USING THE UNICORE GATEWAY 4.0.1

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

Version 4.0.1, February 11th 2003

1 Changes

1.1 From 4.0.0

- ListVsites now returns full NJSs only. A new request type "com.fujitsu.arcon.upl.ListPorts" lists the full set of ports.
- AFT interface changed to support Xspace to Xspace file transfer

1.2 From V3

- Handles registration from NJSs
- Limits number of active threads
- Packages renamed

1.3 Hints on using proxies

Certificate Revocation Lists are checked using HTTPS. Some sites may require that all HTTP traffic is routed through a proxy server. If this is required, then the following lines should be added to the gateway.properties file:

```
proxySet=true
proxyHost=<proxy machine>
proxyPort=<proxy port number>
```

2 Introduction

The Gateway is a site's point of contact for all Unicore connections. It will accept data from clients and pass it onto the Unicore servers inside the site.

The Gateway filters connection attempts into a site. It will only accept and pass data that originates from acceptable clients. All connections use SSL in authenticated mode. This means that a client must present a valid X509 certificate before a connection is allowed. One of the criteria for a valid certificate is that it is issued by a recognised Certificate Authority. The Gateway can be initialised with a list of Certificate Authorities that it will recognise.

2.1 Terms

Vsite A collection of resources made available through Unicore. This is generally a high-performance computer plus associated resources such as storage. A site can have more than one Vsite. A single Gateway can provide access to more than one Vsite.

NJS (Network Job Supervisor) The Unicore server component that oversees the execution of Unicore jobs for a Vsite.

2.2 Where to put the Gateway

The Gateway should be placed on or in front of your site's firewall. The Gateway machine should have all except essential services removed or disabled. No remote connections should be allowed except to the port(s) that the Gateway uses.

You will need to configure your firewall to allow connections from the Gateway machine to the NJS(s) your site runs (the connections need to be enabled only from a known address – the Gateway machine – to known addresses – the NJSs – on the known port).

3 Configuration

This section describes how to configure a Gateway distribution for your system.

The distribution of the Gateway has the following directory structure. This structure is assumed by the control scripts (but it can be changed if necessary)

```
gateway/
    docs/
        documentation and example code
    conf/
        gateway.properties
        connections (optional)
        logs/
    lib/
        ajo.jar
        gateway.jar
    bin/
        scripts to control the Gateway
```

"conf/" is the Gateway configuration directory

"gateway.properties" is the Gateway configuration file.

"connections" is the file describing the connections that the Gateway can make (the NJSs that it provides access to)

"logs/" is the directory that the Gateway will use for the log files

"lib/" is the directory containing the code used by the Gateway

The Gateway assumes that the files that it uses are stored in the configuration directory (gateway/conf below) and all file paths given in the configuration files are interpreted as are relative to this directory.

Gateway configuration parameters are found in the "gateway.properties" file

3.1 Prerequisites

- An installation of at least Java 1.3.x (1.4.x can be used)
- If you are using Java 1.3.x, then the JSSE (Java Secure Socket Extension) libraries are required. These should be installed in the Gateway lib directory as jsse.jar, jnet.jar and jcert.jar. These libraries are available from java.sun.com.
- A copy of the Unicore AJO classes in the ajo.jar file (this should be supplied with the distribution, updates are available from the site where you got this distribution).
- PKI certificates from your Certificate Authority. You will need a private key for the Gateway (with a certificates for the public key signed by your Certificate Authority). You will also need the public certificates for all the Certificate Authorities that you will accept.

3.2 Set the Gateway contact details

- Set the configuration parameter "gw.gateway_host_name" to the name (or IP address) of the Gateway machine.

  This name must be one that can be addressed by all the clients that will contact the Gateway (including all client NJSs).
- Set the number of the port on which the Gateway listens for client connections using the configuration parameter "gw.port"

3.3 Initialise the list of supported Vsites (NJSs)

You can choose how the list of supported NJSs is generated. This can be either through dynamic registration, statically through a configuration file or a combination of the two.

To allow dynamic registration of NJSs:

- Edit the gateway.properties line starting "gw.named_njs" to include the "name" of the NJSs that you wish to allow to register with the Gateway (remove it if you are willing for any NJS to register).

  The "name" is any substring of the NJSs certificate's Distinguished Name field which contains the name that you supplied when creating the certificate and your organisations name.

To use static initialisation only:

- Set the configuration parameter "gw.connections" to the name of a file in the Gateway configuration directory ("connections" in the default installation).
- Edit this file ("connections") to define the Vsites. One per line with the format:

  `<Vsite name> <NJS machine> <NJS port>`

  Where `<Vsite name>` is the name that clients will use to refer to this Vsite, `<NJS machine>` is the name of the machine on which the NJS is running (as seen by the machine on which the Gateway is running) and `<NJS port>` is the port on which the NJS is listening for Gateway connections (this must be the same as the NJS configuration parameter "njs.gateway_port").
- Add the following line to the gateway.properties file:

  ```
  gw.named_njs=no_njs_registration_allowed
  ```

3.4 Configure SSL

- Load a key store containing the private key for the Gateway using the configuration parameter "gw.identity"
- Load a password to unlock the above key store by setting the configuration parameter "gw.password".

  Note that this option keeps the password permanently is a file. If you would prefer to delete the password after use, then write the password to a file named "password" in the Gateway configuration directory and *do not* set any value for "gw.password". The Gateway will read the contents of this file when it starts up and then delete the file.
- List the files containing the public certificates of Certificate Authorities (CAs) trusted to issue client certificates.

  Use the configuration parameter "gw.trusted_cas" to list the files (names are separated by ":")

3.5 Check scripts

- Check the supplied scripts in the bin directory. You may need to review the start up scripts (start_gateway and start_gateway_p) for the correct value of "java".

3.6 Set file permissions

The scripts signal the Gateway by writing files into the configuration directory. This means that any user who can write to the configuration directory can issue commands to the Gateway. Therefore the configuration directory should allow writes only for users allowed to administrator the Gateway.

The configuration values in the gateway.properties file determine which clients are accepted so it should also have restricted access.

4 Scripts

The Gateway distribution includes a number of scripts to start, stop and control the execution of the Gateway (Unix only)

The Gateway configuration directory is determined by the scripts using the following rules:

- Check the arguments to the script for the directory
- Use the value of the environment variable "UNICORE_GW"
- Use the local directory.

Note that the scripts assume the default file names as above. The best way to use files with different names is to supply a link e.g.

```
ln –s ajo_4.0.0-build1.jar ajo.jar
```

The scripts also assume that the names of the log files are the ones created by the Gateway.

The scripts check to see if an instance of the Gateway is running. The test for a running Gateway is based on the LAST_PID file in the configuration directory which, if it exists, contains the process id of the last started Gateway. If there is no process with this pid or there is no LAST_PID file, then the Gateway is assumed not to be running.

4.1 The Scripts

4.1.1 change_log_file

```
change_log_file configuration_directory
```

Gateway closes current log file and opens a new one (if running).

`configuration_directory` is optional.

4.1.2 change_log_level

```
change_log_level new_level configuration_directory
```

Change the Gateway's logging level (if running).

`new_level` is one of S, W, I, C, T, D (see elsewhere for descriptions of the logging levels)

`configuration_directory` is optional.

4.1.3 invalidate_crls

```
invalidate_crls configuration_directory
```

Invalidate all loaded CRLs and so force a reload from the issuing CAs.

`configuration_directory` is optional.

4.1.4 list_log_files

```
list_log_files type configuration_directory
```

Return the names of some of the log files.

If the type is "a", then the names of all the log files are returned.

If type is "l", then the names of the log files created since the Gateway was last started are returned ("L" returns the complement of this, the names of all the log files *except* those created since the last start).

Otherwise, type must be an integer, say n. If n is postive, then the names of the latest n files are returned e.g.

```
list_log_files 1
```

Returns the name of the current log file.

If n is negative the complement is returned e.g.

```
list_log_files –1
```

Returns the names of all except the current log file.

`configuration_directory` is optional.

4.1.5 start_gateway

```
start_gateway configuration_directory
```

Start the Gateway (if not running).

This will start the Gateway and execute it as a background process ("nohup"ed) with all output redirected to a logging file.

`configuration_directory` is optional.

4.1.6 start_gateway_p

```
start_gateway_p configuration_directory
```

Start the Gateway (if not running) prompting for a password.

This will start the Gateway and execute it as a background process ("nohup"ed) with all output redirected to a logging file.

configuration_directory is optional.

4.1.7 stop_gateway

```
stop_gateway configuration_directory
```

Stop the Gateway (if running).

`configuration_directory` is optional.

4.2 Example uses of the scripts

These examples assume that UNICORE_GW has been set to the Gateway's configuration directory.

View the current log file:

```
cat `list_log_files 1` | more
```

Purge the log directory, leaving the files since last started (note the capital "L"):

```
rm `list_log_files L`
```

Implement a site specific policy for log file cycling:

- Set the configuration parameter "gw.change_log_files" to some large value (e.g. 9999) to prevent the Gateway cycling log files at the wrong time.
- Create a cron job as follows:

```
/usr/unicore/my_gateway/bin/change_log_file \
/usr/unicore/my_gateway/conf

mv `/usr/unicore/my_gateway/bin/list_log_files 1 \
/usr/unicore/my_gateway/conf`  our_name
```

(This will of course invalidate the list_log_files script.)

Check for any possible problems since the last start (Severe or Warning messages):

```
cat `list_log_files 1` | grep " .\*"
```

Review the Gateway's processing of its configuration

```
cat `list_log_files 1` | grep " C: "
```

5 Logging

There are 6 levels of logging. In order of severity they are:

Severe: serious problem. Execution cannot or should not continue.

Warning: there is a problem or unexpected condition. Processing can continue - usually correctly, but occasionally may result in other problems (e.g. a client presented an invalid certificate.)

Information: useful information, not necessarily related to any problems (e.g. this level is used to log successful connections with name, source machine etc)

Configuration: messages relating to the Gateway configuration, reading of values, files and their interpretation.

Talk: verbose output,. May be useful to trace execution or gain hints about precursors to problems.

Debug: very verbose output.

If the logging level is set to a particular severity messages of that severity and higher will be logged. Messages of lower severity will not be logged.

The default logging level is Configuration (which will log C, I, W and S messages).

The Gateway generates structured names for the log files that will give the same order when sorted alphabetically and when sorted by creation time. This fact is exploited by the "list_log_files" script.

Lines of the log file are also structured. The first word as the time of creation of the message, the second the date, the third word notes the severity level of the message with important messages (severe or warning ending in a "*", the rest are terminated by a ":")

6 Configuration Parameters

The Gateway will read its configuration from the "gateway.properties" file in the configuration directory[1].

6.1 Syntax

A configuration parameter is of the form: name=value.

Spaces after the equals sign (except leading and trailing) are included in the value.

A value can extend beyond one line if the new line is escaped by \.

Blank lines are ignored.

Comment lines start with a # (inline comments are not allowed)

Incorrectly spelled parameter names will be ignored.

6.2 Parameters

gw.logging_level

Sets the level of logging.

Acceptable values are `S(evere), W(arning), I(nformation), C(onfiguration), T(alk) and D(ebug).`

The default is C.

This value can be changed at run time through the **change_log_level** script.

gw.gateway_host_name

optional (but recommended)

[1] This value can be changed by editing the `start_gateway` and `start_gateway_p` scripts but this may invalidate the assumptions of other scripts.

Default is the default IP address or machine name returned by system calls.

The name (or IP address) of the Gateway machine as seen by all possible clients.

gw.return_host_address

optional

Default is false (return the machine name)

If this is set the Gateway uses the IP address as determined by systems calls as the Vsites contact address.

gw.port

optional

Default is 4433

The port used by the Gateway to listen for client connections.

gw.connections

optional

No Default (no static NJSs)

The name of the file defining the Vsites supported by the Gateway (and possibly the client listeners used).

The name can include directory components which, if necessary, are relative to the Gateway configuration directory. Absolute paths are valid.

If set to “registered_vsites_only”, then no file is read.

gw.named_njs

optional

If present, then only NJSs who present certificates whose distinguished name contains one of the strings in this field are accepted to dynamically register with the Gateway.

Strings are separated by “:”

Examples:

To prevent any NJS regsistration:

```
gw.named_njs=no_njs_registration_allowed
```

To allow NJSs issued for the organisation “Fujitsu Laboratories of Europe”:

```
gw.named_njs=Fujitsu Laboratories of Europe
```

gw.max_threads

optional

Default is “25”

The maximum number of threads that can be active. If this number is exceeded, then connections will be refused until some threads become free.

gw.identity

optional

Default is “server_cert”

The name of the file containing the key store (private key plus certificate) used by the Gateway to accept SSL connections from clients.

The name can include directory components which, if necessary, are relative to the Gateway configuration directory. Absolute paths are valid.

gw.password

optional

Default Gateway looks for password in file names “password” in its configuration directory.

The password used to unlock the Gateway’s private key(s).

gw.trusted_cas

required

A “:” delimited list of file names. Each of these files must contain one or more public certificates of the trusted CAs. SSL connections accepted only if the client presents a certificate that has been issued by one of these trusted CAs.

gw.change_log_files

optional

Default is “24”

When the Gateway should close the current log file and open another.

If this is a string starting with “h”, then a new log file is opened every hour on the hour.

If this is a string starting with “d”, then a new log file is opened at the start of every day (midnight local time).

If the string is a positive number, say n, then a new log file is opened every n hours after the time that the Gateway was started up (e.g. If the Gateway is started at 3:09 pm with gw.change_log_files=24 a new log file will be opened every day at 3:09 pm, but if it was started with gw.change_log_files=daily a new log file would be opened every midnight).

Any other value is invalid.

gw.conn_timeout

optional

Default is 15 minutes

Connections from clients are closed when they are idle for longer than this time (in minutes).

gw.check_crls

optional

Default=false

Set to “true” if you want the Gateway to consult Certificate Revocation Lists (see below)

gw.crls_grace_ratio

optional

Default is –1.0 (none)

The extension that the Gateway allows to a CRL’s validity period when the update of the CRL fails. This is a fraction of the initial validity period.

gw.default_crl_validity

optional

Default is 24 (hours)

The validity period of a CRL if it does not contain one.

gw.file_watch_interval

optional

Default is 1000 (milliseconds)

The Gateway checks the configuration directory for a command file every file_watch_interval milliseconds.

6.3 Influencing the Java run-time

proxySet

optional

if set must be true

Default is false

Pass HTTP request through a proxy

proxyHost

optional (required if ProxySet is true)

Name of the machine running the proxy

proxyPort

optional (required if ProxySet is true)

port number for the proxy

6.4 Unused

The following configuration parameters are no longer used:

gw.log_is

gw.debug_threads

gw.debug_connections

gw.change_log_int

gw.log_directory

gw.flush_log

gw.vsites

7 Certificate Checking

The Gateway performs the following checks on certificates presented by a client requesting a connection[2].

- The certificate that has been issued by one of the trusted Certificate Authorities.
- The certificate has not been revoked by the issuing CA (see below)
- The certificate is within its validity period (the local time is later than the "not before" time of the certificate and the local time is before the "not after" time of the certificate).

8 Certificate Revocation Lists

The Gateway will check for revoked certificates every time that a new SSL connection is made by a client if the following two conditions are met:

- the initialisation value `gw.check_crls` is set to `true`
- the client's certificate contains a location for the Certificate Revocation List (CRL) in the X509 V3 extension "CRL Distribution Point" (OID 2.5.29.31)

8.1 CRL Update policy

The Gateway will update its copy of a CA's CRL as necessary.

[2] This list is probably incomplete as much of the checking is performed by default by the library used to handle SSL connections (The Sun Java Secure Socket Extension library).

The CA is assumed to require CRL checking by setting the "CRL Distribution Point" value in the certificates that it issues. The Gateway will only accept connections using certificates from such a CA if it has an up to date CRL from the CA. If the Gateway does not have an up to date CRL when a certificate is presented, then it will request a new CRL (if this request fails, then the new connection is rejected)[3].

All CRLs are out of date when the Gateway is started.

The Gateway marks a CRL as out of date when the next update time set by the CA has passed. If there is no next update time, then a CRL is valid for 24 hours[4].

The Gateway Administrator can make **all** CRLs out of date by using the `invalidate_crls` command.

CRLs are sometimes fetched during the certificate checking phase and so some new connections may experience a longer than usual start up time while the new CRL is accessed.

8.2 Limitations

8.2.1 Gateway only

The CRLs are consulted by the Gateway (and not by the NJS) and thus the checking is done only for the Consignor's[5] certificate. The Endorser's certificate is never checked for entry in a CRL by the Gateway.

As the Consignor and Endorser of a root AJO are the same the Endorser's certificate is effectively checked when the root AJO is consigned. However this checking is only done by the primary Usite and not by Usites executing sub AJOs.

It is also possible that a certificate may be entered into a CRL after the consign of a root AJO but before the execution of the sub AJO (or after an SSL connection is established to the Gateway which can be held open for many AJO consigns).

If this behaviour is unacceptable to a site, then it should enhance the UUDB code that the NJS uses to check Endorser certificates against CRLs during the `unicoreplus.UUDB.incarnate(User user)` method.

8.2.2 Limited parsing of CRL Distribution Points

The CRL checking code in the Gateway assumes that here is just one CRL Distribution Point and that this has a URI using the HTTP or HTTPS protocols.

---

[3] The initialisation value `gw.crls_grace_ratio` can be used to set a period for which the Gateway will reuse an old CRL if loads of a updated CRL from the CA fail. The "grace ratio" is a floating point number and is the fraction of the time that an old CRL was valid for which the old CRL will be used. For example, if the initial CRL was valid for 1 day and the grace ratio was set to 0.5 the Gateway will continue to use the initial CRL after its expiry for 12 hours (or until a load of the updated CRL succeeded). After 12 hours with no successful load all connections using certificates signed by the CA are rejected (until an updated CRL is successfully loaded). By default there is no grace period (ratio = 0). The grace ratio can be greater than 1.0.

There is no grace period if the administrator invalidates the CRLs or at Gateway startup.

[4] This can be adjusted using the initialisation value `gw.default_crl_validity` which gives the time (in hours)for which a CRL is valid if there is no information from the CA or if the information is inconsistent (e.g. the next update time has passed).

[5] The *Consignor* is the entity (identified by a X509 certificate) that sent the AJO to a Usite, this can be either the User who created the AJO or a NJS that is consigning a sub-AJO. The *Endorser* is the entity (always a User) that created the AJO. Thus the Consignor and Endorser are the same for root AJOs but differ for sub-AJOs. For more details see the UPL document on the BSCW server.

8.2.3 Assumes Web server certificates are issued by a trusted CA

If a CRL is fetched from a HTTPS URL the Gateway will trust the Web server only if its certificate has been issued by one of its trusted CAs.

9 Advanced Configuration.

This section describes how to:

- configure listeners for client connections so that one Gateway can present more than one certificate from the same instance
- configure the supported Vsites to use different connection methods

This file read is set by the configuration parameter “gw.connections”.

9.1 Passwords

As there are muliple certificates it may be necessary to supply more than one password to unlock their keystores. The methods described previously can be used to supply a single password for all keystores there are also two ways to supply individual passwords.

The Gateway code first checks the Listener definition line for a password (see below). The Gateway code then looks for the value from the gw.password configuration value (this will be a single global value). Lastly, the Gateway reads the password file (as described previously) but assumes an extended format when there is more than one word in the file. The extended format is pairs of words, the first being the Listener name, the second the password.

9.2 Syntax

- Comment lines start with #.
- Blank lines are ignored.
- Required fields are positional.
- The fields with an = are optional and not positional.
- There must be no space after any =.
- Tokens are delimited by spaces (none allowed inside).
- The <> are not part of a valid tokens.
- Definitions must be on a single line.

9.3 Specification of Listeners

The Gateway can present one certificate per port.

If the site would like to accept connections using more than one certificate (for example if it is part of Unicore and Eurogrid), then it is possible to do this by configuring the Gateway to listen on more than one port for connections from clients.

The format of a Listener definition line is (on one line):

```
listener=<name> port=<number> identity=<file_name>
password=<password> trusted=<file_name_list>
```

listener=

required

The name of the listener.

port=

optional

Defaults to the value of the configuration parameter gw.port.

The number of the port on which this listener will listen.

identity=

optional

Defaults to the value of the configuration parameter gw.identity.

This file contains a keystore (either PKCS12 or JKS) that contains either a single private key and certificate or, if there is more than one entry, one of the entries must have an alias which is the same as the listener's name.

The certificate is used to identify the Gateway to clients.

password=

optional

Defaults to the value of the configuration parameter gw.password.

This is a string used to unlock the identity keystore. This password is used to unlock the keystore and for each key pair within the keystore.

This option can be used if the password are different for different identity keystores.

trusted=

optional

Defaults to gw.trusted_cas

A list of files names, delimited by ":".

Each file contains one or more public certificates of Certificate Authorities that are trusted to issue Unicore user certificates.

9.4 Specification of Vsites

The format of a Vsite definition line is (on one line):

```
<Vsite name> <NJS machine> <NJS port> class=<class for
connecting to NJS> type=<type> ajo=<ajo spec>/<ajo
version>  SSL
```

or

```
vsite=<Vsite name> <NJS machine> <NJS port>
class=<class for connecting to NJS> type=<type>
ajo=<ajo spec>/<ajo version>  SSL
```

<Vsite name>

required

The name of the Vsite

<NJS machine>

required

The name of the machine running the NJS.

<NJS port>

required

The port on which the NJS is listening for Gateway connections (this must match an entry in the Vsite's NJS configuration file).

class=

optional

Defaults to com.fujitsu.arcon.gateway.VsiteConnectionFactoryImpl.

The Java class that will be used to transfer the client data to the target NJS. The API for this class is described below.

type=

optional

Defaults to NJS

The NJS type to be placed in the org.unicore.Vsite instance (default is NJS)

ajo=<ajo spec>/<ajo_version>

optional

Defaults to the specification and version from the AJO Jar file used by the Gateway.

The name of the AJO specification support by the Vsite(see org.unicore.Vsite).

SSL

Optional

Defaults to plain sockets.

If present the Gateway will contact the NJS using SSL.

9.5 API for classes to handle the UPL

The Gateway accepts connections from clients. It then reads the UPL header object from the input stream. If the UPL header object is an instance of ListVsites, then the Gateway will reply with the appropriate ListVsitesReply. Otherwise, the Gateway passes the connection on to a `VsiteConnection` instance that it gets from the `VsiteConnectionFactory` for the target Vsite. The `VsiteConnectionFactory` is an instance of the class specified on the "class=" field of a Vsite definition line (see above).

As the actual class for the `VsiteConnectionFactory` is dynamically loaded instances of it are created by calling a no argument constructor (which must therefore be public).

A specification of the interface (Javadoc) can be found in the “docs” directory.

9.5.1 Example

The Gateway contains a sample implementation of these interfaces (source code for this can be found in the “docs” directory). This implementation provides a path of communication between instances of NJS Alternative File Transfer mechanisms so that the source of a file transfer can get a destination directory and Xlogin.

An instance of an AFT connection handler is created by adding the following Vsite definition line to the “connection” file:

```
# This is a Vsite connection called "njs_aft" to an "AFT server" on "arcon" listening
# on port "1204".
# Connections will be managed by the
"com.fujitsu.arcon.gateway.AFTVsiteConnectionFactory"
# class.
# The Vsite has type "AFT"

vsite=njs_aft arcon 1204 class=com.fujitsu.arcon.gateway.AFTVsiteConnectionFactory
type=AFT
```