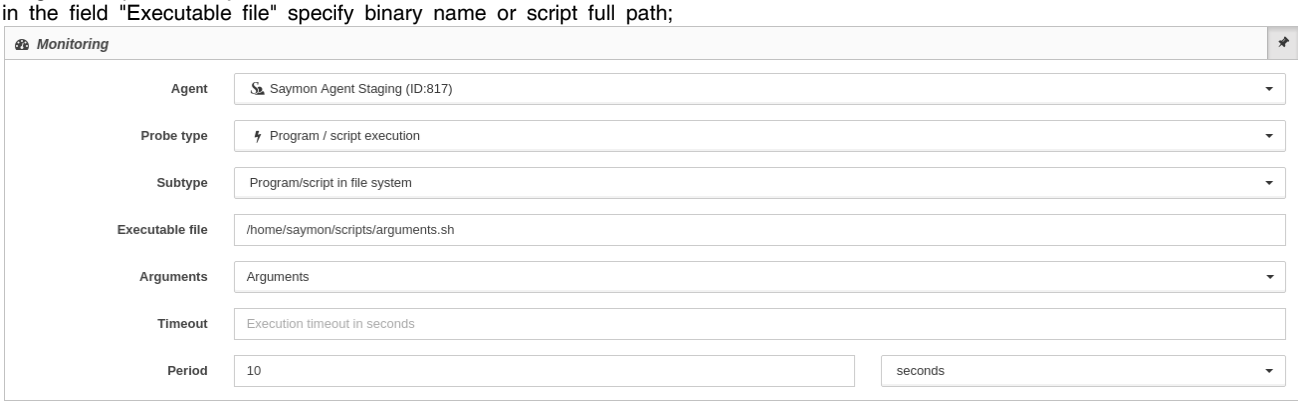
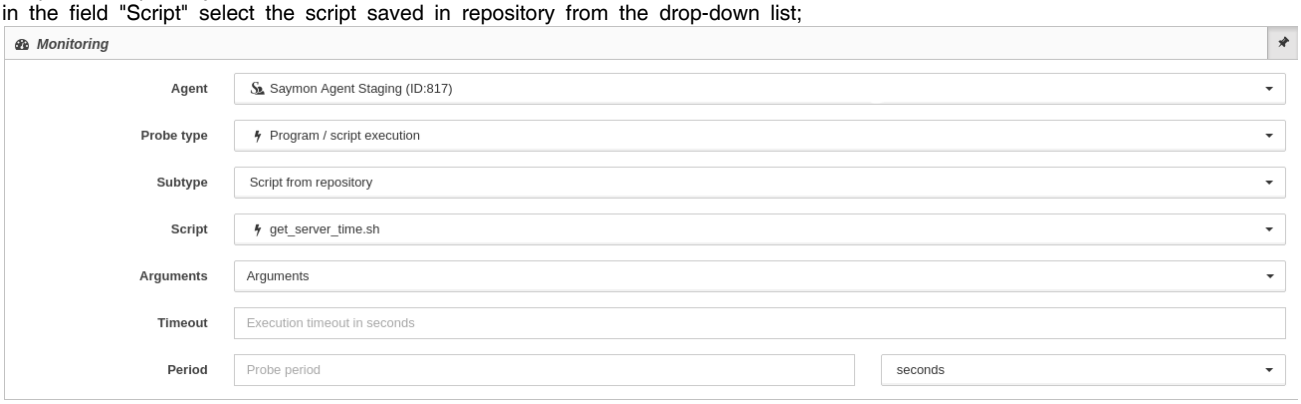
Program or script execution

`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 numer of fields required to fill in is different:

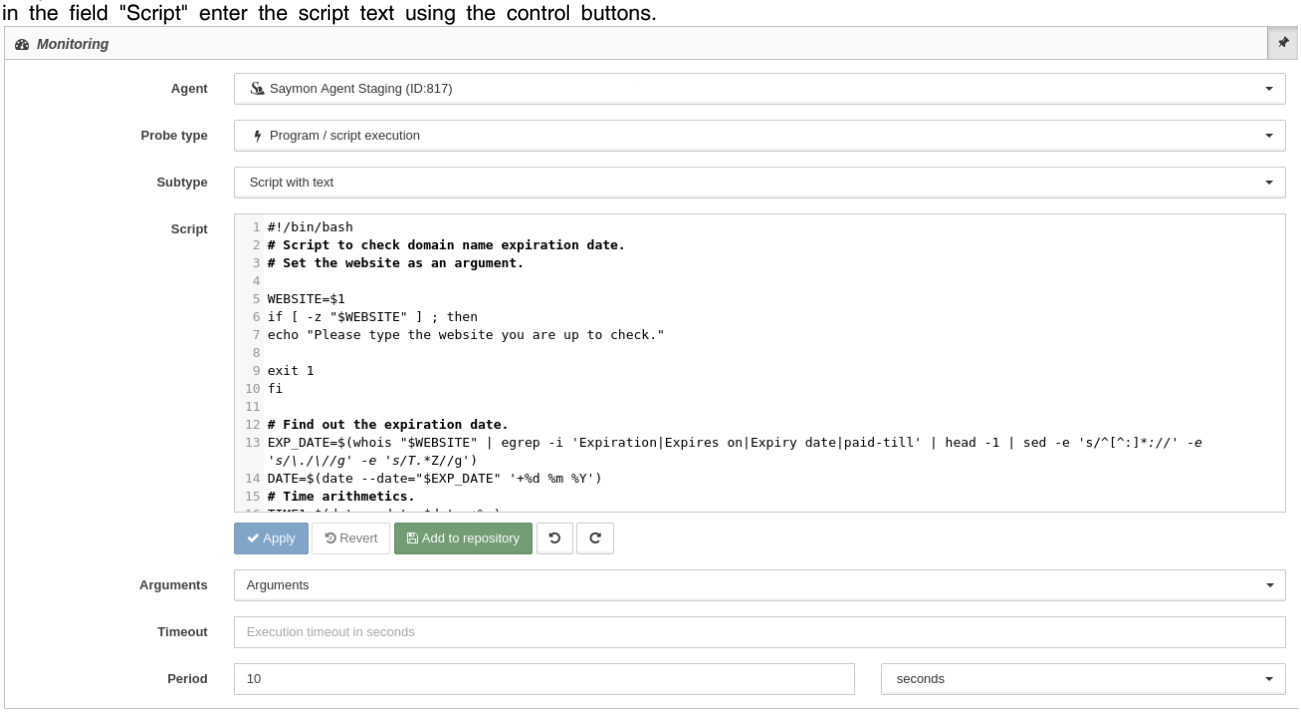
- Program/script in file system
  in the field "Executable file" specify binary name or script full path;



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

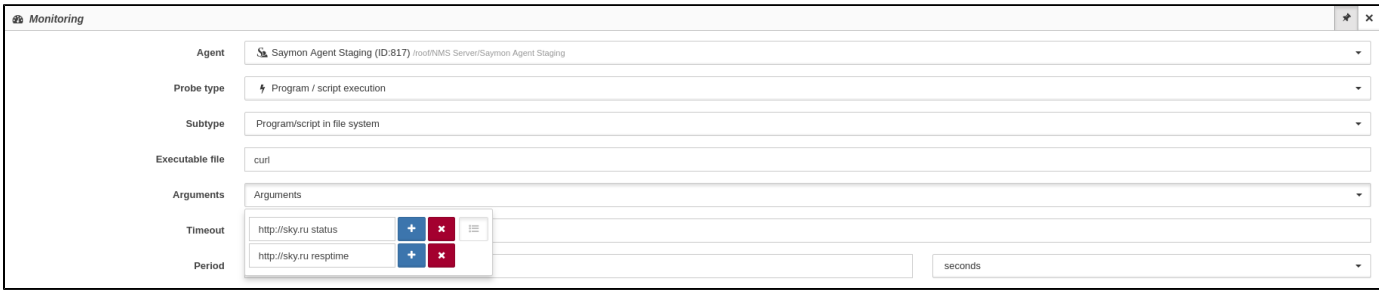


- Script with text
  in the field "Script" enter the script text using the control buttons.



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:



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.memcolumns 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 headerand 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%"}
```