This command show templates information

```
show_template [Template name]
```

Parameters:

- **Template name:** One can search by template name. We can use wildcards.

show_templates

This command shows all templates defined in the system.

```
show_templates
```

show_trigger_events

This command shows the events generated by a trigger.

```
show_trigger_events [triggerID]
                    [count]
```

- **[triggerID]:** ID of the trigger we want tho show.
- **[count]:** Number of events to show (Default: 1)

show_triggers

This command shows triggers that belong to a template.

```
show_triggers [template]
```

Parameters:

- **[templates]:** Template or zabbix-templateID.

show_usergroup

This command shows user group information.

```
show_usergroup [usergroup]
```

Parameters:

- **usergroup:** User group name. One can use wildcards.

show_usergroups

This command shows user groups information.

```
show_usergroups
```

show_usermacro_host_list

This command shows all host with a defined usermacro

```
show_usermacro_host_list [usermacro]
```

Parameters:

- **usermacro:** Name of the zabbix usermacro. The system will format this value to use the macro format definition needed by Zabbix. e.g. site_url will be converted to ${SITE_URL}

show_usermacro_template_list

This command shows all templates with a defined macro

```
show_usermacro_template_list [macro name]
```

Parameters:

- **usermacro:** Name of the zabbix usermacro. The system will format this value to use the macro format definition needed by Zabbix. e.g. site_url will be converted to ${SITE_URL}

show_users

This command shows users information.

```
show_users
```

show_zabbixcli_config

This command shows information about the configuration used by this zabbix-cli instance.

```
show_zabbixcli_config
```

ulink_template_from_host

This command unlinks and clear one/several templates from one/several hosts

```
unlink_template_from_host [templates]
                          [hostnames]
```

Parameters:

- **[templates]:** Template or zabbix-templateID. One can define several values in a comma separated list.
- **[hostnames]:** Hostname or zabbix-hostID. One can define several values in a comma separated list.

update_host_inventory

This command updates one hosts' inventory

```
update_host_inventory [hostname]
                      [inventory_key]
                      [inventory value]
```

Inventory key is not the same as seen in web-gui. To look at possible keys and their current values, use "zabbix-cli --use-json-format show_host_inventory <hostname>"

update_host_proxy

This command defines the proxy used to monitor a host

```
update_host_proxy [hostname]
                  [proxy]
```

Parameters:

- **hostname:** Hostname to update
- **proxy:** Zabbix proxy that will monitor [hostname]

update_usergroup_permissions

This command updates the permissions for an usergroup on a hostgroup.

```
define_usergroup_permissions [usergroup]
                             [hostgroups]
                             [permission code]
```

Parameters:

- **[usergroup]**: Usergroup that will get a permission on a hostgroup
- **[hostgroups]**: Hostgroup names where the permission will apply.

  One can define several values in a comma separated list.
- **[permission]**:
  - deny: Deny [usergroup] all access to [hostgroups]
  - ro: Give [usergroup] read access to [hostgroups]
  - rw: Give [usergroup] read and write access to [hostgroups]

Authors

In alphabetical order:

Rafael Martinez Guerrero
E-mail: rafael@postgresql.org.es / rafael@usit.uio.no
PostgreSQL-es / University Center for Information Technology (USIT), University of Oslo, Norway

License and Contributions