

# Program or script execution

Program / script execution probe runs executable file specified in Executable file field with arguments specified in Arguments field and returns data from `stdout` and `stderr` streams. Depending on the selected probe type, the number of fields required to fill in is different:

- Program/script in file system  
in the field "Executable file" specify binary name or script full path;

The screenshot shows the 'Monitoring' configuration form for a 'Program / script execution' probe. The 'Subtype' is set to 'Program/script in file system'. The 'Executable file' field contains the path '/home/saymon/scripts/arguments.sh'. The 'Arguments' field is empty. The 'Timeout' field is labeled 'Execution timeout in seconds'. The 'Period' is set to 10 seconds.

- Script from repository  
in the field "Script" select the script saved in repository from the drop-down list;

The screenshot shows the 'Monitoring' configuration form for a 'Script from repository' probe. The 'Subtype' is set to 'Script from repository'. The 'Script' field contains the name 'get\_server\_time.sh'. The 'Arguments' field is empty. The 'Timeout' field is labeled 'Execution timeout in seconds'. The 'Period' is set to 'Probe period' seconds.

- Script with text  
in the field "Script" enter the script text using the control buttons.

The screenshot shows the 'Monitoring' configuration form for a 'Script with text' probe. The 'Script' field contains a bash script for checking domain expiration dates. Below the script field are control buttons: 'Apply', 'Revert', 'Add to repository', and 'C'. The 'Arguments' field is empty. The 'Timeout' field is labeled 'Execution timeout in seconds'. The 'Period' is set to 10 seconds.

```
1 #!/bin/bash
2 # Script to check domain name expiration date.
3 # Set the website as an argument.
4
5 WEBSITE=$1
6 if [ -z "$WEBSITE" ]; then
7 echo "Please type the website you are up to check."
8
9 exit 1
10 fi
11
12 # Find out the expiration date.
13 EXP_DATE=$(whois "$WEBSITE" | egrep -i 'Expiration|Expires on|Expiry date|paid-till' | head -1 | sed -e 's/^[^:]*://' -e 's/|/ /g' -e 's/T.*Z//g')
14 DATE=$(date --date="$EXP_DATE" +%d %m %Y)
15 # Time arithmetics.
```

For each script subtype, you can specify additional arguments as necessary.

If one of the arguments is a string with spaces, you need to specify each argument in its own field:

Monitoring

Agent: Slaymon Agent Staging (ID:817) hostNMS ServerSlaymon Agent Staging

Probe type: Program / script execution

Subtype: Program/script in file system

Executable file: curl

Arguments: Arguments

Timeout: http://sky.ru status, http://sky.ru resptime

Period: seconds

If executable file returns JSON-formatted string, e.g.:

```
{"cpu": "10", "mem": "20"}
```

it is automatically parsed to a table with stdout.cpu and stdout.mem columns with 10 and 20 values:

stdout.cpu	stdout.mem
10	20

For the multiline tables you need to change JSONdata like this:

```
{  
  "host1": {  
    "cpu": "10",  
    "mem": "20"  
  },  
  "host2": {  
    "cpu": "30",  
    "mem": "40"  
  }  
}
```

In this case you will get stdout.host1.cpu, stdout.host1.mem, stdout.host2.cpu and stdout.host2.mem columns:

stdout.host1.cpu	stdout.host1.mem	stdout.host2.cpu	stdout.host2.mem
10	20	30	40

Just choose "stdout" in "Table for field" dropdown list in the [Agent data](#) section header and getmultilines:

cpu	mem
10	20
30	40

You may also wish to add additional field "host" to your data:

```

{
  "host1": {
    "host": "1",
    "cpu": "10",
    "mem": "20"
  },
  "host2": {
    "host": "2",
    "cpu": "30",
    "mem": "40"
  }
}

```

to get more visual clearness:

host	cpu	mem
1	10	20
2	30	40

And another one good example of JSON-data:

```

{
  "MEM": {
    "memoryType": "MEM",
    "bytesTotal": 4130643968,
    "bytesUsed": 3002249216,
    "bytesAvailable": 1128394752,
    "percentUsed": 72.68235266119164
  },
  "SWAP": {
    "memoryType": "SWAP",
    "bytesTotal": 536866816,
    "bytesUsed": 469790720,
    "bytesAvailable": 67076096,
    "percentUsed": 87.50600819403225
  },
  "TOTAL": {
    "memoryType": "TOTAL",
    "bytesTotal": 4667510784,
    "bytesUsed": 3472039936,
    "bytesAvailable": 1195470848,
    "percentUsed": 74.38740040841435
  }
}

```

Examples of scripts for \*nix:

```
#!/bin/sh
# Example of stdout output
# Search for TEST.sh script running
echo `ps -ef | grep "TEST.sh" | grep -v grep | wc -l`
#!/bin/sh
# Example of stdout JSON-output
# Search for TEST.sh script running
TEST=$( ps -ef | grep "TEST.sh" | grep -v grep | wc -l )
echo {"TEST": "$TEST"}
```

Examples of scripts for Windows:

```
@echo off
REM Example of stdout output
REM Search for RDP service running
for /F "tokens=*" %%i in ('tasklist.exe /svc ^| find /c
"TermService"') do set TERMSRV=%%i
echo %TERMSRV%
@echo off
REM Example of stdout JSON-output
REM Search for RDP service and test.cmd file running
for /F "tokens=*" %%i in ('tasklist.exe /svc ^| find /c
"TermService"') do set TERMSRV=%%i
for /F "tokens=*" %%i in ('tasklist.exe /v ^| find /c "test.
cmd"') do set TEST=%%i
echo {"TERMSRV": "%TERMSRV%", "TEST": "%TEST%"}
```