NJS V4.0 CHANGES

Sven van den Berghe, ***Fujitsu Laboratories of Europe***

4.0.3 – 23 July 2003

4.0.2 – 25 April 2003

4.0.1 - 7 February 2003

4.0.0 - 1 November 2002

1 4.0.3 Changes

1.1 New SSO type

Added the configuration option to request "GridFTP" proxy handling (write the SSO to Upsace without declaring the Vsite as Globus type).

1.2 Change to file handling in Uspace

Redid file gathering prior to UPL streaming from Uspace (to another Uspace). Now does in place, without ln to a special directory. *This requires a change to the GetDirectory.pm TSI file.*

1.3 Minor bug fixes

- NJS logger now returns full (canonical) path for log file names.
- Corrected MakePortfolio with an AlternativeUspace so that it changes directory correctly.
- Internal handling of Script type more generic
- Tested for null values in Fortran defines and undefines (pre-processing)
- Added test in InvocationLineDeacon.incarnate for empty deacon as a (very) late test that the IDB is complete.
- Added space between command and command line arguments in ExecuteTasks etc.
- Fixed problem with claiming tickets and resource checking
- Added MKFIFO_CMD to IDB
- Changed example_idb with 0 as minimum for resources
- Properly map NJSs according to NJS property if present.
- Fixed failed fifo transfers (e.g. authorisation) so that jobs stops

2 4.0.2 Changes

2.1 Using FIFOs

The MakeFifo and MoveFifoToOucome AbstractActions are now implemented.

A new version of the GetDirectory.pm TSI Perl script will read from fifos without hanging the NJS.

The Stream Idiomatic AbstractAction has been rewritten and the NJS modified so that a single file can be copied directly from one Uspace to another Uspace. Either the source or destination files (or both) can be fifos.

2.2 Remote NJS administration

The NJS administration commands can now be issued through AbstractActions (only if a site explicitly allows each command and to users in the UUDB with the ADMINISTRATOR role).

See Section 7.5 of the NJS manual.

2.3 TSI Connections

We are planning to change that way that the TSI processes are started up and managed by the NJS. The aim of this is to provide greater security. The basis of this change will be to create TSI processes per user (rather than have a TSI worker process execute scripts for many different users).

This means a change to the way that TSI connections are requested by the NJS, including the AFT and Resource Broker code. The changes are evident in changes to the NJS interface classes (package com.fujitsu.arcon.njs.interfaces).

The TSI change has not been made yet but in anticipation of the change all NJS code should use the new interface methods. The newly deprecated methods will be removed.

The TSI Connection API is also exposed so that it will be possible to implement different TSI classes (e.g to support OGSA/Globus TSIs). This is not yet complete.

2.4 Detecting completed Batch Jobs

The processing at the end of a Batch job has been modified so that systems that are slow to write stdout and stderr files are properly handled.

A batch job is now considered done if it is no longer on the BSS *and* stdout and stderr exist *and* stderr contains the exit status (UNICORE EXIT STATUS).

If the AbstractAction that was incarnated to the Batch job has been aborted or cancelled, then the exit status test is not applied.

In some rare cases where the Batch job finishes abnormally and does not write an exit status the Unicore job will now not terminate until the user aborts the AbstractAction.

2.5 UUDB Interface

The interface to UUDBs has been extended to some NJSs to consign AJOs that they have been created. The ADMINISTRATOR role is now used.

The NJS now allows four roles (com.fujitsu.arcon.njs.interfaces.Role):

USER: can endorse any AJO, can UPL control (consign etc) only AJOs that it has endorsed.

NJS: cannot endorse any AJOs, can consign AJOs endorsed by others, can UPL control AJOs that it consigned.

TRUSTED_NJS: can endorse AJOs (but not those that result in code executing on a TSI), can consign AJOs endorsed by others, can UPL control AJOs that it consigned.

ADMINISTRATOR: As for USER but may be allowed to perform other administration functions.

The FLE UUDB allows these roles to be attached to certificates through strings appended to the Xlogins.:

- “njs-“ for NJS,
- “trusted-“ for TRUSTED_NJS,
- “admin-” for ADMINISTRATOR
- nothing (or, say, “user-“) for USER.

The NJS will interpret USERs whose xlogin is prepended by “njs-“ as TRUSTED_NJSs so that the current FZJ UUDB will work (but, of course, only with Eurogrid certificates and not the new LRZ certificates).

2.6 Miscellaneous

The FLE implementation of the UUDB is now included in the NJS jar file. UUDB distribution has been changed to reflect this. Now assumes Java 1.4 (so

does not reference external libraries for JSSE implementations). Fixed some minor bugs.

Fixed writing to Outcomes of sub-AJOs so that more logging information is returned to the client.

Fixed a problem with the case of the SSL request in the Gateway connection files (in previous V4 releases no SSL could be established between the Gateway and NJS if initialised from a connections file)

Clear AbstractActions completely when they are removed due to Termination Time expiry.

Always clear Uspaces containing Globus proxies.

TSI debug mode will now pass the –v and –x parameters to child scripts.

File names INCLUDEd in the IDB are now processed with any DEFINEd strings.

3 4.0.1 Changes

3.1 Gateway ListVsites

The UPL ListVistes request should return the NJSs that the Gateway provides access to. However, the Arcon NJS has additional, non-NJS, ports that provide other services (such as Alternative File Transfer and the broker operation mode – see below). Previous versions of the Gateway incorrectly returned all the port types in response to a ListVistes.

Version 4.0.1 of the Gateway only returns full NJSs (type NJS or Globus) in reply to a ListVsites.

In addition the Gateway will also recognise a new request class "com.fujitsu.arcon.upl.ListPorts" and reply with a list of all ports that it supports (full Vistes, AFT entry points and Broker NJSs).

3.2 IDB

- INCLUDE keyword to read from another file. The example IDB file has been restructured to make use of this and separate the stable parts ofteh IDB from the parts that are usually modified during installatuion.
- APPLICATION keyword added. This brings together the various definitions required for an application rather than scattering SoftwareResources and INVOCATION definitions all over the IDB. It also allows a grouping of definitions (e.g. a number of different resources required by a single plugin).

3.3 AbstractAction Termination Time

The interpretation of this has been changed to make it compliant with the OGSA notion of service termination. AbstractActions whose Termination Time passes are now CANCELled rather than ABORTed. This means that the AbstractAction and its Outcome will be deleted completely from the NJS.

3.4 Miscellaneous

- The NJS administration interface no longer pauses when the NJS is paused (this means that administration commands can reconnect to paused NJSs).
- New administration command (debug_scripts) to have NJS created wrapper scripts produce verbose, debug, output.
- There is new advice in the IDB about keeping Uspaces after AJO execution within loops (techniques used for debugging installations and new code). See the example IDB supplied with the full installation package.
- The Jar files for NJS, Gateway and AJO classes are now signed.

3.5 Broker only NJS

An NJS can be initialised to run as a Broker only and not execute AbstractActions other than those related to the brokering process. See the documentation for njs.operation_mode in the Using NJS documentation.

3.6 Broker interface

See the NJS interface documentation for more details.

- Added getStorageLocation to return the incarnation of a Storage resource at a Vsite (this is the root directory, a job can add a file name and subdirectories to get to an actual file).
- The Broker has access to the initial resource set of the NJS through the initialisation call. However, the resource will be dynamic at some point, so added getCurrentResources.
- The Broker can generate a UPL ListVsites request to a Gateway (and a ListPorts)
- The Broker can consign AJOs asynchronously (with a callback on completion of the AJO).
- The NJS can call the Broker to CANCEL, ABORT, HOLD or RESUME the processing of a CheckResources through the "apply" method.
- The NJS passes on Tickets claiming some resources to the Broker through the "claim" method.

3.7 AFT Interface

Added new methods to transfer a single from a local Xspace to a remote Xspace (without staging through Uspace).

4 Compatibility

The V4 NJS does not understand 3 AJOs.

It will not accept AJOs from V3 clients or other NJSs still on V3. It will also not load and saved AJOs from a V3 NJS, so the first start after an upgrade must have an empty state directory (pointed to by configuration value **njs.save_dir**)

4.1 Renamed packages

All NJS API code is now in the `com.fujitsu.arcon.njs.interfaces` package. Previous code that used the `unicoreplus`, `olorin.*` and `com.fecit.arcon.njs.interfaces` packages will need to be changed to use the new names.

Some of the interfaces now throw a new exception type.

4.2 IDB

Most of V3.x IDBs are valid.

The IMPORT, EXPORT and COPY_FILE sections have been removed[1] and must be deleted from the IDB before starting a V4 NJS.

The FILE_COPY section in the example IDB must be copied into the IDB file.

The LIST_DIRECTORY incarnation requires a new context (see example IDB)

The following changes should also be made to produce valid incarnations:

- Ensure that all copies include subdirectories e.g.

  ```
  COPY_CMD cp –r
  ```
- Add incarnation rules for the Symbolic Link task (see example IDB)
- IDB LN_CMD should be defined as "ln –s"

4.2.1 Defining storage

The definition of available storage servers has been simplified with the introduction of a new model in the AJO. The IDB now accepts a new keyword,

[1] From NJS 4.0.0 Beta 10

STORAGE, that matches the new storage model. See Section 4.3 of the IDB Document for more details.

The 4.0 NJS accepts the old Storage declarations (and maps them to the new Storage classes in the AJO).

The NJS handles the new version of the Storage resources and does not use the old version. It will return new versions in response to GetResourceDescription. If an AJO contains old versions the NJS will map these to equivalent new versions.

The NJS now uses storages to determine where to execute tasks e.g.

PutPortfolio with a Spool storage will do the same as a Spool.

DeclarePortfolio with and AlternativeUspace resource will declare the Portfolio (and reveal it) in the target uspace.

4.2.2 Checking

The NJS now dies during startup if an INVOCATION in the IDB refers to an undefined CONTEXT (SOFTWARE_RESOURCE)

4.2.3 Absent Script types

Some sites are unable to offer all of the default AJO script types (Bourne shell, Korn shell, C shell and Perl). The NJS now allows this through the following changes:

- The IDB does not *require* invocations for all script types
- The NJS adds a Context to the resources if an invocation is supplied for a particular script type (so letting clients know which script types are supported)
- The NJS checks that the script type requested by a task is supported before trying to execute

4.2.4 Verbatim text

Previous versions of the IDB used "[" and "]" to delimit verbatim text (text copied unchanged from the IDB to an incarnation or a field of a resource object). This version now allows the following extensions (see Section 2.1.4 of the IDB Document):

- Verbatim text may also be start with a double quote. It then ends at the next double quote.
- Where the context is unambiguous, then verbatim text can be delimited by white space (if the first non-space character is not a double quote, a "[" and no qualifier is expected)

4.2.5 Omit dummy definitions

Previous versions of the NJS required a number of empty dummy incarnations in the IDB, These can now be omitted.

4.2.6 Application metadata

Application metadata can be read from a file and inserted into the appropriate fiedl of org.unicore.resources.Application (see Section 4.7 of the IDB Document).

4.2.7 Import

The definition for IMPORTs no longer needs the MODIFY, READONLY and TAKECOPY contexts. These are allowed but only the TAKECOPY context is used.

4.3 TSI

The **tsi_ls, Initialisation.pm** and **EndPrfocessing.pm** files should be updated.

Otherwise no changes are needed.

4.4 Gateway

No changes needed to the Gateway except that it *must* use a 4.0 version of the AJO

However, support for NJS registration is provided in 4.0.0 of the Gateway and so this must be installed if you want to use registration.

New functions

4.5 New njs_admin commands (for TSI)

tsi [up] [down] [status] [stop] [refresh]

Controls the TSI and the NJS view of the TSI status.

See Section 7.4 of the Using the NJS Document.

The **stop** subcommand will only work if the TSI **Initialisation.pm** file is updated.

test_commands xlogin

Run some simple tests to check that the NJS built in Unix commands are properly initialised in the IDB (e.g. that the paths are correct). These tests are not exhaustive.

xlogin is a valid user that will be used to execute the commands.

4.6 "Dynamic" resource updating

njs.dynamic_resource_file=*none*

The name of a file whose contents will be added to the resources advertised by the NJS. The NJS polls this file and update the resource set contents if the file contents change.

This is a simple initial implementation to test how useful this concept is. The only resource type updated is "org.unicore.unicore.TextInfoResoutce". The file's contents are in the Java "Properties" format i.e. :

```
tag = value
```

For example:

```
cpu_load=75%

message=Routine maintenance at 14:00 UTC Today
```

4.7 Registration

NJSs can register with Gateways. If a registration is accepted by a Gateway, then the Gateway will accepts jobs targeted at the NJS and will advertise the NJS's existence to clients.

Registration means that there is less configuration of the Gateways, since NJSs can supply much of the required information (e.g. AJO class version used by the NJS, NJS address). It also means that an NJS can be moved to different machines without having to stop the Gateways.

Registrations must be maintained by an NJS. If they are not renewed after a certain time the Gateway will deregister the NJS. An NJS will automatically maintain all registrations while it is running.

Registration is turned on by setting the **njs.gw_registration** configuration value to true. The Gateways contacted are those on the **njs.gateway** list.

Registrations can be checked and updated at run time by using the **gw_registrar** command of the NJS admin utility.

Gateways need to be configured to accept NJS registration.

4.7.1 New admin commands

gw_registrar [list|delete|update|add]

Administer NJS registrations with Gateways.

This command is only available if the **njs.gw_registration** configuration value is set to **true**.

In the commands below *gateway_uri* stands for a Gateway machine address, plus port number as below:

```
arcon.fle.fujitsu.com:4433
```

list

List the current registration with the time and status of the last registration attempt

delete *gateway_uri*

Delete the Gateway from the list of gateways to register with (this does not cancel a current registration but will not renew it so in time, at most 30 minutes, the Gateway will stop contacting and listing the NJS)

update *gateway_uri*

Register with the Gateway (e.g. if the Gateway has been restarted this will immediately reregister rather than wait for the normal update – up to 15 minutes)

add *getway_uri*

Add the Gateway to the list of registrations.

5 Behaviour changes

Implements AJO 4, including If, IfSuccess, RepeatGroup, ForGroup, WaitUntil, ProbeTime, FileCheck, termination times.

Note that there has been a change to the way that If and RepeatGroup tests are formulated and to the way that data is passed from an executable to the NJS (see the org.unicore.ajo.Decision and org.unicore.outcome.Decidable interfaces in the AJO and Section 3.1)

5.1 Decidable/Decision interfaces

The NJS uses the Decidable/Decision interfaces introduced at AJO 4.0.0 Draft8.

The NJS sets the environment variable UC_DECISION_FILE for all incarnations of ExecuteTasks (and thus LinkTask, FortranTask, UserTask and ExecuteScriptTask).

The NJS reads the file pointed to by $UC_DECISION_FILE at the end of the execution of all ExecuteTasks (whether they end successful or not successful) and sets the Decision string to the file contents.

The NJS automatically creates an instance of org.unicore.resources.Context named “MakeReturnCodeDecision”. If an ExecuteTask contains this in its resource set, then the NJS will incarnate the ExecuteTask with code that writes the incarnation’s return code to the Decision file (for more see the IDB documentation – Section 6.1).

5.2 Less blocking in interactions with TSI

Previous versions of the NSJ have blocked a thread during a complete interaction with the TSI, from trying to get a connection to having the TSI do the required work. This meant that when a TSI was unavailable (usually because the target machine was down) the NJS would seize up and nothing could be done to many AbstractActions (in particulare trying to get the status of a blocked AbstractAction would often also block).

This version of the NJS has changes to the code so that it does not block when trying to get a TSI connection. So if a TSI becomes unavailable blocked actions can be aborted (also TerminationTimes are applied) and getting status should also return (possibly with out of date values).

There have been some changes to external behaviour as a result of this:

- Some com.fujitsu.arcon.njs.interfaces methods may throw a TSIUnavailableException
- An AJO with streamed files will not be accepted if there is no TSI (previously this will have hung in the consign until a TSI was available)
- A RetrieveOutcome where the AJO is done and has files to stream back will fail if there is no TSI (again, previously this will have hung and not returned until the TSI was available)

The NJS sets a new Return Code value in the UnicoreResponse to the ConsignJob or RetrieveOutcome. This value is –2 and the message give the reason (TSI unavailable or TSI busy)

The proper response to all three of these new behaviours is to try again after a suitable delay.

5.3 AbstractJob Vsite instance must be set

The NJS now checks that and AJO is intended for the Vsite (allowing the user to say that she does not mind redirection by setting the MUTABLE type – for Resource Brokering).

This means that the Vsite must be set for all AJOs, the NJS will reject the AJO if it is not set.

5.4 Hidden Portfolios

Portfolios that arrive in a uspace through streaming or a PutPortfolio are now hidden. This means that their files are not in the uspace until they are referred to by a task in the AJO (e.g. DeclarePortfolio).

The files are "revealed" in the subdirectory of the uspace referenced in the USpace storage that the **revealing** task contains.

5.5 Subdirectories

You can now use subdirectories in uspaces. The current working directory of a task is set to the subdirectory found in the USpace resource. All task incarnations should now recursively copy subdirectories (including streaming, see above for required IDB changes).

Note that Portfolios are created in subdirectories. To locate one you need to have the task's cwd set to the subdirectory in which the Portfolio was created.

You can include directory paths in a Portfolio but note that because of the way that Unix cp works if a Portfolio containing, say "dir/file", is copied (e.g to Spool it) the result files do not contain the path (the Spooled Portfolio contains "file"). To preserve subdirectories use just the subdirectory name in the Portfolio (e.g. "dir")

5.6 Overwrite

Overwrite flags are honoured.

5.7 Renaming

To rename a file use CopyFile.

5.8 NJSs can create AJOs

Generally other NJSs are not trusted to execute AJOs. They are allowed to consign and retrieve AJOs created and signed by Unicore users but nothing else. However, to run some advanced Unicore functions (e.g. Resource Brokering) NJSs need to be able to create and run AJOs on behalf of users (e.g. to make concrete requests about available resources). This can be allowed, but the trust must be established individually for each external NJS.

This method of allowing trust applies to the default UUDB shipped with the NJS only. Other implementations of the UUDB may or may not allow this, consult their documentation.

To trust another NJS you need to add its public certificate to the UUDB using the "add_njs" command. An Xlogin is required by the command but never used.

Any attempt to execute code on the TSI by a trusted TSI is disallowed. The only AbstractActions that are allowed are those that execute within the NJS and also consigning sub-AJOs.

6 Job Status Polling

The way that the NJS polls for job status on the BSS can be changed. This form is much less efficient and should only be used if the normal form cannot.

The new behaviour requires changes to the TSI code.

Initiate the new behaviour by changing the IDB keyword "EXECUTION_TSI" to "EXECUTION_TSI_Q". The NJS will now issue request to the code in the "GetStatusListing.pm" TSI file with the following values:

```
#TSI_BSSID <bss id>

#TSI_IDENTITY <user> <project>

#TSI_USPACE_DIR <path to job's uspace dir>
```

The reply from the TSI is in the same format as the normal reply. This is expected to contain at least the status of the requested BSS Id (if it does not, then the job is assumed to have finished execution).

The NJS now also accepts “DONE” as a status from the TSI.