UNICORE TSI: MANUAL

UNICORE Team

Document Version:	1.0.0
Component Version:	6.4.0
Date:	19 Apr 2011

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

Contents

1 Overview

The TSI is one point where UNICORE's seamless model meets local variations and so will need to be adapted to the target system.

The TSI is a Perl script (requires Version 5.004 or later). If you must use a previous version of Perl, then you should read and change the first few lines of the "tsi" file.

The TSI performs the work on behalf of UNICORE users and so must be able to execute processes under different uids and gids. Therefore, in production it must be run with sufficient privileges to allow this (during development and test it can be run as a normal user).

2 Installation

Several TSI implementations are available in the sub-directory "./tsi". Additional TSIs for other environments are available in sub-directory "./tsi_contrib". These are either not tested or are intended for older batch systems. They are provided because they might still be useful for somebody. As a rule of thumb you can assume that those older versions require some modifications in order to work with the current version of UNICORE - please contact the support mailing list and we can provide help.

The NOBATCH TSI is used when no batch sub-system is present. It needs also all files common to all installations from directory ./tsi/SHARED. The specific TSIs with batch sub-system are composed of all common files from ./tsi/SHARED plus the files for the specific operation system and/or batch sub-system, e.g. ./tsi/aix_ll

Execute the installation script Install.sh and follow the instructions to copy all required files into a new TSI installation directory.

After adaptation of the "tsi" file you should execute to set restricted file permissions for the TSI files and directory

The TSI uses the auxiliary script tsi_ls to list files. This is supplied with the TSI code and the NJS configuration needs to be changed so that the tsi_ls file in your TSI installation can be found. This is done by specifying the full path to tsi_ls in the configuration file, which depends on the version of UNICORE you are using:

UNICORE 6 xnjs.xml:

```
<property name="CLASSICTSI.TSI_LS" value="/my_full_tsi_path/perl/tsi_ls"/>
```

UNICORE 5 IDB file:

```
-DEFINE TSI_LS /my_full_tsi_path/perl/tsi_ls
```

Similarly, a "tsi_df" file is used by UNICORE 6 to report the free disk space. In the UNICORE 6 xnjs.xml file, enter:

```
<property name="CLASSICTSI.TSI_DF" value="/my_full_tsi_path/perl/tsi_df"/>
```

An essential task of the installation process is the correct setting of the file permissions which is described in the following paragraph

3 File permissions

The permissions on the TSI Perl files should be set to read only for the owner. As the TSI is executed as root you should never leave any of these files (or the directories) writable after any update.

Note, however, that the tsi_ls and tsi_df files must be world readable (the directory permissions must also be set to world executable), because it has to be read from any user id when executing a ListDirectory request.

The recommended permissions are set by executing the command been generated by a previous call of Install.sh.

In particular, Install_permissions.sh sets the file permissions to world readable for tsi_ls and world executable for the tsi_installation_directory. However, this is not sufficient. All parent directories of tsi_installation_directory have to be world executable as well (world readable is NOT required). For this reason, a short path to the TSI might be preferable.

4 Configuring

The TSI can be configured by editing the TSI files. Basic configuration is done in the conf/tsi.properties file. The further configuration has been concentrated into the "tsi" file and the part of this file that should be changed is clearly marked. This includes the locations of the commands to interact with the BSS and communications with the NJS - note that the TSI will listen for contacts from the NJS and also opens contact with the NJS. Additionally you can review the SharedConfiguration.pm file where are additional settings (common to all BSSes/OS TSI variants) which are rarely changed. Again the configuration section is clearly marked there.

Changes outside this part should not be necessary (except for new portings, cf. next paragraph), but if they are made they should be passed on to the UNICORE developers so that they can be incorporated into future releases of the scripts (send mail to unicore-support@lists.sf.net or use the trackers at http://sourceforge.net/projects/unicore).

The necessary changes can be different for different systems and so you should read the first part of your "tsi" file where the required changes are marked and commented.

5 Porting

Most variations are found in the batch subsystem commands, porting to a new BSS usually requires changes to the following files:

```
Submit.pm
GetStatusListing.pm
```

UNICORE 5 only: On certain systems also the following file has to be modified: Endprocessing.pm

sub-directories.

It is recommended to start from a up-to-date and well-documented TSI, e.g. the linux_torque variation. If you have further questions regarding porting to a new batch system, please use the unicore-support or unicore-devel mailing lists (cf section Contact).

6 OPTIONAL: Job Freezing (UNICORE 5 only)

There is a command to control a job on a BSS - FREEZE. This is intended for BSSs that allow a job to release all resources while remaining in the execution queue. This is optional on the TSI, if there is no explicit code, then the HOLD command will be executed in its place. To implement a FREEZE:

1) Edit the "tsi" file and create the "freeze_cmd" variable to execute the command for your BSS, use hold_cmd as the template.

2) Copy the appropriate vesrion of JobControl.pm to your TSI code directory (preserving permissions of the installation)

3) Stop the TSI and restart it (e.g. using the "tsi refresh" command in the njs_admin utility)

Job Status:

A frozen job will have a new state in BSS listings. This can be reported to the NJS from the GetStatusListing.pm module as FROZEN. Modify the appropriate code if you want to report this.

7 Execution model

The TSI has two modes of execution. The first process to be started is the TSI shepherd which will respond to NJS requests and start up TSI workers to do the work for the NJS.

Since the TSI runs with setuid permissions it must authenticate the source of commands as a genuine XNJS. To do this the TSI is initialised with the address of the machine that runs the XNJS together with a port number that the NJS will listen on for TSI worker connections. The TSI shepherd will only accept requests from the NJS machine and all TSI processes will establish connections to the XNJS machine/port.

If the XNJS process dies any TSI workers that are connected to the XNJS will also die. However, the TSI shepherd will continue executing and will supply new TSI processes when the NJS is restarted. Therefore, it is not necessary to restart the TSI daemon when restarting the XNJS.

If a TSI worker stops execution the XNJS will request a new one to replace it.

If the TSI shepherd stops execution, then all TSI processes will also be killed. The TSI shepherd must then be restarted, this does not happen automatically.

8 Directories used by the TSI

The TSI must have access to the directories specified in the IDB to hold Uspaces, Outcomes and Spooled files. These directories are written with the TSI's uid set to the xlogin for which the work is being performed and so must be writable by all xlogins .

9 Starting

If installed from an Linux package, the TSI can be stopped using the init script

```
/etc/init.d/unicore-tsi start
```

The TSI can be started with or without command line arguments.

When executed with command line arguments the format is:

```
perl tsi njs_machine njs_port my_port
```

where the NJS is executing on njs_machine and is listening for TSI worker connections on njs_port (njs_port must match the first port number in the SOURCE entry of the EXECUTION_TSI section in the NJS's IDB file). A TSI process in shepherd mode will listen for NJS requests on my_port (my_port must match the second port number in the SOURCE entry of the EXECUTION_TSI section in the NJS?s IDB file).

Alternatively, the TSI can be started without command line arguments. In this case the variables $main::njs_machine, $main::njs_port, $main::my_port must be set in the tsi Perl file for your system.

As a third alternative, the TSI can be started using the script "start_tsi" (cf. section Scripts).

Depending on the shell used to start the TSI it may be necessary to execute these commands through nohup if you want to log out afterwards.

10 Stopping the TSI

If installed from an Linux package, the TSI can be stopped using the init script

```
/etc/init.d/unicore-tsi stop
```

The TSI shepherd can be killed (preferably using SIGTERM). Since this results in the killing of all TSI processes this should only be done when the NJS has been stopped. However, under Linux it was found that killing the TSI shepherd will not kill the TSI workers.

The TSI can also be killed using the script "kill_tsi" (cf. section Scripts). This will kill the TSI shepherd and the tree of all spawned processes including the TSI workers.

TSI worker processes will stop executing when the XNJS stops executing.

It is possible to kill a TSI worker process but this could result in the failure of a job (but the NJS will recover and create new TSI processes).

11 TSI logging

The TSI can be configured that all the processes return any logging information to the NJS. The TSI daemon writes log information to stdout and stderr, to save these they should be redirected to a file.

12 Scripts

Several scripts are available to simplify the starting (and if needed killing) of the TSI. Before using the scripts it might be necessary to adapt the path to Perl in the scripts.

```
start_tsi [-d] [conf_dir]
```

start_tsi starts the TSI based on the evaluation of the properties file conf_dir/tsi.properties. The properties file determines the path to the TSI, the NJS machine, and the ports for the connections between TSI processes (shepherd and worker) and the NJS. If conf_dir is not specified the current working directory is searched for the properties file. An example file is available in conf/tsi.properties.

The process number of the shepherd TSI is saved in file conf_dir/LAST_TSI_PIDS.

If the TSI does not send its log information to the NJS, it is saved in current date, time, and the port numbers.

Option -d starts the TSI under the interactive Perl debugger.

```
find_pids [conf_dir]
```

find_pids evaluates the process number of the shepherd TSI from file conf_dir/LAST_TSI_PIDS. It shows the tree of all child processes (including the TSI workers) which have been spawned by the shepherd process.

```
kill_tsi [conf_dir]
```

In general, the TSI processes will be stopped through the njs_admin command *tsi stop*. However, there might be situations where this is no longer possible (NJS hangs, . . .). kill_tsi uses find_pids to determine all shepherd and worker processes (and their child processes). Finally all these processes are killed.

```
../bin/list_log_files type [conf_dir]
```

list_log_files is identical to the scripts which are available for the Gateway and the NJS. The script returns the names of all or some of the log files in the default logging directory conf_dir/logs. Please read the corresponding Gateway/NJS documentation for details.

`find_tsi [conf_dir]` **UNICORE 5 only**

find_tsi can be used for the creation of a TSI properties file

It analyses the NJS properties file and the IDB file (which path is found in the NJS properties file). In general, the NJS is executing on a remote machine. For this reason, the NJS properties file and the IDB file are copied first to the TSI configuration directory using *rcp*. Because *rcp* is not available for a privileged user, find_tsi has to be executed as a non-privileged user. However, the generated TSI properties file can be used by a privileged user to start the TSI by the start_tsi script.

find_tsi uses the file conf_dir/path2njs to determine the (remote) address of the NJS properties file. The only entry of path2njs should be a line with syntax: [host:]/directory/where/to/find/the/properties/file/ If the NJS is running on the local system, the host information can be omitted.

The evaluation of the IDB file determines the path to the local TSI files from the definition of TSI_LS in the IDB file.

13 Contact

UNICORE Homepage: http://www.unicore.eu

Support mailing list: unicore-support@lists.sourceforge.net

Developers mailing list: unicore-devel@lists.sourceforge.net (needs registration)