Table of Contents

UNICORE Monitoring Probes Test Plan

EMI Product Description and Version

This test plan covers the set of Nagios plugins used to test UNICORE services in 2.2.1 version.

The overall documentation of the product is available in UNICORE Monitoring Infrastructure Probes Administrator and Developer Guide.

The following sections describe all types of the provided tests:

1. Unit tests
2. Deployment and configuration tests
3. Basic functionality tests

Unit tests

The main language of probes is Perl, but they use Groovy and Java. Probes are simple structural scripts that cannot be unit-tested. Still, every script uses common library functions (that setups enviroment, creates and deletes temp files, runs UNICORE clients, etc.) that are unit-tested using Perl Test::More API. There are 44 tests that can be run using the command:

```
perl test/umi2commons.pl
```

Additionally, there are written tests that checks if Groovy files are OK (Groovy is an interpreted language and there is a need to test if UNICORE API has not been changed when testing probes with new versions of UNICORE libraries). This is done using Groovy-to-Java compiler, all work is done by executing the script:

```
sh packaging/testgroovy.sh
```

Deployment and configuration tests

Yum can be used to install the described package:

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

To install the package the EMI repository must be enabled.

For upgrading package from version that was released previously, run yum update:

```
yum update unicore-nagios-plugins
```

Configuration

Test that is done by each probe is defined using simple file, named "probe configuration file". Readme file of each probe contains a section "Configuration" which describes what configuration variables are needed to appropriate execution of the probe. The script will exit with "UNKNOWN" status unless all variables are configured properly. For example:

Readme file of check_sms consist configuration section with the following content:

- UCC_PATH: absolute path to UNICORE Commandline Client binary (in version 1.4.0 or higher)
- UCC_CONFIG: absolute path to valid configuration file for UCC (remember to set absolute paths to keystore and trustore in this file!)

- REGISTRY_URL: address of registry that contains SMS you want to check
- LOGS_DIR: directory where the script should store UCC logs and temporary files
- SMS_ADDRESS: URL of SMS to be checked, can be given in any ucc-acceptable form, i.e. https:// URL or u6:// meta-URL.
- FILE_SIZE_KBS: size of file used to test (default: 1000 Kbytes)

That means that an administrator needs to prepare a file with content:

```
UCC_PATH="/usr/bin/ucc"
UCC_CONFIG="/etc/nagios/ucc.config"
REGISTRY_URL="https://127.0.0.1:8080/REGISTRY/services/Registry?res=default_registry"
LOGS_DIR="/var/log/unicore/"
SMS_ADDRESS="https://127.0.0.1:8080/SE-SITE/services/StorageManagement?res=default_storage"
FILE_SIZE_KBS="1000"
```

Basic functionality tests

Basic functionality tests are executed on local UNICORE system deployment (including Gateway, Registry, UNICORE/X, UVOS and Workflow System). This deployment is realized by a set of scripts that automates: downloading packages, configuring them and executing tests.

Before running the functionality tests there is need to download and autoconfigure servers:

```
cd test/deplyment; make all; cd ../..
```

Then tests can be executed using:

```
./test_all.sh
```

The expected output is the result of all passed tests.

Basic tests cover all positive results of the tests and majority of negative results:

- test for check_gateway
    - normal workflow: probe checks for properly working gateway
    - error cases:
        - check if returning state is WARNING if there is some service missing
        - check if returning state is CRITICAL if there is unable to connect server
- test for check_uvos
    - normal workflow: probe checks for properly working UVOS server
    - error cases:
        - check if returning state is CRITICAL if UVOS returns "Access denied" information
        - check if returning state is CRITICAL if there is unable to connect UVOS server
- test for check_registry
    - normal workflow: probe checks for properly working registry
    - error cases:
        - check if returning state is CRITICAL if some services are missing
        - check if returning state is CRITICAL if none services are found
        - check if returning state is CRITICAL if there is unable to connect to the registry (server is down)
- test for check_unicorex
    - normal workflow: probe checks for properly working UNICORE/X
    - error cases:
        - check if returning state is CRITICAL if there is unable to connect to the UNICORE/X (server is down)
- test for check_cip

        - error cases:
            - check if returning state is CRITICAL if there is unable to connect CIP (UNICORE/X server is down)
            - check if returning state is WARNING if CIP does not process XNJS information (jobs info)
            - check if returning state is WARNING if CIP has not been initialized during UNICORE/X startup
- test for check_application
    - normal workflow: checks if the job that uses application Date in version 1.0 is properly submitted, finished ant output checked with condition string "not_empty(#stdout)"
    - error cases:
        - check if probe returns WARNING state if condition cannot be fulfilled
        - check if probe returns UNKNOWN state if condition cannot be parsed
        - check if probe returns CRITICAL state if there is unable to connect UNICORE/X
- test for check_storagefactory
    - normal workflow: checks if probe is able to create Storage using SF service, if created storage is able to file upload and download and if it's can be destroyed
    - error cases:
        - check if probe returns CRITICAL state if there is unable to connect UNICORE/X (StorageFactory containter)
- test for check_sms
    - normal workflow: checks if the probe is able to test Storage Management Service by uploading and downloading file are the result is OK state
    - error cases:
        - check if probe returns CRITICAL state if there is unable to connect UNICORE/X (SMS containter)
        - check if probe returns CRITICAL state if there SMS is not properly configured (lack of filesystem write rights)
- test for check_freespace
    - normal workflow: checks if the probe gets number of free space on Storage Management service and properly sets OK state
    - error cases:
        - check if returning state is WARNING if high "w" thereshold is given
        - check if returning state is CRITICAL if high "c" thereshold is given
        - check if returning state is UNKNOWN when SMS connection cannot be established
- test for check_workflowservice
    - normal workflow: checks if the probe is able to contact WorkflowFactory and retreive number of workflows, should return OK state
    - error cases:
        - check if returning state is CRITICAL it probe is unable to connect workflow server
- test for check_servorch
    - normal workflow: checks if the probe is able to get information from GRIS and set OK state
    - error cases:
        - check if returning state is CRITICAL it probe is unable to connect GRIS service
        - check if returning state is CRITICAL it GRIS service does not have information of any resources

Still not covered probes: check_versions, check_workflow, check_activemq.

This topic: EMI > UNICOREMonitoringProbesSVVP
Topic revision: r8 - 2012-09-03 - unknown