



INNOVATIVE ECONOMY
NATIONAL COHESION STRATEGY

UNICORE MONITORING INFRASTRUCTURE PROBES ADMINISTRATOR AND DEVELOPER GUIDE

Mariusz Strzelecki

Document Version:	1.5
Component Version:	2.3.2
Date:	21 01 2013

This work is co-funded by the EC EMI project under the FP7 Collaborative Projects Grant Agreement Nr. INF50-RI-261611.

This work was co-funded by the PL-Grid project under the EU Grant Agreement Nr. POIG.02.03.00-00-007/08-00.



Contents

1	General Documentation	1
1.1	What's new	1
1.2	Known issues	2
2	System Administrator Documentation	2
2.1	Installation Guide	2
2.2	Configuration Guide	2
2.3	Probes Reference Card	4
3	Developer Guide	4
3.1	API Documentation	4
3.2	Build Documentation	6
4	Component changelog	6
4.1	Version 2.3.2 (2013-01-17)	6
4.2	Version 2.3.0 (2012-11-06)	7
4.3	Version 2.2.1 (2012-08-30)	7
4.4	Version 2.1.0 (2012-04-14)	8
4.5	Version 2.0.10 (2011-10-11)	8
4.6	Version 1.9.9 (2011-08-30)	8
4.7	Version 1.9 (2011-08-12)	8
4.8	Version 1.4 (2011-05-27)	8
4.9	Version 1.3 (2011-01-25)	8
4.10	Version 1.2 (2010-12-01)	9
4.11	Version 1.1 (2010-10-01)	9
4.12	Version 1.0 (2010-06-23)	9

This document describes part of the [Nagios-based UNICORE Monitoring Infrastructure](#) developed in [PL-Grid](#) and [EMI](#) projects.

1 General Documentation

UNICORE Monitoring Infrastructure Probes (shortly: UMI-Probes) is a package that consists of scripts that can be used to test the functionality of each "main" [UNICORE](#) components:

- UNICORE Gateway,
- UNICORE Registry,
- UNICORE/X (including CIP component),
- UNICORE SMS implementations,
- UVOS,
- UNICORE Workflow Factory,
- UNICORE Service Orchestrator,
- UNICORE Common Information Service,
- UNICORE StorageFactory,
- UNICORE accounting system message broker (ActiveMQ).

Additionally, a script that checks appropriate functionality of any application installed in the Grid environment is available.

All scripts can be easily used as probes in Nagios-based monitoring environment. Each of them presents the result of the test in well-known [Nagios probes format](#) and is compatible with [Nagios probes development standards](#). The main programming languages of probes are Perl, Java and Groovy. The majority of scripts are dependent on availability of standard UNICORE clients: UNICORE Commandline Client and UVOS Commandline Client (they are available in the newest EMI release) and some Perl modules: perl-Error, perl-Sort-Versions, perl-XML-RSS (all available via CPAN).

1.1 What's new

In 2.3.2 version of probes:

- ability to check exit code of a job in check_application has been added
- unknown error in check_workflow has been fixed
- unknown error in check_workflow when workflows limit is reached has been fixed
- unknown error when unable to create temporary file because of file existence has been fixed
- unknown error in check_workflow when registry.listServices() fails has been fixed

1.2 Known issues

- Probes package is dependent upon the UCC package in specific version. Most probes are written in Groovy using UCC internals and UCC still has not a stable API - probes may work with newer UCC package that was specified in the spec file but this cannot be assured.
- This release is not backward compatible to previous ones because of changes in configuration files syntax (due to http://sourceforge.net/tracker/index.php?func=detail&aid=3554834&group_id=248204&atid=218115).

2 System Administrator Documentation

UMI-Probes component is available as rpm package, available to install in Red Hat-compatible systems and as deb package to install on Debian-compatible systems. The following guides are ensured to work on Scientific Linux 5.5, Fedora 14 and Debian 6.

2.1 Installation Guide

As mentioned above, probes are written in Perl, Java and Groovy and use UNICORE standard clients. Assuming that EMI repository is enabled, component can be installed using simple command:

```
# yum install unicore-nagios-plugins
```

This package provides:

- Nagios commands configuration in `/etc/unicore/monitoring-probes/commands.cfg`
- Documentation and Licence in directory `/usr/share/doc/unicore/monitoring-probes/`
- Probes in directory `/usr/libexec/grid-monitoring/probes/pl.plgrid/UNICORE/`

Each probe is placed in directory named as probe itself and consists of main test program (executable script named as probe with `.pl` extension), readme in text format (with `.README` extension) and readme in html format (with `.html` extension). There can be other files (groovy scripts or java classes) that are internally used by tests.

2.2 Configuration Guide

Each probe needs appropriate configurations:

- configuration of related UNICORE client,

- logging configuration for probe,
- configuration of probe itself.

All configuration-making processes can be easily automated by using package UMI-Autoconf, released by PL-Grid project.

Samples of UNICORE **clients configurations** are attached in their packages and has to be changed to be able to connect to the Grid. This is recommended to test prepared clients configuration by executing commands:

```
$ uvos-clc -b getMyIds
$ ucc list-sites
$ ucc list-storages
$ ucc run /usr/share/doc/unicore/ucc/samples/date.u
$ ucc workflow-submit \
/usr/share/doc/unicore/ucc/samples/workflows/date-with-stageout.swf
```

If all of this commands ends without any error that means that clients configuration is ready to use with probes.

The second step is to prepare log4j **logging configuration**. This is standard log4j configuration file that will be used by UNICORE clients (both UCC and UVOS CLC are written in Java), samples are placed in files `/etc/unicore/ucc/logging.properties` and `/etc/unicore/uvos-clc/log4j.properties`. Each probe needs two configuration files: for standard execution and for debug purposes. There are strict naming conventions: files has to be named `log4j-[clientname].properties` or `log4j-[clientname]-debug.properties` (where `[clientname]` is `ucc` or `uvosclc`). Every probe at each run looks for appropriate logging configuration to use in directories in the following order:

1. location of probe configuration,
2. location of UNICORE clients configuration,
3. location of logging directory of each probe (the least recommended way).

If configuration is not found, probe will not start and will display suitable message.

Finally, the third type of configuration is **probes configuration**. Every probe gets its "what is to be tested" information from configuration file. In some cases few probes can use the same configuration file (especially if they are used to monitor one grid site). The structure of file is pasted below:

```
# Comments need to be started with hash
# Comment

UCC_PATH="/usr/bin/ucc"
```

```
# Above line means that variable UCC_PATH is set to /usr/bin/ucc
# (quotes are mandatory)

LOGS_DIR="/var/log/unicore/monitoring/icm.edu.pl"
```

In each probe configuration there is a section that describes what values need to be set in each probe configuration file.

2.3 Probes Reference Card

All probes are written using Nagios probes standard. That means that every probe

1. uses Perl as the main programming language (but disables the usage of Nagios embedded perl),
2. has definable directory for storing logs and temporary files,
3. has the ability to set timeout for probe execution (option `-t` or `--timeout`)
4. has the ability to set verbosity level (option `-v` or `--verbosity`) to one of
 - 0 → prints only one line with status,
 - 1 → default, prints line with status with optional debug info,
 - 2 → prints data like `-v 1` and additionally information about probe environment (configuration parsing, client running, debug info from UCC),
 - 3 → prints data like `-v 2` and disables deletion of temporary files after even successful execution,
5. shows readme with `-h` or `--help` flag given,
6. shows version of every probe with `--version` option,
7. puts shell command into log file before execution.

Descriptions of all the probes are attached into next sections:

3 Developer Guide

3.1 API Documentation

All described probes use one-file Perl library that makes writing new ones quite easy. This file is located at `umi2/commons.pm` and consists of several "public" (exported) functions:

- `exit_plugin` - the most preferred way of exiting probes. Takes two arguments. The first is status line in format `[STATUS]: [message]` where `[STATUS]` is one of: `OK`, `WARNING`, `CRITICAL`, `UNKNOWN`. This line will be shown as probe output in every verbose mode (and of course appropriate exit code of script will be set to meet Nagios API requirements). The second argument are optional debug data. Provided string will be evaluated and displayed by probe in first verbose mode.
- `setup_plugin` - this function has to be called at the beginning of probe execution. It takes two parameters: location of readme file (to display its fragment as help message if `--help` option is provided) and version of probe (that will be displayed when a user calls script with `--version` flag). Procedure gets options from the commandline and stores it in external `%config` variable. Next sets timeout of probe to value specified by a user in command line or 300 seconds by the default. Then loads configuration file and also stores the data in `%config` hash. Finally changes working directory to `[LOGS_DIR]/[plugin_name]` - that is place where log files and temporary files will be stored.
- `message` - shows message to user according to requested verbose mode. Takes two parameters - one is message to display and the second is the least verbose mode to attach this message to the output.
- `check_conditions` - checks some conditions and if any is met, exits probe. Takes one argument - array of conditions. Every element is hash with three keys: `test`, `output` and `show_debug`. First, `test`, is the logical condition that will be evaluated. If evaluation gives false message (empty string or 0) function tests next one. Otherwise it calls `exit_plugin` subroutine with `output` and `show_debug` parameters. If `show_params` is 0 then debug messages will not be shown.
- `create_temp_file` - creates a temporary file with name provided as first argument. The file is stored in current directory (set by `setup_plugin`) and is saved to be deleted at the end of probe execution.
- `check_config` - checks if variable required by script execution is available in configuration file. Takes two parameters: first is a comma-separated list of configuration variables. If any of them is not given in configuration, probe exits with `UNKNOWN` status. But if there is a second (optional) parameter defined, the function does not quit probe and just returns a number of undefined options.
- `run` - executes external command with timeout checking. The command can be one of: `ucc`, `uvosclc`, `java` and is given as first argument. The second parameter is a line of arguments to be passed to command (configuration files for UNICORE clients and Registry URL for UCC are included in command line by script). The third argument is path to the file where output will be saved (it is preferred to pass path returned by above described `create_temp_file` or just `/dev/null` if probe output does not matter). The fourth parameter is optional and should be set if `stderr` stream has to be attached to output (if verbose mode is more or equal 2 this flag is set by default). Both UCC and UVOS CLC are executed with appropriate environment variables that sets path to `log4j` properties file.
- `check_file_existence` - checks if file passed as the first parameter exists in file system. Can be easily used if developer is not sure if script that is to be executed is properly defined.

- `is_debug_enabled` - returns if verbose mode is more or equal two.

Additionally there some other options of library that developers may need:

- `$main::CLEANUP` variable - if set, it is executed at the end of probe execution. Can be set to for example clear Grid objects at the end of each execution (see `check_application` source code).

3.2 Build Documentation

Build of component is done by UNICORE packman tool (in fact modified version of packman, see `packaging/packman-opts.xml` file). There are three main targets of packaging script:

- `./packaging/packman.sh probes-clean` - deletes `.class` files and temporary build workspace directories
- `./packaging/packman.sh probes-compile` - compiles two `.java` classes (used by `check_uvos` and `check_gateway`)
- `./packaging/packman.sh all-rpm` - packages component into four files: binary rpm, binary tar, source rpm and source tar.

Documentation is built using UNICORE docman tool. This can be run by command: `./packaging/docman.sh`.

4 Component changelog

4.1 Version 2.3.2 (2013-01-17)

Bug fixes:

- Ability to check exit code of a job in `check_application` (https://sourceforge.net/tracker/?func=detail&atid=2181153&aid=3597723&group_id=248204)
- Unknown error in `check_workflow` (https://sourceforge.net/tracker/?func=detail&atid=2181132&aid=3591380&group_id=248204)
- Unknown error in `check_workflow` when workflows limit is reached (https://sourceforge.net/tracker/?func=detail&atid=2181132&aid=3597746&group_id=248204)
- Unknown error when unable to create temporary file because of file existence (https://sourceforge.net/tracker/index.php?func=detail&aid=3600974&group_id=248204&atid=2181132)
- Unknown error in `check_workflow` when `registry.listServices()` fails (https://sourceforge.net/tracker/index.php?func=detail&aid=3601182&group_id=248204&atid=2181132)

4.2 Version 2.3.0 (2012-11-06)

FRs:

- multiple protocols support in check_sms (https://sourceforge.net/tracker/?func=detail&aid=3566964&group_id=248204)

Bug fixes:

- check_certificate with openssl 0.9.8 (https://sourceforge.net/tracker/?func=detail&aid=3571431&group_id=248204&atid=2181132)
- unknown error when export fails in check_workflow (https://sourceforge.net/tracker/index.php?func=detail&aid=3571441&group_id=248204&atid=2181132)
- unknown error on get-file error in check_sms (https://sourceforge.net/tracker/?func=detail&aid=3572287&group_id=248204&atid=2181132)
- unknown error in check_workflowservice connection problems (https://sourceforge.net/tracker/index.php?func=detail&aid=3579125&group_id=248204&atid=2181132)
- unknown error on check_servorch connection problems (https://sourceforge.net/tracker/index.php?func=detail&aid=3579125&group_id=248204&atid=2181132)

4.3 Version 2.2.1 (2012-08-30)

Next EMI release, changes:

- https://sourceforge.net/tracker/index.php?func=detail&aid=3554834&group_id=248204&atid=2181153
- https://sourceforge.net/tracker/index.php?func=detail&aid=3554827&group_id=248204&atid=2181153
- http://sourceforge.net/tracker/?func=detail&aid=3554818&group_id=248204&atid=2181153
- http://sourceforge.net/tracker/?func=detail&aid=3554823&group_id=248204&atid=2181153
- https://sourceforge.net/tracker/index.php?func=detail&aid=3488648&group_id=248204&atid=2181132
- http://sourceforge.net/tracker/?func=detail&atid=2181132&aid=3554805&group_id=248204
- http://sourceforge.net/tracker/?func=detail&aid=3554807&group_id=248204&atid=2181132
- http://sourceforge.net/tracker/?func=detail&aid=3554808&group_id=248204&atid=2181132
- http://sourceforge.net/tracker/?func=detail&aid=3554813&group_id=248204&atid=2181132
- http://sourceforge.net/tracker/index.php?func=detail&aid=3554817&group_id=248204&atid=2181132
- http://sourceforge.net/tracker/?func=detail&aid=3554824&group_id=248204&atid=2181132
- http://sourceforge.net/tracker/?func=detail&aid=3554828&group_id=248204&atid=2181132
- http://sourceforge.net/tracker/?func=detail&aid=3512706&group_id=248204&atid=2181132

4.4 Version 2.1.0 (2012-04-14)

In 2.1.0 version the check_storagefactory probe has been added. Some bugs has been fixed:

- malfunction in check_freespace where there are multiple home storages
- unnecessary creation of ucc.log files where running tests
- check_servorch malfunction in new service orchestrator
- missing TD support in check_workflow

Sourceforge links:

- http://sourceforge.net/tracker/?func=detail&aid=3484915&group_id=248204&atid=2181132
- http://sourceforge.net/tracker/?func=detail&aid=3484919&group_id=248204&atid=2181132
- http://sourceforge.net/tracker/?func=detail&aid=3488648&group_id=248204&atid=2181132
- http://sourceforge.net/tracker/?func=detail&aid=3512706&group_id=248204&atid=2181132

4.5 Version 2.0.10 (2011-10-11)

First official EMI release. All probes are tested (by unit- and functionality tests), documentation is reviewed.

4.6 Version 1.9.9 (2011-08-30)

- finished refactoring of all probes, outdated documentation of package

4.7 Version 1.9 (2011-08-12)

- completed ETICS packaging scripts, first EMI release

4.8 Version 1.4 (2011-05-27)

- added new probes: check_freespace, check_versions, check_cip, check_cis

4.9 Version 1.3 (2011-01-25)

- finished refactoring of 4 probes: check_gateway, check_uvos, check_workflow and check_workflowservice
- all UMI scripts separated into two sets: UMI-Probes and UMI-Autoconf

4.10 Version 1.2 (2010-12-01)

- check_servorch probe for checking Service Orchestrator

4.11 Version 1.1 (2010-10-01)

- check_workflow:
 - checking availability of required or unused TSFs
- installation:
 - new format of dependency tree (included in samples)
 - faster problem finder (use of Nagios predictive check, smaller intervals)
 - automated generation of Logger services for each probe
- documentation:
 - added separate readme files for each probe
 - attached documentation in PDF format

4.12 Version 1.0 (2010-06-23)

First UNICORE Monitoring Infrastructure package public release