Insight in Mechanics of Zabbix Modules

Gleb Ivanovsky

ZABBIX

September 9, 2016

Outline

1. Extend Zabbix
   - Why?
   - How?
2. Write modules
   - Prerequisites
   - Code example
   - Compilation
3. Use modules
4. Share modules

Reasons to extend Zabbix

Reasons to extend Zabbix

- Zabbix can't support *everything* out of the box

Reasons to extend Zabbix

- Zabbix can't support *everything* out of the box
- Zabbix has certain restrictions

Ways of extending Zabbix

Scripts

- `script[...]`
- `system.run[...]`
- `UserParameter=...`

Modules

- server
- proxies
- UNIX agents

Ways of extending Zabbix

Scripts

- `script[...]`
- `system.run[...]`
- `UserParameter=...`

Pros:

- easy to write
- easy to modify

Cons:

- slow
- resource-hungry

Modules

- server
- proxies
- UNIX agents

Pros:

- fast
- lightweight

Cons:

- hard to write
- *modify* means *recompile*

What you will need

Building inside Zabbix source tree

1. get Zabbix code
2. run `./bootstrap` (if Zabbix code is from SVN repository)
3. run `./configure ...`

What you will need

Building inside Zabbix source tree

1. get Zabbix code
2. run `./bootstrap` (if Zabbix code is from SVN repository)
3. run `./configure ...`

Building outside of Zabbix source tree

1. get `include/module.h`
2. replace `#include "zbxtypes.h"` with

```
#include <inttypes.h>
#define zbx_uint64_t     uint64_t;
```

3. replace `#include "sysinc.h"` with necessary standard headers (like `#include <stdlib.h>`, etc.) in module source

Minimalistic module API

src/modules/my_module/my_module.c:

```
#include "sysinc.h"     /* or standard headers here */
#include "module.h"

int     zbx_module_api_version(void)
{
        return ZBX_MODULE_API_VERSION;
}
```

include/module.h:

```
#define ZBX_MODULE_API_VERSION  2
```

Implementing custom checks

include/module.h:

```
#define get_rkey(request)                       ...
#define get_rparams_num(request)                ...
#define get_rparam(request, num)                ...

typedef struct {...}    ZBX_METRIC;
typedef struct {...}    AGENT_REQUEST;
typedef struct {...}    AGENT_RESULT;

#define SET_UI64_RESULT(res, val)               ...
#define SET_DBL_RESULT(res, val)                ...
#define SET_STR_RESULT(res, val)                ...
#define SET_TEXT_RESULT(res, val)               ...
#define SET_LOG_RESULT(res, val)                ...
#define SET_MSG_RESULT(res, val)                ...

#define SYSINFO_RET_OK          0
#define SYSINFO_RET_FAIL        1
```

Implementing custom checks

src/modules/my_module/my_module.c:

```
static int      my_hello(AGENT_REQUEST *request, AGENT_RESULT *result);

static ZBX_METRIC keys[] =
{
        {"my.hello", CF_HAVEPARAMS, my_hello, "World"},
        {NULL}
};

static int      my_hello(AGENT_REQUEST *request, AGENT_RESULT *result)
{
        char    *name;

        if (NULL == (name = get_rparam(request, 0)))
        {
                SET_MSG_RESULT(result, strdup("There is noone to greet..."));
                return SYSINFO_RET_FAIL;
        }

        SET_STR_RESULT(result, asprintf("Hello, %s!", name));

        return SYSINFO_RET_OK;
}
```

Implementing history export

include/module.h:

```
typedef struct
{
        zbx_uint64_t    itemid;
        int             clock;
        int             ns;
        const char      *value;
        const char      *source;
        int             timestamp;
        int             logeventid;
        int             severity;
}
ZBX_HISTORY_LOG;

typedef struct
{
        void    (*history_float_cb)(const ZBX_HISTORY_FLOAT *history, int history_num);
        void    (*history_integer_cb)(const ZBX_HISTORY_INTEGER *history, int history_num);
        void    (*history_string_cb)(const ZBX_HISTORY_STRING *history, int history_num);
        void    (*history_text_cb)(const ZBX_HISTORY_TEXT *history, int history_num);
        void    (*history_log_cb)(const ZBX_HISTORY_LOG *history, int history_num);
}
ZBX_HISTORY_WRITE_CBS;
```

Supported since Zabbix 3.2

Implementing history export

src/modules/my_module/my_module.c:

```
static void     my_history_log_cb(const ZBX_HISTORY_LOG *history, int history_num);

ZBX_HISTORY_WRITE_CBS   zbx_module_history_write_cbs(void)
{
        static ZBX_HISTORY_WRITE_CBS    my_callbacks =
        {
                NULL,   /* my_history_float_cb */
                NULL,   /* my_history_integer_cb */
                NULL,   /* my_history_string_cb */
                NULL,   /* my_history_text_cb */
                my_history_log_cb
        };

        return my_callbacks;
}
```

Supported since Zabbix 3.2

Implementing history export

src/modules/my_module/my_module.c:

```
static void     my_history_log_cb(const ZBX_HISTORY_LOG *history, int history_num)
{
        int     i;

        for (i = 0; i < history_num; i++)
        {
                /* do something with: */
                /* history[i].itemid, */
                /* history[i].clock, */
                /* history[i].ns, */
                /* history[i].value, */
                /* ... */
        }
}
```

Supported since Zabbix 3.2

Implementing anything you wish

All *standard* toys are at your disposal:

Implementing anything you wish

All *standard* toys are at your disposal:

- shared memory

Implementing anything you wish

All *standard* toys are at your disposal:

- shared memory
- semaphores

Implementing anything you wish

All *standard* toys are at your disposal:

- shared memory
- semaphores
- sockets

Implementing anything you wish

All *standard* toys are at your disposal:

- shared memory
- semaphores
- sockets
- pipes

Implementing anything you wish

All *standard* toys are at your disposal:

- shared memory
- semaphores
- sockets
- pipes
- forks

Implementing anything you wish

All *standard* toys are at your disposal:

- shared memory
- semaphores
- sockets
- pipes
- forks
- files

Implementing anything you wish

All *standard* toys are at your disposal:

- shared memory
- semaphores
- sockets
- pipes
- forks
- files
- ...

Compile module

Building inside Zabbix source tree

- copy module sources to `src/modules/`
- `cd src/modules/my_module`
- run `make` or `gcc -fPIC -shared -o my_module.so my_module.c -I../../../include`

Compile module

Building inside Zabbix source tree

- copy module sources to `src/modules/`
- `cd src/modules/my_module`
- run `make` or `gcc -fPIC -shared -o my_module.so my_module.c -I../../../include`

Building outside of Zabbix source tree

- put `module.h` and `my_module.c` into one directory
- run `gcc -fPIC -shared -o my_module.so my_module.c`

Start using module

Backend

1. edit configuration file:
   - `LoadModulePath=...`
   - `LoadModule=...`
   - `LoadModule=...`
   - `...`
2. restart Zabbix daemon

Start using module

Backend

1. edit configuration file:
   - `LoadModulePath=...`
   - `LoadModule=...`
   - `LoadModule=...`
   - `...`
2. restart Zabbix daemon

Frontend

1. create item
2. choose **Type**
   - for server and proxy modules: **Simple check**
   - for agent modules: **Zabbix agent** or **Zabbix agent (active)**

Start using module

Backend

1. edit configuration file:
   - `LoadModulePath=...`
   - `LoadModule=...`
   - `LoadModule=...`
   - `...`
2. restart Zabbix daemon

Frontend

1. create item
2. choose **Type**
   - for server and proxy modules: **Simple check**
   - for agent modules: **Zabbix agent** or **Zabbix agent (active)**

History export is enabled automatically if loaded module exports `zbx_module_history_write_cbs()` function

Share your results

Share your results

Proud of your module?

Upload it or post a link to your site at
`https://share.zabbix.com/`

Share your results

Proud of your module?

Upload it or post a link to your site at
`https://share.zabbix.com/`

Looking for a way to extend Zabbix?

Find a selection of brilliant modules (and other stuff) on
`https://share.zabbix.com/`

Share your results

Proud of your module?

Upload it or post a link to your site at
`https://share.zabbix.com/`

Looking for a way to extend Zabbix?

Find a selection of brilliant modules (and other stuff) on
`https://share.zabbix.com/`

Need help developing your dream module?

Zabbix Team can help
`http://www.zabbix.com/development_services.php`

What are you waiting for?

What are you waiting for?

Step 1:

take Zabbix

What are you waiting for?

Step 1:

take Zabbix

Step 2:

add few modules