

## 5<sup>th</sup> DEISA Training Session

CINECA, Bologna, 30<sup>th</sup> October 2007

### DESHL Exercises

#### 1 Prerequisites

- The DESHL Client is a Java Application that requires Java Run Time Environment version 1.5 to be installed. The DESHL client has been tested with the SUN Java Runtime Environment.

Check which version of Java is installed on your computer

On MS Windows type `java -version` in a command window

On UNIX/Linux type `java -version` in a console window

If the execution of the command fails or if you have a java version older than 1.5, then go to the SUN web page to download and install the latest release on your local workstation.

- To use the DESHL client, you will need a valid keystore in .p12 format containing your DEISA personal certificate, and the root certification authority certificate (in .pem format) for the organisation which issued your personal certificate. These must be present in the same directory. (For more information on certificates please see Appendix I, at the end of this document.)
- You will also need to download the CA certificate for each of the DEISA gateways. These are available at <http://winnetou.matrix.sara.nl/deisa/certs/unicerts.tar.gz>, these certificates must be unpacked into the directory containing your keystore and its root certificate.

#### 2 Installation

The following steps are required to install and configure the DESHL 4.1 client:

1. Download the installer
2. Install and configure the DESHL
3. Verify the settings

## **2.1 Download the installer**

Download the DESHL 4.1-RC2 installer from the JRA7 project site:  
<http://forge.nesc.ac.uk/download.php/219/DESHL-4.1-RC2-installer.jar>

Either save it, or execute it directly if your browser supports execution of jar files.

## **2.2 Install and configure the DESHL via the installer GUI**

### **Run the GUI installer**

If the installer did not immediately run in the previous step then run it now:

In a file explorer window double click on the installer jar file to start the installation.  
Alternatively, to install from the command line on UNIX:

```
$ java -jar DESHL-4.1-RC2-installer.jar
```

Alternatively, to install from the command line on Windows open a DOS command window and run the following:

```
C:\>java -jar DESHL-4.1-RC2-installer.jar
```

If problems are experienced when running the installer, extra debug information can be obtained by running with these additional arguments:

```
$ java -DTRACE=true -DSTACKTRACE=true -jar  
    DESHL-4.1-RC2-installer.jar
```

## Configure the DESHL using the GUI installer

The graphical installer opens up with a welcome screen for DEISA JRA7 followed by a license agreement. You will need to accept the license agreement before being allowed to install DESHL.

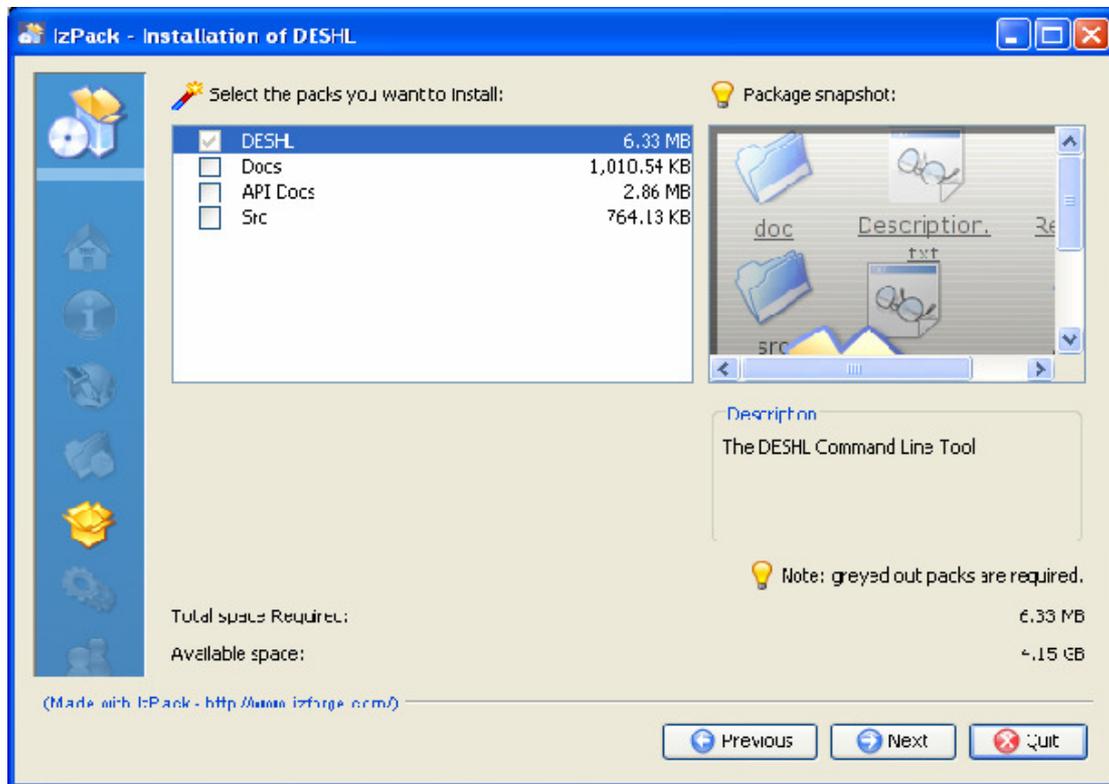
When prompted for an installation path, enter the following if installing on Windows:

```
C:\Program Files\DESHL-4.1-RC2
```

If installing on UNIX, specify a DESHL-4.1-RC2 directory in your local workstation home directory, for example:

```
/home/malcolm/DESHL-4.1-RC2
```

Next choose the packages to install as shown in Figure 1

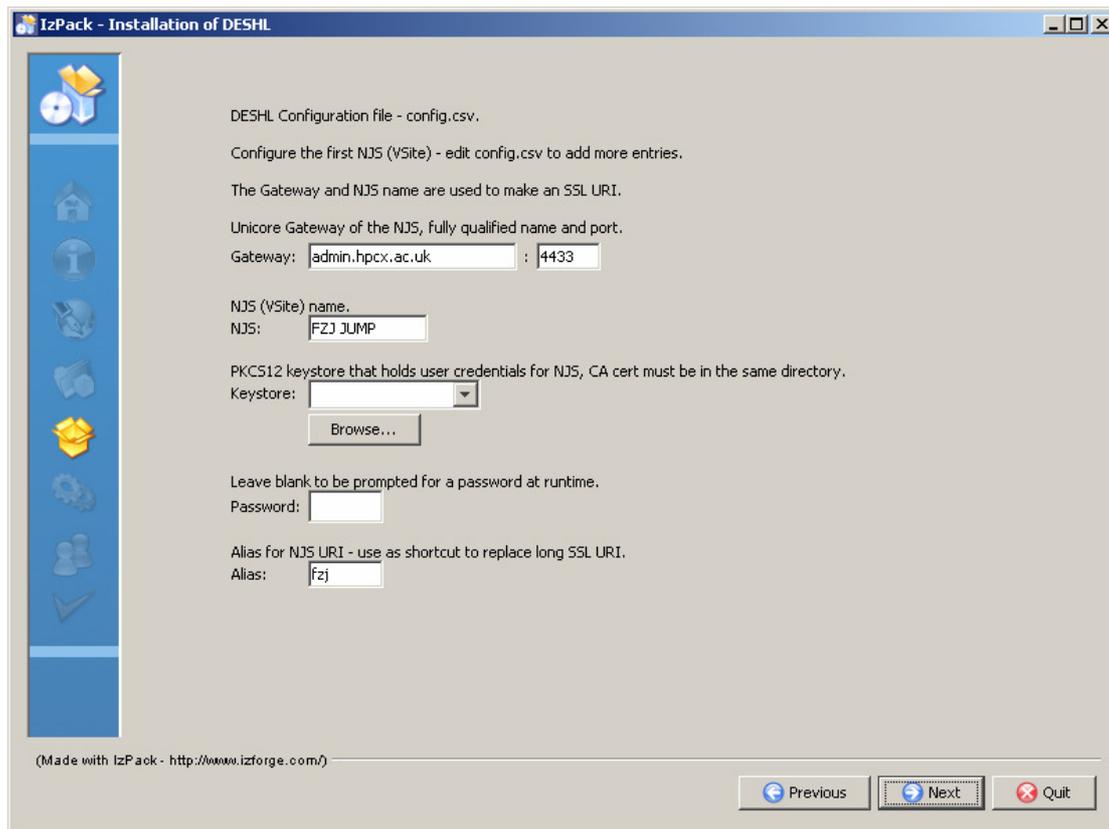


**Figure 1: DESHL Package selection.**

The base DESHL package is always required and cannot be deselected. Additional material can be selected or deselected for installation. The packages are:

- Docs – manuals for DESHL including the user manual and design document.
- API Docs – the Javadocs for the DESHL code.
- Src – the Java source for DESHL.

The next window, see Figure 2, shows how to configure one UNICORE NJS that can be used to interact with the DESHL infrastructure.



**Figure 2: UNICORE NJS details.**

The following information is expected (obtainable from the appropriate local site administration):

- The UNICORE NJS URI, this consists of:
  - The address and port of the UNICORE Gateway for that NJS
  - The name of the NJS (note that "%20" must be used instead of any spaces in the name)
- Use the 'Browse' button to navigate to the PKCS12 file that holds your user credentials. The directory in which this file resides must also contain the CA certificate for the issuer of your credentials, in '.pem' format.
- Password: For the purposes of the hands-on session, please leave this blank. (The password for the keystore can optionally be stored in the configuration file, but this is insecure and not recommended.)
- An alias for the NJS URI. Define a shortcut for the NJS URI when using data staging and job management commands, this will make using the DESHL much easier by not having to type long path names. For example, "idris", "fzj", etc. The alias must not contain any spaces.

It should be possible to access any DEISA site from any DEISA gateway; however it is strongly recommended that you configure the DESHL client to always use your home site's gateway. A list of the DEISA gateways available for the DESHL exercises is shown in the table below:

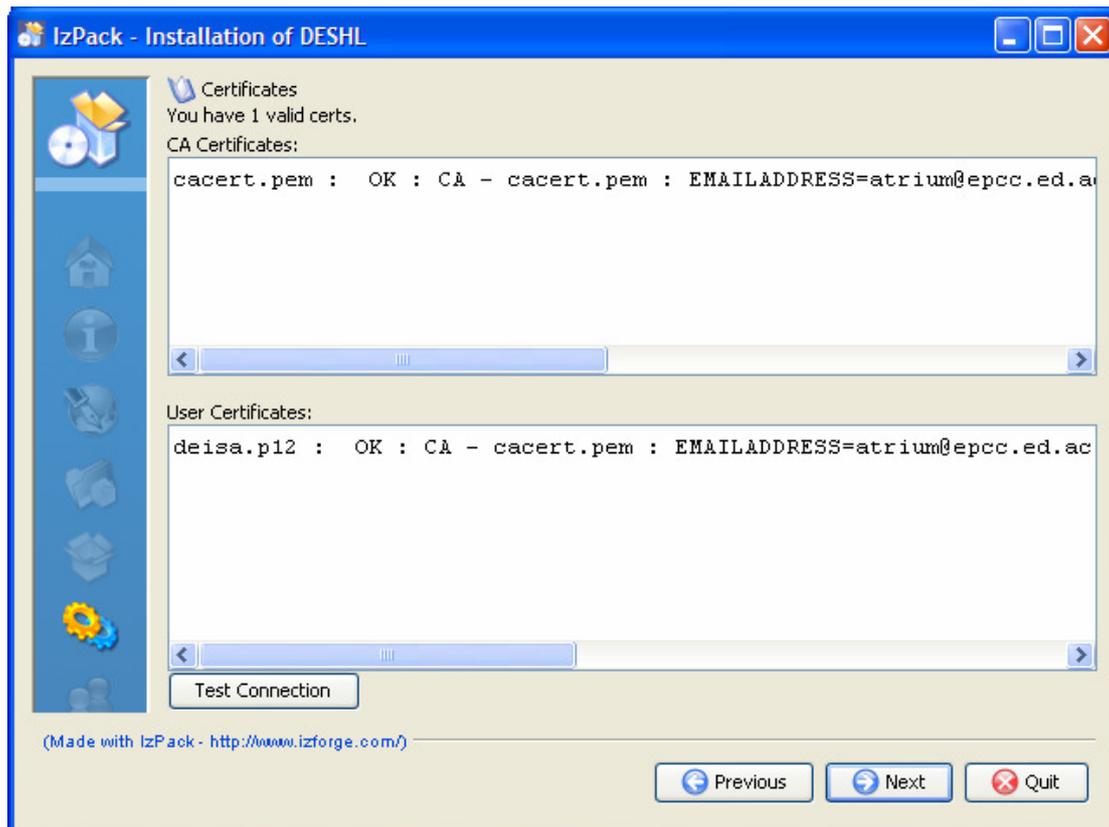
<b>Sitename</b>	<b>Gateway Address</b>	<b>NJS name</b>	<b>Port</b>
FZJ	zam177.zam.kfa-juelich.de	FZJ%20JUMP	4000
CINECA	fe04-ext.fec.cineca.it	CINECA-SP5	4433
RZG	unigate.rzg.mpg.de	RZG%20SP4	4433
SARA	uni-gw1.sara.nl	SARA%20ASTER	4004
IDRIS	norma2.idris.fr	IDRIS%20ZAHIR	4433
CSC	uni.csc.fi	CSC	4000
BSC	opsuni01.bsc.es	BSC%20Marenostrum	4433
LRZ	unicore.lrz.muenchen.de	LRZ%20HLRB%20II	4433
EPCC	admin.hpcx.ac.uk	EPCC%20HPCx	4433

*(The BSC gateway will normally be accessible to those users who have BSC as their home site, but the supercomputer MareNostrum will not be available for the exercises nor during the DEISA Test User validity period. Please also note that IDRIS is unavailable for non-EU attendees.)*

When the “Next” button is clicked, a dialog is displayed to enter the keystore password to allow your certificate to be validated. (This password is NOT stored in the configuration file.)



Once the password has been entered, the installer checks for a root CA certificate for the authority which issued your personal certificate. The list of certificates found is displayed in the next screen:



If your certificate cannot be validated, e.g. due to an incorrect password or a root certificate not being found, then the “Next” button is deactivated and it will not be possible to continue with the installation. (Please ask for assistance if this happens.)

Using this information, the installer installs all selected materials into the chosen directory. It also sets up the DESHL configuration file ‘config.csv’ with the details given for the NJS URI. (This configuration file can be manually edited after installation to change the settings, but this should not be necessary for the exercises.)

Once complete all the necessary files are installed and configured to allow the DESHL to be used with the chosen NJS.

The installer does not add the DESHL executable to the path – add this manually as follows:

In UNIX, from the DESHL install directory:

```
$ export PATH=$PWD/bin:$PATH
```

In Windows, if DESHL install directory is C:\deshl:

```
C:\deshl>set PATH=C:\deshl\bin;%path%
```

### 2.3 *Verify the settings*

A simple test to verify that the DESHL client is correctly installed and configured is to try to list the contents of the home storage at the configured site. For example:

```
$ desh1 list ssl://norma2.idris.fr:4433/IDRIS%20ZAHIR/deisa_home/
```

Or if using shortcuts:

```
$ desh1 list idris/deisa_home
```

For the site, the DESHL client should display a list of the files and directories in the top level of the HOME storage. (The HOME storage is the site-specific home directory for the user defined in the UNICORE incarnation database.)

If this is the first time that you have run a DESHL command, note that there will be a short delay while a new local server is started, then you will be asked to enter the password for your keystore. (Once the server is running, it should not be necessary to re-enter your password during the exercises.)

```
Starting new server ...  
Please enter password for keystore /home/malcolm/mykeystore.p12:
```

## 2.4 Help System

The DESHL client contains a help option, `-h`. This can be used to display the set of commands which are currently supported, and the options for a specific command.

```
$ deshl -h

[-h] command [command-options]
Where:
  -h,--help    Print this message.

With commands:
  fetch
  copy
  isDir
  move
  status
  jobs
  remove
  makeDir
  list
  submit
  sites
  terminate
  isFile
  exists

Run deshl <command> -h for individual command options.
```

## 3 Using the DESHL client

The following exercises assume that a shortcut to the configured site was specified during installation. For purposes of illustration, this will be assumed to be “`idris`”, however you should substitute your own shortcut in the examples.

Script files which are copied to a DEISA site and then executed must be saved as ANSI Unix-compatible format before copying to the DEISA site. Note that neither Notepad nor Wordpad are suitable as these create files with carriage return and linefeed line delimiters that will not run correctly.

One possible editor is `emacs` available here:

<http://www.gnu.org/software/emacs/windows/faq2.html#where-precompiled>

Files are created as `dos` files but can be converted to `unix` format:

Press `ESC-X` then type  
`set-buffer-file-coding-system <RET>`  
then type  
`unix <RET>`

### 3.1 Upload, submit and manage a simple job

Create a new directory on your local workstation and call it `exercises`

Go into this directory, and verify that the `deshl` executable has been correctly added to your path by running `deshl -h` from the command prompt.

In your `exercises` directory, create files called `hello-submit.sh` and `hello.sh`, and add the contents to the respective files as shown below:

#### **`hello-submit.sh`**

```
#!/bin/bash
# Test job script for DESHL using SAGA.
#
# SAGA JobDefinition based directives:
#$ SAGA_FileTransfer = file:///jobs/hello.sh#DEISA_HOME > hello.sh
#$ SAGA_HostList = idris
#$ SAGA_JobCmd = hello.sh
```

#### **`hello.sh`**

```
#!/bin/bash

echo "Here I am on `hostname` in $PWD. Current date is `date`"
```

Please note that you must edit `hello-submit.sh` to specify the execution site:

```
#$ SAGA_HostList = idris
```

The `hello-submit.sh` file contains a set of SAGA directives for submitting and managing the execution of a job, the job in this case being the executable script, `hello.sh`.

The rest of this exercise will show you how to import the executable script to a DEISA site, and then use the SAGA script with the DESHL command line tool to submit the job and retrieve the output from the job.

The following steps will be performed:

1. Verify if a remote directory exists
2. Create a remote directory for the executable script
3. Upload the executable script to the
4. Edit the SAGA script
5. Submit the job
6. Get the job status
7. Cleanup the job and retrieve the job output

#### **1. Verify if a remote directory exists**

First we want to determine if the remote directory exists:

```
$ desh1 exists idris/deisa_home/jobs
```

If the remote directory exists, the following message should be displayed:

```
exists: target idris/deisa_home/jobs exists
```

Otherwise, the following message is displayed:

```
exists: target idris/deisa_home/jobs does not exist
```

## 2. Create a remote directory for the script

If the remote directory does not exist, create it using the `deshl makeDir` command:

```
$ deshl makeDir idris/deisa_home/jobs
```

Verify that the remote directory now exists by listing the contents of its parent directory. This should show the `jobs` directory in the listing.

```
$ deshl list idris/deisa_home
...
drwx    32768    Feb 07 2007 01:22:01    jobs
...
```

## 3. Upload the executable script to the server

You now must upload the executable script, `hello.sh`, from your local machine to the DEISA site where the job will be run. To do this, use the `deshl copy` command:

```
$ deshl copy hello.sh idris/deisa_home/jobs/hello.sh
```

If a file of the same name already exists in the target location, the following error will be displayed:

```
Error running command: copy: Copy failed.
Likely cause: Remote file exists, specify overwrite flag to
overwrite.
(Check logs for further details.)
```

If this happens, then repeat the copy with `-f` to overwrite the existing file:

```
$ deshl copy -f hello.sh idris/deisa_home/jobs/hello.sh
```

## 5. Submit the job

Now that your script on the remote site has execute permissions, you can submit a job to run it.

```
$ deshl submit hello-submit.sh
```

This returns a job identifier, which we can use to monitor the job's status

## 6. Get the job status

```
$ deshl status idris%2F957383131
Job: idris%2F957383131, has status: Running
```

You can request more details of the status using `-f`

```
$ deshl status -f idris%2F957383131

Job: idris%2F957383131, has status: Done
  Site: ssl:// norma2.idris.fr:4433/IDRIS%20ZAHIR
  Name: hello.sh
  Running (Unicore:RUNNING): 10/10/2007 18:24:05
  Done    (Unicore:FINISHED):10/10/2007 18:26:19
```

Once the job has successfully completed, the status will be reported as `Done`.

## 7. Cleanup the job and retrieve the job output

In your `exercises` directory, create a subdirectory called `output`.

Once the job has completed, the output from the job can now be retrieved and any file resources consumed by the job released using the `fetch` command. The `fetch` command allows you to retrieve the job output into a specified directory; for this example you will retrieve into the local directory `output` that you have just created.

```
$ deshl fetch -d output idris%2F957383131
```

This will place two files in the specified directory; one of these is the job's output to standard output, and the other is the job's output to standard error. Look at the files in the `output` directory to verify this.

### 3.2 *Staging in data for a job, staging out results*

In this part of the tutorial, you will stage in data required by a job, run the job and then retrieve data produced by the job. The example script in this case will perform a simple concatenate on two specified files and the file containing the joined files will be the final result.

In your `exercises` directory, create files called `catfiles_submit.sh` and `concat.sh`; add the contents as listed below.

```
catfiles-submit.sh

#!/bin/bash
# Test job script for DESHL using SAGA.
#
# SAGA JobDefinition based directives:
#$ SAGA_FileTransfer = file:///jobs/concat.sh#DEISA_HOME > concat.sh
#$ SAGA_FileTransfer = file:///tutorial/filea#DEISA_HOME > filea
#$ SAGA_FileTransfer = file:///tutorial/fileb#DEISA_HOME > fileb
#$ SAGA_FileTransfer = file:///tutorial/filec#DEISA_HOME < filec
#$ SAGA_HostList = idris
#$ SAGA_JobCmd = concat.sh
```

```
concat.sh
#!/bin/bash

cat filea fileb > filec
```

Again, please note that you will have to change the value of `SAGA_HostList` to specify the site that you are submitting the job to.

Use the `deshl copy` command to import the executable script file, `concat.sh`, to the jobs directory on the remote site.

```
$ desh1 copy concat.sh idris/deisa_home/jobs/concat.sh
$ desh1 list idris/deisa_home/jobs/concat.sh
```

Use the `deshl makeDir` and `deshl copy` commands to create a tutorial directory on the remote host, then copy two files from your local machine into the remote tutorial directory as `filea` and `fileb`. Confirm that the files have been copied using the `deshl list` command.

For example:

```
$ desh1 makeDir idris/deisa_home/tutorial
$ desh1 copy localFileA.dat idris/deisa_home/tutorial/filea
$ desh1 copy localFileB.dat idris/deisa_home/tutorial/fileb
$ desh1 list idris/deisa_home/tutorial
```

You can now submit the job as before:

```
$ desh1 submit catfiles-submit.sh
```

You can monitor the job status using the `deshl status` command. When the job has completed, an output data file called `filec` will have been produced in the remote tutorial directory. Confirm that this file exists by listing its properties:

```
$ desh1 list idris/deisa_home/tutorial/filec
-rw-   63      Mar 01 2007 03:29:57   filec
```

Export this file to your local machine using the `deshl copy` command:

```
$ desh1 copy idris/deisa_home/tutorial/filec filec.dat
```

Inspect `filec.dat` to confirm it contains the concatenated contents of the two original files. Remember to cleanup the job using the `deshl fetch` command.

### 3.3 *Listing available modulefiles at a DEISA site*

In this part of the tutorial you will submit a job to list the available modulefiles at a DEISA site. The job will output information about the environment at the DEISA site. You will then retrieve the output from the job.

In your `exercises` directory, create files called `listmodules_submit.sh` and `listmodules.sh`; add the contents as listed below.

***listmodules.sh***

```
#!/bin/bash
module load deisa
module list
module avail
```

***listmodules-submit.sh***

```
#!/bin/bash
# Test job script for DESHL using SAGA.
#
# SAGA JobDefinition based directives:
#$ SAGA_FileTransfer = file:///jobs/listmodules.sh#DESIA_HOME > listmodules.sh
#$ SAGA_HostList = idris
#$ SAGA_JobCmd = listmodules.sh
```

Again, please note that you will have to change the value of `SAGA_HostList` to specify the site that you are submitting the job to.

Use the `deshl copy` command to import the executable script file, `listmodules.sh`, to the `jobs` directory on the remote site.

```
$ desh1 copy listmodules.sh idris/deisa_home/jobs/listmodules.sh
$ desh1 list idris/deisa_home/jobs/listmodules.sh
```

You can now submit the job:

```
$ desh1 submit listmodules-submit.sh
```

You can monitor the job status using the `deshl status` command. When the job has completed, retrieve the job output using the `fetch` command as before.

```
$ desh1 fetch -d output idris%2F957384927
```

This will create two new files in the output directory, one containing the job's output to stdout and the other containing the job's output to stderr. Open the stderr file (`.err`) using a text editor. The contents of the file should be similar to the following:

```

This script was created and executed by Unicore
UNICORE - start of user output on stderr

Currently Loaded Modulefiles:
 1) mode/64                5) endian/big            9) compilerwrappers/yes
 2) fortraninteger/4      6) c++/8.0              10) deisa
 3) fortranreal/4        7) c/8.0
 4) fortrandouble/8      8) fortran/10.1
----- /usr/local/pub/Modules/modulefiles/init -----
deisa
----- /usr/local/pub/Modules/modulefiles/environment -----
compilerwrappers/no          fortraninteger/4(default)
compilerwrappers/yes(default) fortraninteger/8
endian/big(default)         fortranreal/4(default)
endian/little               fortranreal/8
fortrandouble/16           mode/32
fortrandouble/8(default)   mode/64(default)
----- /usr/local/pub/Modules/modulefiles/compilers -----
c/7.0                       c++/8.0(default)       java/1.4(default)
c/8.0(default)             fortran/9.1
c++/7.0                    fortran/10.1(default)
----- /usr/local/pub/Modules/modulefiles/libraries -----
blacs/3(default)           fftw/3                  pssl/3(default)
blacssmp/3(default)       hdf5/1.6(default)      psslsmp/3(default)
blas/4(default)           lapack/3.0(default)    pwsmp/6
blacssmp/4(default)       mass/4(default)        pwsmp/7(default)
essl/4(default)           nag/20                 scalapack/1.7(default)
esslsmp/4(default)        nag/21(default)        wsmpp/6
fftw/2.1.5(default)       netcdf/3(default)      wsmpp/7(default)
----- /usr/local/pub/Modules/modulefiles/tools -----
emacs/21(default)         omniORB/4.0(default)   tcl/8.4(default)
gmake/3(default)          openssl/3(default)     tk/8.4(default)
hpm/2.5(default)          perl/5.8(default)      totalview/7
nedit/5(default)          python/2.4(default)    totalview/8(default)
----- /usr/local/pub/Modules/modulefiles/applications -----
cpmd/3.9                  gopenmol/2.32(default) lammps/01Oct06(default)
cpmd/3.11(default)        torb/1.22(default)     lammps/12Feb07
cpmd2cube/jan05          torbwsmpp/1.22(default) namd/2.6(default)
cpmd2cube/apr06(default) wien2k/05(default)

UNICORE - end of user output on stderr
UNICORE EXIT STATUS 0 +

```

Finally, shutdown the local DESHL server by running a “quit” command:

```
$ deshl quit
```

## Appendix I – Certificate Management

A certificate compliant to the EU Grid PMA minimal requirements is required to use the DESHL. Details of how to obtain this certificate vary from country to country, therefore please contact the appropriate certificate issuing authority in your region for details on how to apply for a certificate. For each site that the user wishes to access, the DESHL client requires both the user's certificate and the issuing authority's root certificate to be made available in a directory which is accessible to the DESHL client at runtime (the directory location is set in the DESHL configuration). Both certificates must be X.509 (v3) compliant (DER encoded as base64 with additional header and footer lines). The user's certificate must also be packaged in a PKCS12 bundle with the same password as for the certificate (this is the same as for UNICORE).

Please also see the DEISA Primer:

[https://www.deisa.org/userscorner/primer/access\\_to\\_grid.php#2.1.3](https://www.deisa.org/userscorner/primer/access_to_grid.php#2.1.3)

The public certificate of the CA (Certificate Authority) that signed a user certificate **MUST** be placed in the same directory as the PKCS12 file holding that user certificate and it **MUST** be a '.pem' file.

If you have an X.509 certificate, with the public key in `usercert.pem` and the private key in `userkey.pem`, you can convert it to the PKCS12 format using the `openssl` (<http://www.openssl.org>) tool:

```
$ openssl pkcs12 -export -in usercert.pem -inkey userkey.pem -out  
user.p12 -certfile CApublicKey -name "user1"
```

First, you will have to type the passphrase that you chose for your original PEM certificate when you requested it to the RA, then a new passphrase for the PKCS12 keystore. OpenSSL is available also for the Windows operating system, visit:

<http://www.openssl.org/related/binaries.html>.

You can also run `openssl` under `cygwin` on Windows, see <http://www.cygwin.com/>.

It is also possible to export a user certificate and private key pair from a web browser in PKCS12 format.

For example in Internet Explorer from '*Tools/Internet Options...*' select the '*Content*' tab then '*Certificates*' button, choose the personal certificate to export and follow the export wizard selecting the PKCS12 format.

In Firefox choose '*Tools/Options...*' select '*Advanced*' then the '*Security*' tab, press the '*View Certificates*' button, and choose which of '*Your Certificates*' to backup then save as a PKCS12 file.