ZABBIX

ZQL – COMMAND LINE TOOL

Petite et dabitur vobis quaerite et invenietis ...

Vladimir Ulogov

Preface

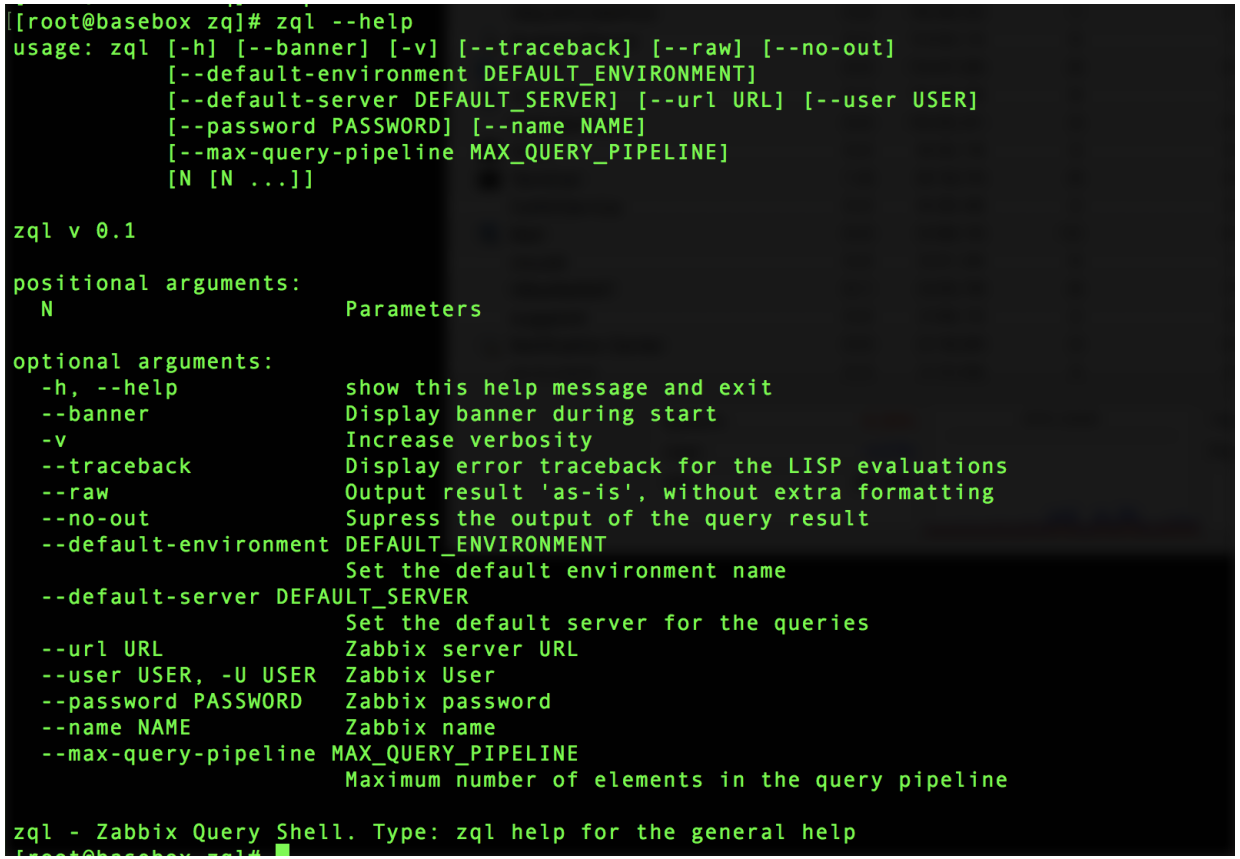
After my previous presentation, which covers the Zabbix Query language itself, you probably wondering: How I can send the queries to the Zabbix using ZQL ?

There are Python module and a Python-based API. I will cover Python API in a separate presentation

Also, you are havinbg an access to a command-line tool called “zql”, which ZQ installation process shall place in */usr/local/bin*.

ZQL. Basics

If */usr/local/bin* is your $PATH, you can call zql from your UNIX(*) command line.

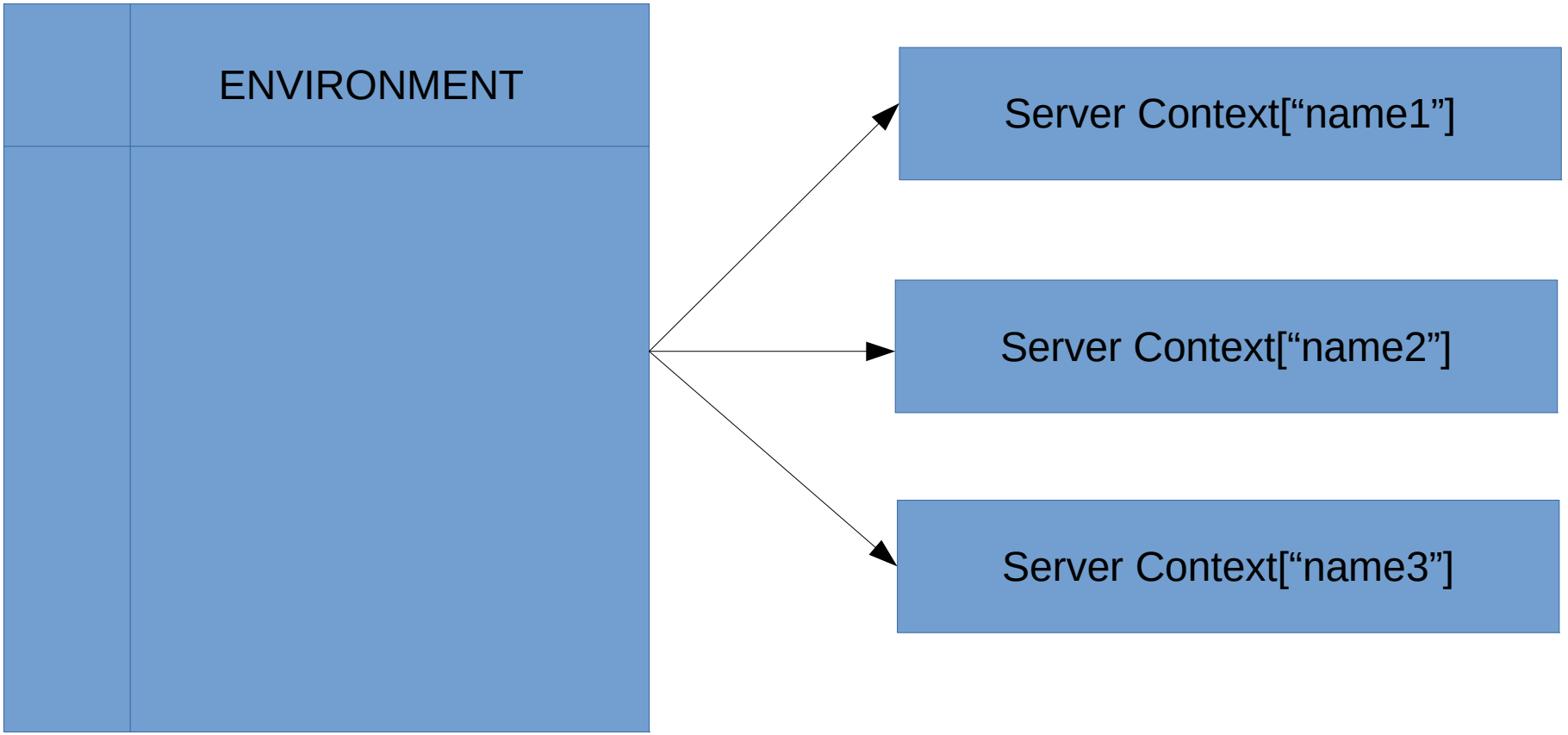


```
[root@basebox zq]# zql --help
usage: zql [-h] [--banner] [-v] [--traceback] [--raw] [--no-out]
           [--default-environment DEFAULT_ENVIRONMENT]
           [--default-server DEFAULT_SERVER] [--url URL] [--user USER]
           [--password PASSWORD] [--name NAME]
           [--max-query-pipeline MAX_QUERY_PIPELINE]
           [N [N ...]]

zql v 0.1

positional arguments:
  N                     Parameters

optional arguments:
  -h, --help            show this help message and exit
  --banner              Display banner during start
  -v                    Increase verbosity
  --traceback           Display error traceback for the LISP evaluations
  --raw                 Output result 'as-is', without extra formatting
  --no-out              Supress the output of the query result
  --default-environment DEFAULT_ENVIRONMENT
                        Set the default environment name
  --default-server DEFAULT_SERVER
                        Set the default server for the queries
  --url URL             Zabbix server URL
  --user USER, -U USER  Zabbix User
  --password PASSWORD   Zabbix password
  --name NAME           Zabbix name
  --max-query-pipeline MAX_QUERY_PIPELINE
                        Maximum number of elements in the query pipeline

zql - Zabbix Query Shell. Type: zql help for the general help
```

(*) Currently, only Unix-type operating systems are supported.

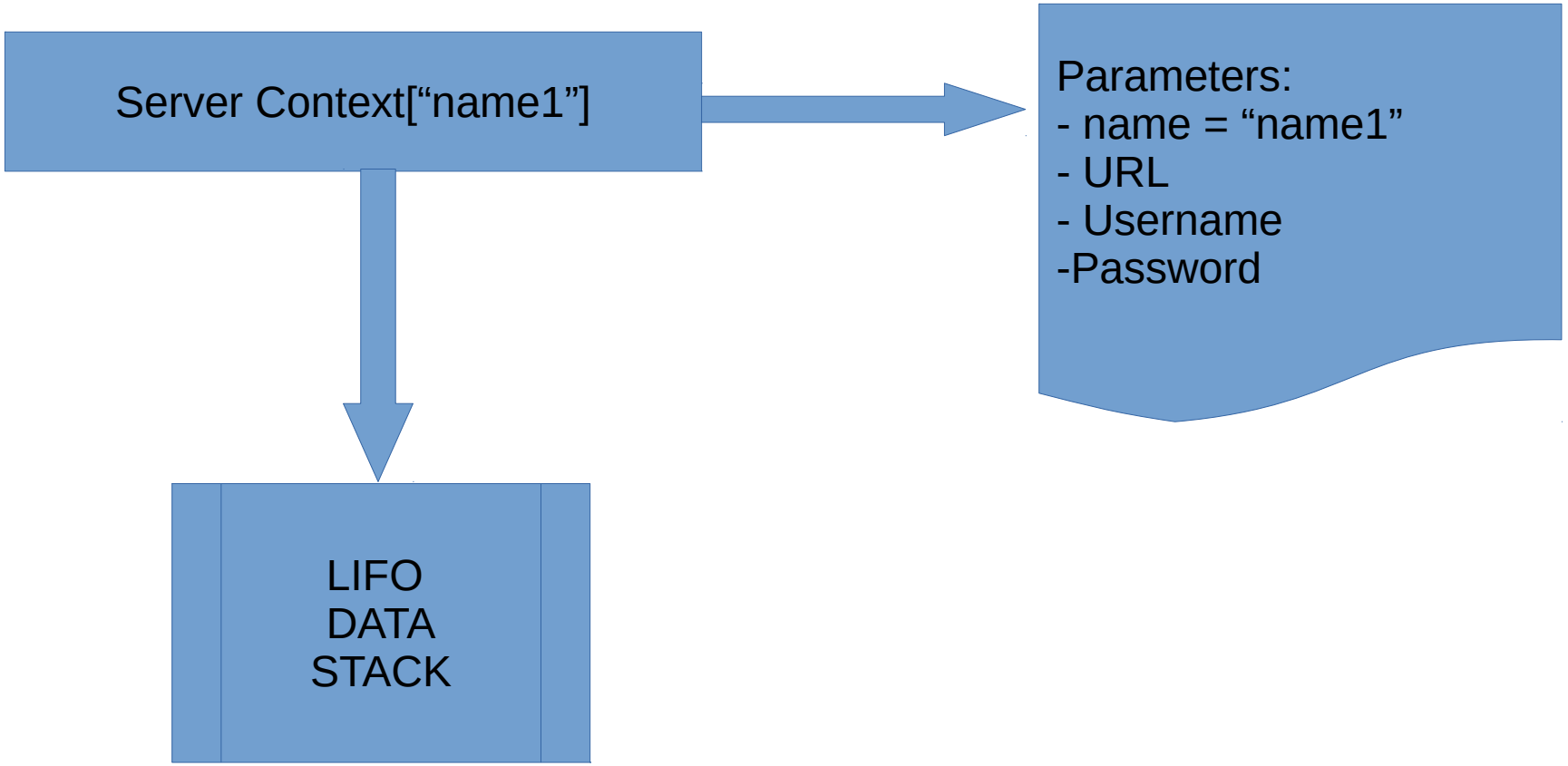
Servers and Context

Before we will explore command line options, let me remind you the concepts of the ZQ environments and a Server Contexts



Servers and Context

There is only one environment in CLI tool. Although, the API supports multiple Environments. At some point this functionality will be available for the CLI.
There is only one Server Context in CLI ver 0.1



Servers and Context

When you are working with ZQL, you have to define the Server context.

Long option	Short option	Description
--url		URL of the Zabbix server. Default is *http://127.0.0.1/zabbix*
--user	-U	Zabbix username. Default is *“Admin”*
--password		Zabbix password. Default is “zabbix”
--default-environment		Set the name of the Zabbix default environment. Default is: *“default”*
--name		Set the name of the Zabbix server. Default is *“zabbix”*

Servers and Context

When you are creating execution context, you have two options:

- Specify exact context. In this case, first “word” in your ZQL query must be a (ZBX …). Otherwise, query processor will not know which server you shall work with.
- Specify a name of the server context, using command-line parameter –defaul-server=<server name>. If provided and valid server name is found, the server context of that server will be passed to your query. Plainly speaking, zql tool will add proper (ZBX …) “word” to your query authomatically.

In the version 0.1, we have only one server context, so, --default-server doesn’t do much, except saving you some typing, but in the future, I am planning to support multiple server contexts.

Misc. Options

Misc. output formatting options.

--raw By default, ZQL will try to “pretty print” the output value. If this option is specified, the query result will be sent to the output without extra formatting.

--no-out If passed, ZQL cli tool will not send any query result to the stdout.

Misc. Options

Here is debug messages and traceback control options.

- --traceback By default, if the query is throwing an error Python traceback is suppressed. All you'll see is a message saying that the query having an error. If you pass –traceback option to ZQL cli tool, partial traceback will be displayed. Very useful for the debugging.
- -v/-vv By default, if you do not pass -v, ZQL will show you only an error messages. If you pass -v, ZQL will display errors and warnings, If you pass -vv, you will see everything, errors, warnings and "OK" messages.
- --banner Will display yu a nice banner along with configurations and its' values.

Runtime Options

And before we shall discuss the queries and evaluations, let's introduce some of the runtime options:

- --max-query-pipeline=<number> When we've talked about ZQL language, I've mentioned, that the ZQL is a stack-based language, where the data is passed between "context words" through the stack. This parameter is defined the maximum size of that stack. Default is 100

Run-time basics

So, we've covered the optional parameters for the ZQL command-line tool. Now, let see what is the ZQL commands and how you can use this tool.

The format of the command line is consisting of optional parameters, command and parameters for the command.

```
$ zql <optional parameters> <command> <command parameters>
```

Example:

```
$ zql -–banner -v eval "(print \"Hello world\")"
```

How to get some help ?

The furst command, that we will discuss, that is “help”. As the name suggests, it will display you some useful information about other commands and there options.

Example:

zql help
-or-
zql help help

As you see, the command “help” without any extra parameters will display you a brief information about other commands and what they are doing. In order to get some assistance about specific command, you can call zql as “*zql <command name> help*”

S-Expressions .... and more s-expresssions ...

ZQL is build around embedded dialect of the LISP language, called **Hy**(*).

“Hy (alternately, Hylang) is a dialect of the Lisp programming language designed to interact with Python by translating expressions into Python's abstract syntax tree (AST). Hy was introduced at PyCon 2013 by Paul Tagliamonte. Similar to Clojure's mapping of s-expressions onto the JVM, Hy is meant to operate as a transparent Lisp front end to Python's abstract syntax. Because Lisp allows for operating on code as data, Hy can be used to write domain-specific languages.”

https://en.wikipedia.org/wiki/Hy

(*) Get the source or read the documentation about Hy *http://docs.hylang.org/en/latest/*

S-Expressions …. and more s-expresssions …

So, the first “real” command that we will review, will be “eval”. This command will take an arguments as strings containing s-expression, evaluate them and send the result to the stdout.

Example:

```
zql --url="http://192.168.101.23/zabbix" -v eval "(do (print \"Answer is\") 42)"

– will send to the stdout --

Answer is
(do (print "Answer is") 42) = 42L
```

In this example, we are evaluating the s-expression (do ..). The first element of (do ..) expression will print on the stdout the message: “Answer is”, and the second element of the (do …) expression, “42” will be a return value of the expression.

S-Expressions …. and more s-expresssions …

You can specify multiple s-expressions, they will be eval;uated independently and sequentially

Example:

```
zql --url="http://192.168.101.23/zabbix" eval "(do \"The answer is \")" "(setv answer 42)"

– will send to the stdout --

(do "The answer is ") = u'The answer is '
(setv answer 42) = 42L
```

In this example, we are evaluating two expressions. The first (do …) expression returns the string “The answer is” and the second one, assign value 42 to variable “answer” and return this value

Queries as a pipeline

And now, let’s look on how you can send a ZQL query to the Zabbix instance:

Example:

```
zql --url="http://192.168.101.23/zabbix" query "(ZBX) (Hosts) (Filter TRUE [hostid Eq 10084]) (Out)"
```

– will send to the stdout --

(ZBX) (Hosts) (Filter TRUE [hostid Eq 10084]) (Out) = {'HOST': [{u'available': u'0', u'tls_connect': u'1', u'maintenance_type': u'0', u'ipmi_errors_from': u'0', u'ipmi_username': u'', u'snmp_disable_until': u'0', u'ipmi_authtype': u'-1', u'ipmi_disable_until': u'0', u'lastaccess': u'0', u'snmp_error': u'', u'tls_psk': u'', u'ipmi_privilege': u'2', u'jmx_error': u'', u'jmx_available': u'0', u'maintenanceid': u'0', u'snmp_available': u'0', u'tls_psk_identity': u'', u'status': u'1', u'description': u'', u'tls_accept': u'1', u'host': u'Zabbix server', u'disable_until': u'0', u'ipmi_password': u'', u'templateid': u'0', u'tls_issuer': u'', u'ipmi_available': u'0', u'maintenance_status': u'0', u'snmp_errors_from': u'0', u'ipmi_error': u'', u'proxy_hostid': u'0', u'hostid': u'10084', u'name': u'Zabbix server', u'jmx_errors_from': u'0', u'jmx_disable_until': u'0', u'flags': u'0', u'error': u'', u'maintenance_from': u'0', u'tls_subject': u'', u'errors_from': u'0'}]}

If you remember your material from the previous talk: “ZQL – The Query language”, this query connects to the default context, requests the list of the hosts, filtering-out host with hostid equal to 10084 and take the result to the stdout.

Queries as a pipeline

Command “query” can accept the list of the queries, just like “eval” accepts the list of the s-expressions and process them sequentially and send the result to the stdout.

Queries as a pipeline

If you remember, I mentioned that there is a direct relation between pipeline-based queries and s-expressions. So, let us review, how the ZQL pipeline could be transformed into a s-expression and then evaluated

I will take a query from the previous example and will translate it into an s-expression. If you remember, the pipeline passes the output from the previous function as indirect parameter of the next one.

(ZBX) (Hosts) (Filter TRUE [hostid Eq 10084]) (Out)

This query will be an equivalent to the following s-expression

(Out (Filter (Hosts (ZBX)) TRUE [hostid Eq 10084]))

Please, do take a note of how the parameters are passed from one element of the expression to another. Please feel free to evaluate this s-expression using “eval” command.

And now, it is a time for Questions.