ZABBIX

# Building your first Python module for ZLM-Cython

Using zlm-cython with ease...

# AGENDA

- What you shall know before you begin.
- Locations
- Building a module
- Testing a module
- Using a module
- What's next ?

# What you shall know before you begin.

- Apparently, you shall know the Python. You shall know it well enough, to be able to write a short and fast snippets of the code. You shall be proficient with all libraries tools and protocols involved in your project. For example: if you are monitoring *REDIS*(1) storage, you shall be familiar not only with Python itself, but with Python module which you will use to access *REDIS*, and with *REDIS* itself.

- While you beginning to developing you module, you shall be also familiar with all “common-sense” Python performance programming techniques. I am recommending to look at the book “High Performance Python” by Ian Ozsvald.

- You shall be also proficient in the area of System Administration and Systems Management well enough, so you can understand the impact of the execution of your code to the target hosts and applications. You must be comfortable when dealing with such hosts, operating systems and all applications involved.

(1) *REDIS* – high-performance in-memory data structure store, used as database, cache and message broker. http://redis.io

# What you shall know before you begin.

- And of course, you must have “hands-on” experience on how to use OS shell and OS commands, how to edit text files, how to use source-code version control tools used by developers in your organization, how your code will be integrating with configuration management, build and deployment tools.

- And this is not the end. You shall be able to create “secure” code. This means, that you code must pass check, performed by your security team. You must be familiar with all local guidelines and requirements related to application security.

- Systems and Application Architect shall know about your software development and deployment activities.

- You must know how to document your code. And actually do that Documentation

- And the last, but not least. You shall know how to test your code for a bugs and a performance issues.

# What you shall know before you begin.

Well … You are not just in IT-monitoring anymore. You are becoming Software Developer and must think and act like one.

# Locations

Open your Zabbix configuration file (1) and locate variable *LoadModulePath*

Your module files and all other directory and files will be relative to the location, defined by this variable.

Verify, that you are loading *zlm_python.so* module is loaded. Look for variable *LoadModule=zlm_python.so*

Please locate directory *pymodules*, which shall be subdirectory of the directory defined in *LoadModulePath* .

If this directory do not exists, please create it and assign ownersyhip to user cefined in variable *User*. It is probably a good idea to set user sticky bit for this directory.

Verify location of your log files as defined by *LogFile*.

(1) zabbix_server.conf or zabbix_agentd.conf.

ZABBIX

# Building a module

```
def this_is_int(ctx, *args):
    return int(42)

def this_is_float(ctx, *args):
    return float(42.0)

def this_is_string(ctx, *args):
    return "This is number 42"

main = this_is_string
```

With text editor of your choice, please create file called ZBX_test.py and define three simple functions.

This function return Integer

This function return Float

Please note:
Default function of the module
Is main()

This function return String

# Building a module

- We support only those three data types: *Integer*, *String* and a *Float*. ZLM-Python will automatically convert Python value to a Zabbix value.
- What you return from the function, will be passed directly to Zabbix
- All exceptions will be properly handled. We will talk about exceptions in a little while.
- You can refer individual functions inside the module from the Zabbix as:

  **Module name.***Function name*. Default function name is *"main"*
- The first parameter, passed to a function by ZLM-Python is *"context"*, usually referred by variable name *"ctx"*. This object is used to pass data between different process and will be explained in more details later on.

ZABBIX

# Testing a module

Before you will try to query the metric through Zabbix Agent or Zabbix Server, you shall Test a metric through *zabbix_agentd -t <metric name>*

Testing this metric

ZLM-Python initializing

ZLM-Python loading configuration

ZLM-Python pre-load our test module

```
[root@zabbix-251 pymodules]# /usr/local/sbin/zabbix_agentd -t py[ZBX_test.this_is_int]
zabbix_agentd [30035]: Warning: Executing ZBX_startup
zabbix_agentd [30035]: Warning: ZLM-python(Startup) Initializning
zabbix_agentd [30036]: Warning: ZLM-python(CM): Context Manager is entering the loop.
zabbix_agentd [30035]: Warning: ZLM-python(Config): Configuration file /usr/local/etc/zlm_python.ini found.
zabbix_agentd [30035]: Warning: ZLM-python(Config): rrd[maxsize]=10
zabbix_agentd [30035]: Warning: ZLM-python(Config): clock_collector[wait]=5.0
zabbix_agentd [30035]: Warning: ZLM-python(Startup) PYTHONPATH=/usr/local/etc/pymodules:/
usr/local/etc/pydaemons::/usr/lib64/python27.zip:/usr/lib64/python2.7:/usr/lib64/python2.7/plat-linux2:/usr/lib64/pyth
on2.7/lib-tk:/usr/lib64/python2.7/lib-old:/usr/lib64/python2.7/lib-dynload:/usr/lib64/python2.7/site-packages:/usr/lib
/python2.7/site-packages
zabbix_agentd [30035]: Warning: ZLM-python(Module): Module ZBX_time has been loaded from /usr/local/etc/pymodules/ZBX_
time.py
zabbix_agentd [30035]: Warning: ZLM_time(startup) Executing
zabbix_agentd [30035]: Warning: ZLM-python(Module): Module ZF has been loaded from /usr/local/etc/pymodules/ZF.py
zabbix_agentd [30035]: Warning: ZLM-python(Module): Module ZBX_test has been loaded from /usr/local/etc/pymodules/ZBX_
test.py
zabbix_agentd [30035]: Warning: ZLM-python(Module): picking up module ZBX_test from the cache
py[ZBX_test.this_is_int]                        [u|42]
zabbix_agentd [30035]: Warning: ZLM-python(Shutdown): Doing so.
zabbix_agentd [30035]: Warning: ZLM-python(Shutdown): CM manager is down.
```

ZBX_test.this_is_int() returns 42.
It is “unsigned Integer”. SUCCESS !

# Testing a module

What if something goes wrong ? What yo will see from *zabbix_agentd -t <metric name>* ?

Let's create a metric collection function, which will do nothing, but throw a Python exception.

```
def raise_exception(ctx, *args):
    raise AttributeError, "You are called a function, which raises an exception"
```

Raise an Exception

AttributeError

… and here is parameter

When called, this function will throw an AttributeError exception and pass the parameter string.

ZABBIX

# Testing a module

What if something goes wrong ? What yo will see from *zabbix_agentd -t <metric name>* ?

```
[root@zabbix-251 pymodules]# /usr/local/sbin/zabbix_agentd -t py[ZBX_test.raise_exception]
zabbix_agentd [30018]: Warning: Executing ZBX_startup
zabbix_agentd [30018]: Warning: ZLM-python(Startup) Initializning
zabbix_agentd [30019]: Warning: ZLM-python(CM): Context Manager is entering the loop.
zabbix_agentd [30018]: Warning: ZLM-python(Config): Configuration file /usr/local/etc/zl
zabbix_agentd [30018]: Warning: ZLM-python(Config): rrd[maxsize]=10
zabbix_agentd [30018]: Warning: ZLM-python(Config): clock_collector[wait]=5.0
zabbix_agentd [30018]: Warning: ZLM-python(Startup) PYTHONPATH=/usr/local/etc/pymodules:/usr/local/etc/pymodules/lib:/
usr/local/etc/pydaemons::/usr/lib64/python27.zip:/usr/lib64/python2.7:/usr/lib64/python2.7/plat-linux2:/usr/lib64/pyth
on2.7/lib-tk:/usr/lib64/python2.7/lib-old:/usr/lib64/python2.7/lib-dynload:/usr/lib64/python2.7/site-packages:/usr/lib
/python2.7/site-packages
zabbix_agentd [30018]: Warning: ZLM-python(Module): Module ZBX_time has been loaded from                    X_
time.py
zabbix_agentd [30018]: Warning: ZLM_time(startup) Executing
zabbix_agentd [30018]: Warning: ZLM-python(Module): Module ZF has been loaded from /usr/local/etc/pymodules/ZF.py
zabbix_agentd [30018]: Warning: ZLM-python(Module): Module ZBX_test has been loaded from /usr/local/etc/pymodules/ZBX_
test.py
zabbix_agentd [30018]: Warning: ZLM-python(Module): picking up module ZBX_test from the cache
zabbix_agentd [30018]: Warning: ZLM-python(Call): Module ZBX_test->raise_exception threw traceback
py[ZBX_test.raise_exception]                    [m|ZBX_NOTSUPPORTED] [ZLM-python: Module ZBX_test->raise_exception threw
 traceback: Traceback (most recent call last):
  File "zlm_python.pyx", line 428, in zlm_python.ZBX_call (zlm_python_pyx.c:9933)
  File "/usr/local/etc/pymodules/ZBX_test.py", line 11, in raise_exception
    raise AttributeError, "You are called a function, which raises an exception"
AttributeError: You are called a function, which raises an exception
]
zabbix_agentd [30018]: Warning: ZLM-python(Shutdown): Doing so.
zabbix_agentd [30018]: Warning: ZLM-python(Shutdown): CM manager is down.
```

Call that metric

Looks okay, so far

Oy ! Our metric threw a traceback and become NOTSUPPORTED

And here goes Python traceback, to help us to troubleshoot the problem.

# Testing a module

When you metric functions exit successfully and generate proper output, you can move to the next test: collecting data through *zabbix_agentd* (if your metric collector designed to run from *zabbix_agentd*)

If your *zabbix_agentd -t <metricname>* is core-dumped, CONGRATULATIONS ! You've found a bug in ZLM-python. Yes, there are bugs and I will try to fix them as soon as I can. Please report this bug to

https://github.com/vulogov/zlm-cython/issues

Please include as much information as possible.

# Testing a module

Try to query your new metric keys (if you are loading the module inside Zabbix Agent)

```
[root@zabbix-251 pymodules]# zabbix_get -s localhost -k py[ZBX_test]
This is number 42
[root@zabbix-251 pymodules]# zabbix_get -s localhost -k py[ZBX_test.this_is_float]
42.000000
[root@zabbix-251 pymodules]# zabbix_get -s localhost -k py[ZBX_test.this_is_int]
42
```

Here, we are calling default function

Calling function which returns String, getting String

Calling function which returns Float, getting Float

Calling function which returns Integer, getting Integer

# Testing a module

The previous attempt was successful, but what if we get an error ?

Collect this metric

```
[root@zabbix-251 pymodules]# zabbix_get -s localhost -k py[ZBX_test.raise_exception]
ZBX_NOTSUPPORTED: ZLM-python: Module ZBX_test->raise_exception threw traceback: Traceback (most recent call last):
  File "zlm_python.pyx", line 428, in zlm_python.ZBX_call (zlm_python_pyx.c:9933)
  File "/usr/local/etc/pymodules/ZBX_test.py", line 11, in raise_exception
    raise AttributeError, "You are called a function, which raises an exception"
AttributeError: You are called a function, which raises an exception
```

And here goes Python traceback, to help us to troubleshoot the problem.

Item will become a NOTSUPPORTED

# Using a module

Now, let's create a Zabbix Item, for the function that returns Integer

```
def this_is_int(ctx, *args):
    return int(42)
```

Name for the metric

Calling ZLM-python

Collecting from zabbix_agend

Module: ZBX_test
Function: this_is_int()

| | |
|---|---|
| Name | Int Item |
| Type | Zabbix agent |
| Key | py[ZBX_test.this_is_int] Select |
| Type of information | Numeric (unsigned) |
| Data type | Decimal |

Type of the Zabbix Item shall match to the type of the data returned from metric function

# Using a module

Now, let's create a Zabbix Item, for the function that returns float

```
def this_is_float(ctx, *args):
    return float(42.0)
```

Name for the metric

Calling ZLM-python

Collecting from zabbix_agend

Module: ZBX_test
Function: this_is_float()

| | |
|---|---|
| Name | Float Item |
| Type | Zabbix agent |
| Key | py[ZBX_test.this_is_float] Select |
| Type of information | Numeric (float) |

Type of the Zabbix Item shall match to the type of the data returned from metric function

# Using a module

Now, let's create a Zabbix Item, for the function that returns string

```
def this_is_string(ctx, *args):
    return "This is number 42"
```

Name for the metric

Calling ZLM-python

Collecting from zabbix_agend

Module: ZBX_test
Function: this_is_float()

| | |
|---|---|
| Name | String Item |
| Type | Zabbix agent |
| Key | py[ZBX_test.this_is_string] Select |
| Type of information | Text |
| Update interval (in sec) | 30 |

Type of the Zabbix Item shall match to the type of the data returned from metric function

ZABBIX

# Using a module

Now, let's create a Zabbix Item, for the function that returns float and ZLM-Python is executed on Zabbix Server

```
def this_is_float(ctx, *args):
    return float(42.0)
```

Name for the metric

Parent items ZLM_Template

Name Float Item. ZLM-python on zabbix_server

Calling ZLM-python

Type Simple check

Collecting on zabbix_server

Key py[ZBX_test.this_is_float,"Ignore this parameter"]

Module: ZBX_test
Function: this_is_float()

ost interface 127.0.0.1 : 10050

User name

Password

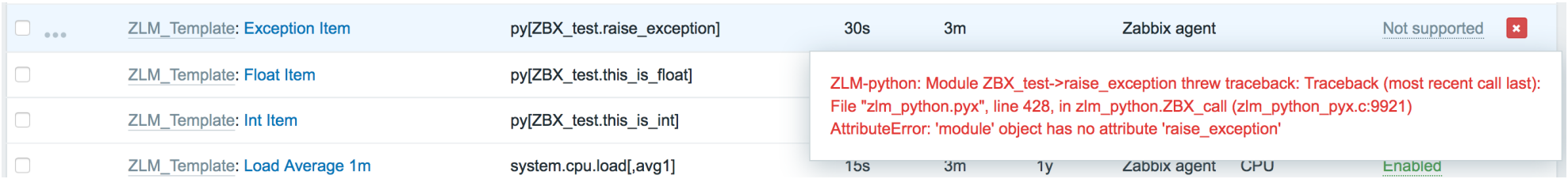
Type of information Numeric (float)

Type of the Zabbix Item shall match to the type of the data returned from metric function

# Using a module

ZLM-python will help you to troubleshoot you metric collectors by properly passing information about exceptions back to Zabbix. If your metric throw an exception, the Item will become “Not supported” and you can get the information about traceback right from the frontend by clicking on the red icon.

Display the information about an exception



# What's next ?

Passing parameters from Zabbix to Python functions

```
def this_is_string(ctx, *args):
    return "This is number 42"
```

Non-positional parameters

You can use ether positional or non-positional parameters. Please note, all parameters are passed as strings. You shall do the proper type sanitation and conversion inside your module.

# What's next ?

Using the ZLM-python Context object

```
def this_is_string(ctx, *args):
    return "This is number 42"
```

Proxy for the context object of the ZLM-Cython core

ZLM-python context object allow you to save some data to the global dictionary and pass it between Zabbix threads. If you assign some variable to the context in one thread, like *ctx.name* = “Value”, you can refer to it later on using this notation: *ctx.name* . Context is global and we do support all Python picklable data types.

質問

Q/A ?

Jautāt ?

Interroger ?

FRAGEN ?

ВОПРОСЫ ?