

DEISA Training Event

FZJ, 13th January 2009

DESHL Exercises

1 Prerequisites

- The DESHL Client is a Java Application that requires Sun Java Run Time Environment version 1.5 (or later) to be installed. (Please note that the DESHL client does not currently work correctly with IBM Java due to a mismatch with security libraries.)

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, please 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 the public CA certificate for the DEISA gateway that you will use to access the DEISA infrastructure. Normally this is also the root certificate for the authority which issued your personal certificate. However, should you want to use the gateway of a site in another country, you will need to download the set of DEISA public gateway certificates. These are available at <http://winnetou.matrix.sara.nl/deisa/certs/unicerts.tar.gz>. These certificates must be unpacked into the same directory as the directory containing your personal keystore and its root certificate.

Note: if you have already installed the UNICORE5 graphical client in a previous exercise, you can skip this step. Instead, copy your .p12 keystore into the same directory as where you unpacked the root certificates during the installation for the UNICORE5 graphical client. When prompted for a certificate during DESHL client installation, select your .p12 keystore in its new location in this directory.

2 Installation

The following steps are required:

1. **Download the installer**
2. **Install and configure the DESHL**

2.1 *Download the installer*

Download the DESHL 4.3 installer from the JRA7 project site:

<https://forge.nesc.ac.uk/download.php/297/DESHL-4.3-installer.jar>

Save it to a location on your workstation.

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

Run the GUI installer

To install from the command line on UNIX:

```
$ java -jar  
-Dgateways.xml=http://winnetou.sara.nl/deisa/hosts/gateways.xml  
DESHL-4.3-installer.jar
```

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

```
C:\>java -jar  
-Dgateways.xml=http://winnetou.sara.nl/deisa/hosts/gateways.xml  
DESHL-4.3-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-4.3
```

If installing on UNIX, specify a DESHL-4.3 directory in your local workstation home directory, for example:

```
/home/malcolm/DESHL-4.3
```

Next choose the packages to install as shown in Figure 1

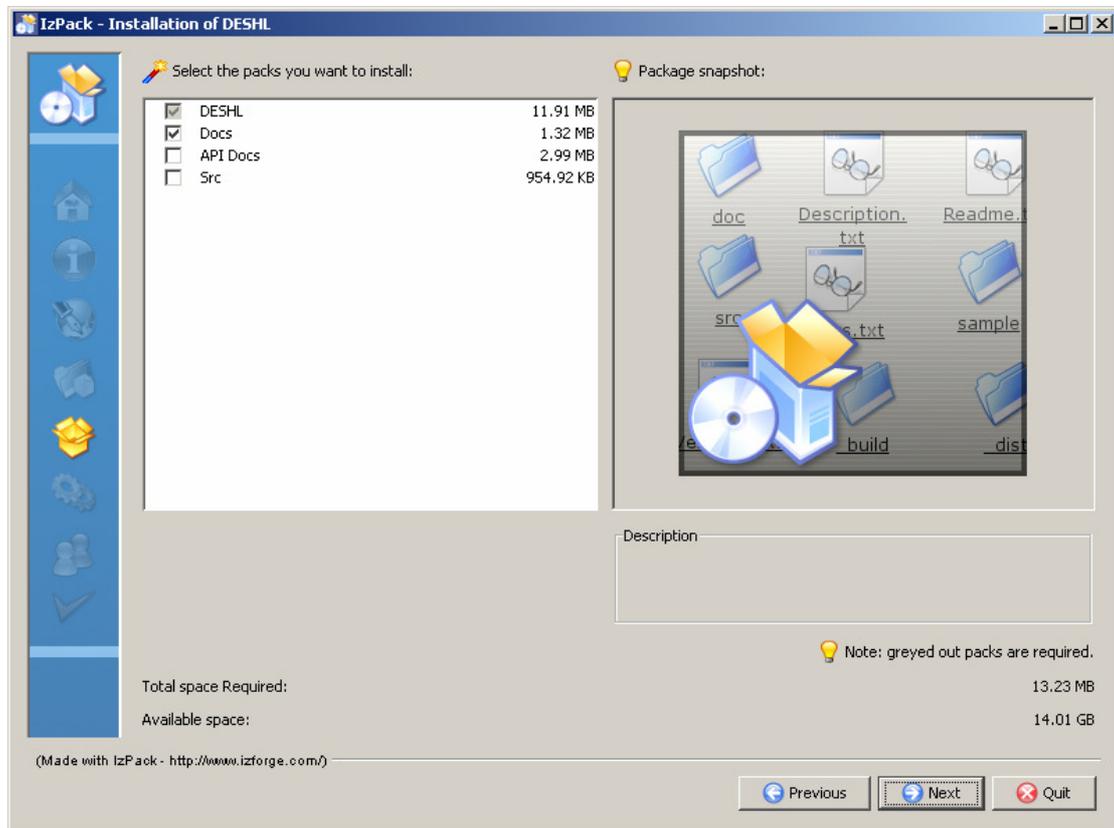


Figure 1: DESHL Package selection.

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

- Docs – user documentation including the DESHL User Manual.
- API Docs – the Javadocs for the DESHL code.
- Src – the Java source for DESHL.

The next window, see Figure 2, shows how to specify the keystore containing user credentials which will be used to access the DEISA infrastructure.

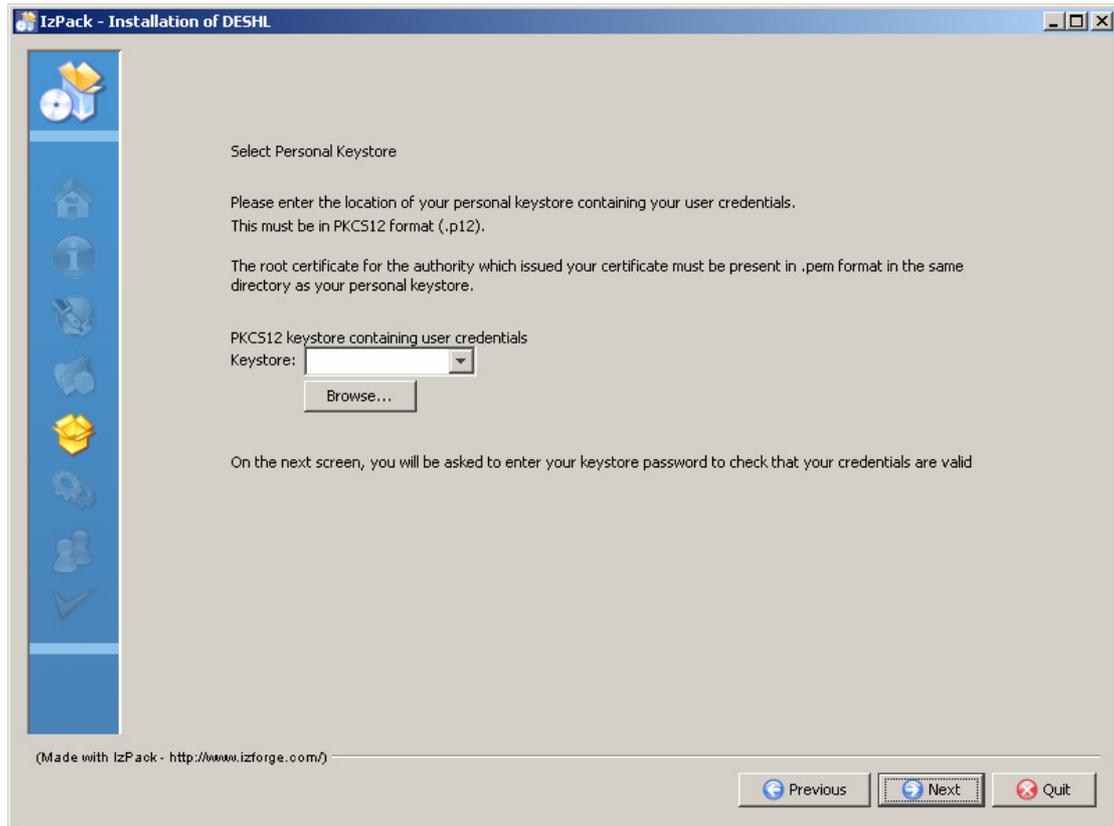


Figure 2: User keystore selection

Click on the “Browse” button to bring up a file browser. Use the file browser to select the PKCS12 (.p12) keystore containing the credentials that you will use to access the DEISA infrastructure. Having selected your keystore, click on the “Next” button to proceed with installation.

(Note that the root public certificate for the authority which issued your certificate must also be in the same directory as the directory where your keystore is located.)

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.)



Figure 3: Password entry for validation

Once the password has been entered, the installer searches the directory containing your keystore for a root CA certificate for the authority which issued your personal certificate. The list of certificates found is displayed in the next screen (Figure 4).

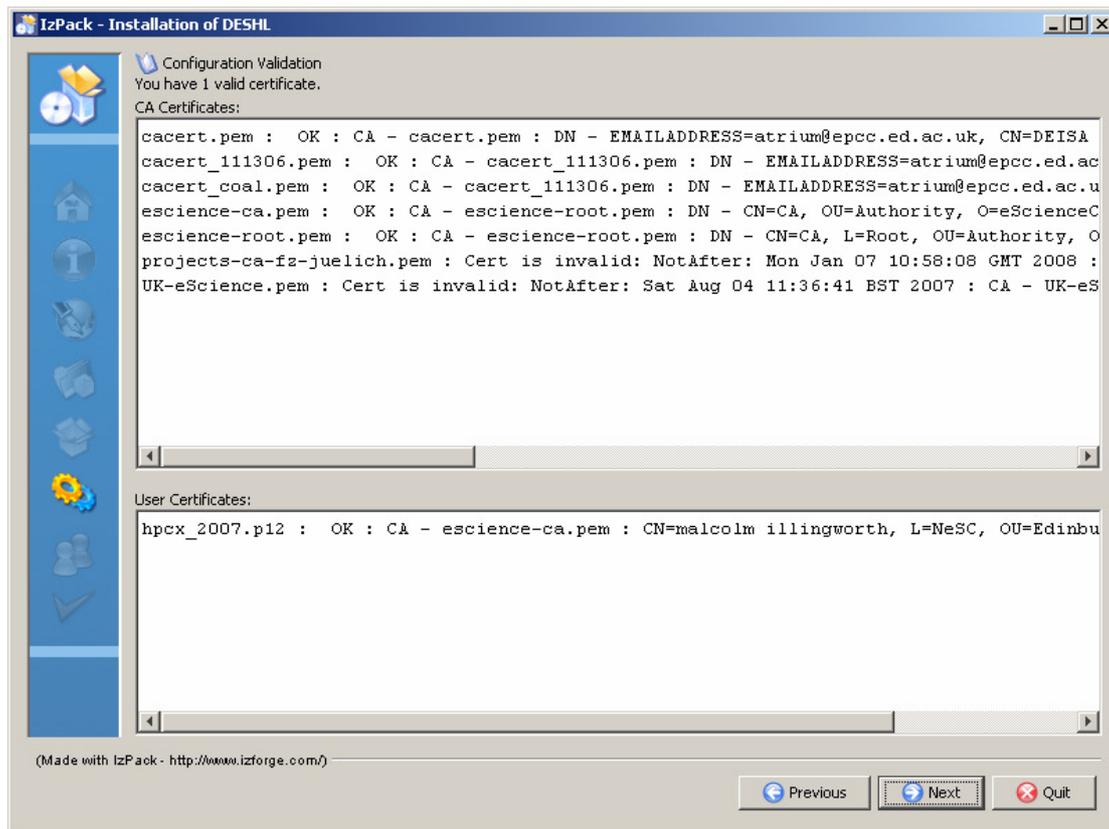


Figure 4: Successful validation

The upper display area shows a list of all root certificates found in the directory where your keystore is located. One of these root certificates must be the root certificate for the authority which issued your personal certificate, or validation will not be possible.

The lower display area shows a list of valid user certificates found. Normally this should just show the PKCS12 keystore which you specified at the start of installation.

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. (Appendix II shows examples of failures to validate a certificate.)

On receiving confirmation that your certificate is valid, click on the “Next” button to continue with the installation. In the next screen you will select a “home” site through which you will interact with the DEISA infrastructure.

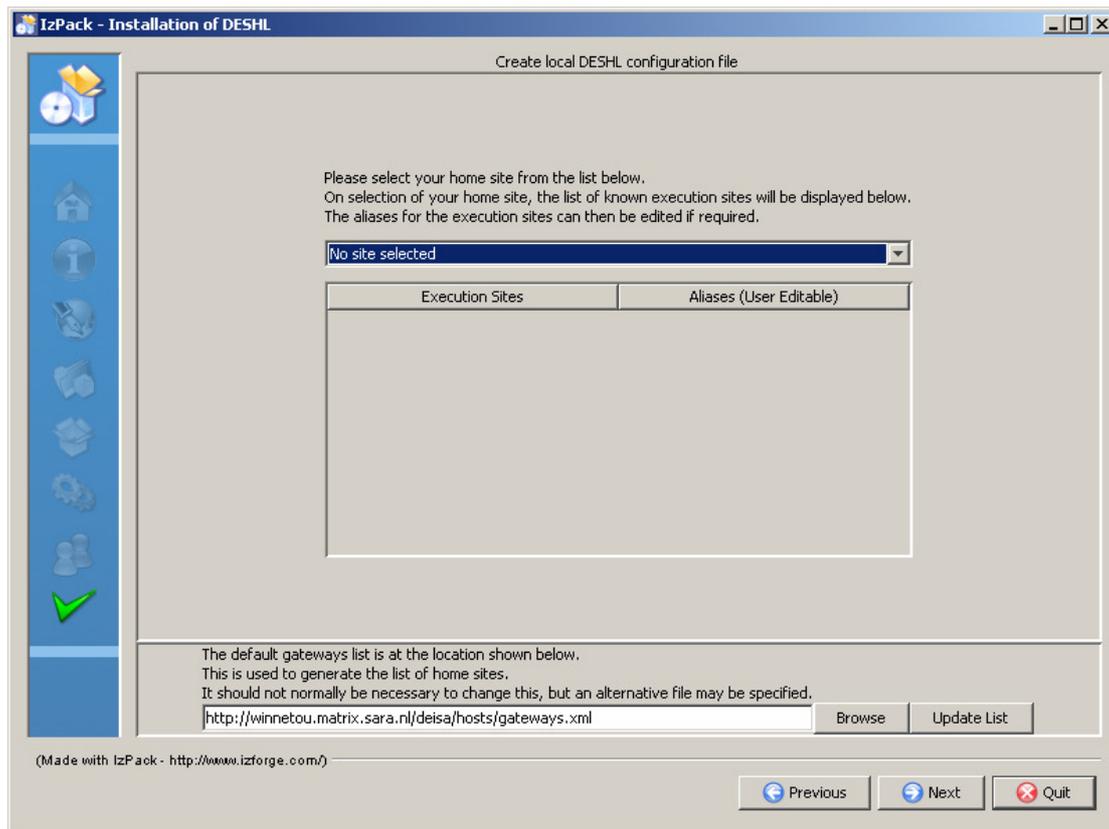


Figure 5: Home site selection and configuration screen

Use the drop-down list to select your home site from the list of home sites.

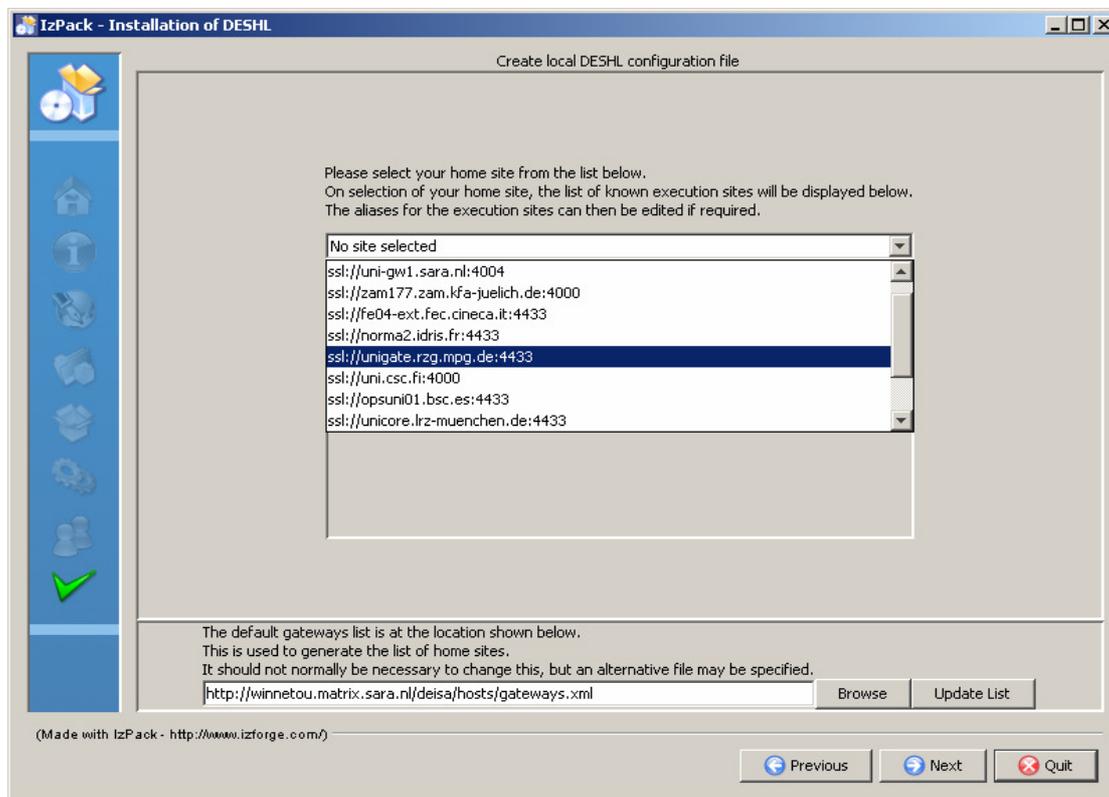


Figure 6: Home site selection

On selection of a home site, the installer will try to contact the site and return a list of the DEISA execution resources available at that site. For example:

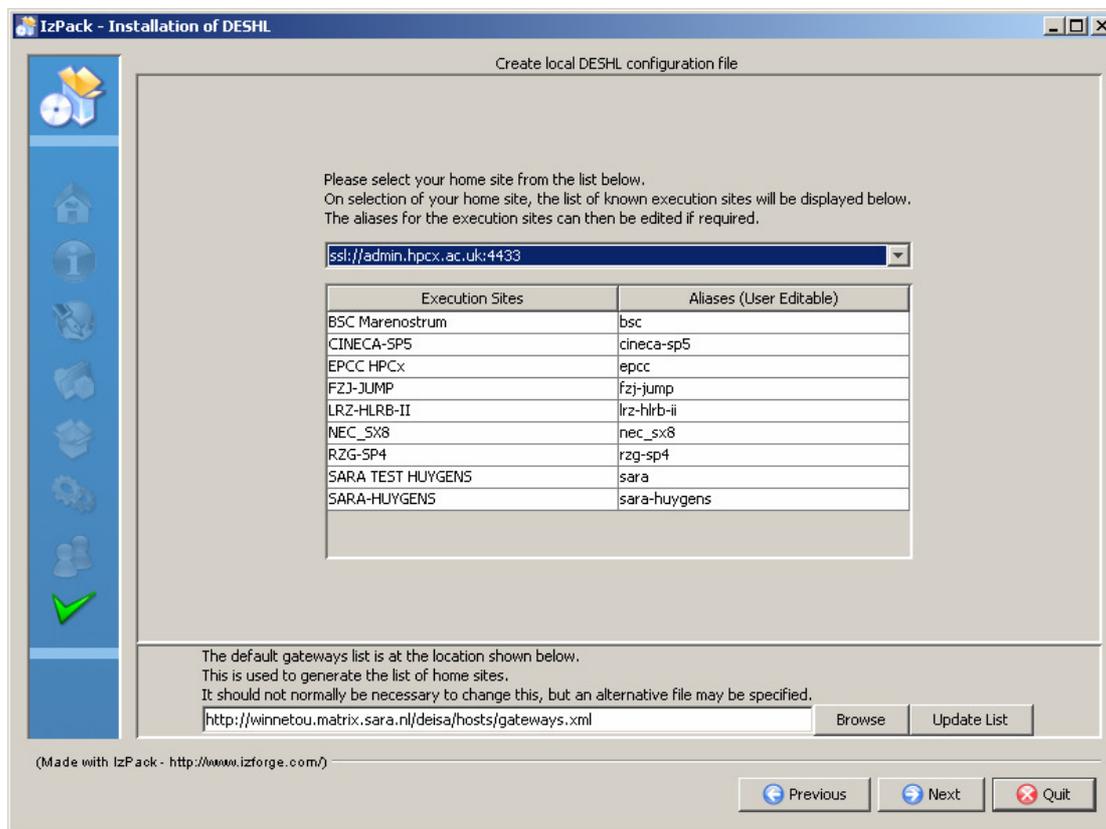


Figure 7: DESHL resources and aliases

On successfully contacting the selected site, the installer displays a list of the site names and a matching set of generated aliases. The aliases can be used to reduce the amount of typing required when using the deshl commands. These can be user edited if required. Click on the “Next” button to complete the installation.

Note that to access via a chosen home site; you must have the root public certificate for that site’s gateway in the same directory as the keystore which you selected at the start of installation. Normally this is the same certificate as the root public certificate for the authority which issued your personal certificate. Please note that if you try to access via a home site for which you do not have a certificate (or the gateway is unavailable), you will see an error similar to the one shown below:



Figure 8: Error contacting home site

When the installer has successfully completed the installation, the following screen is displayed:

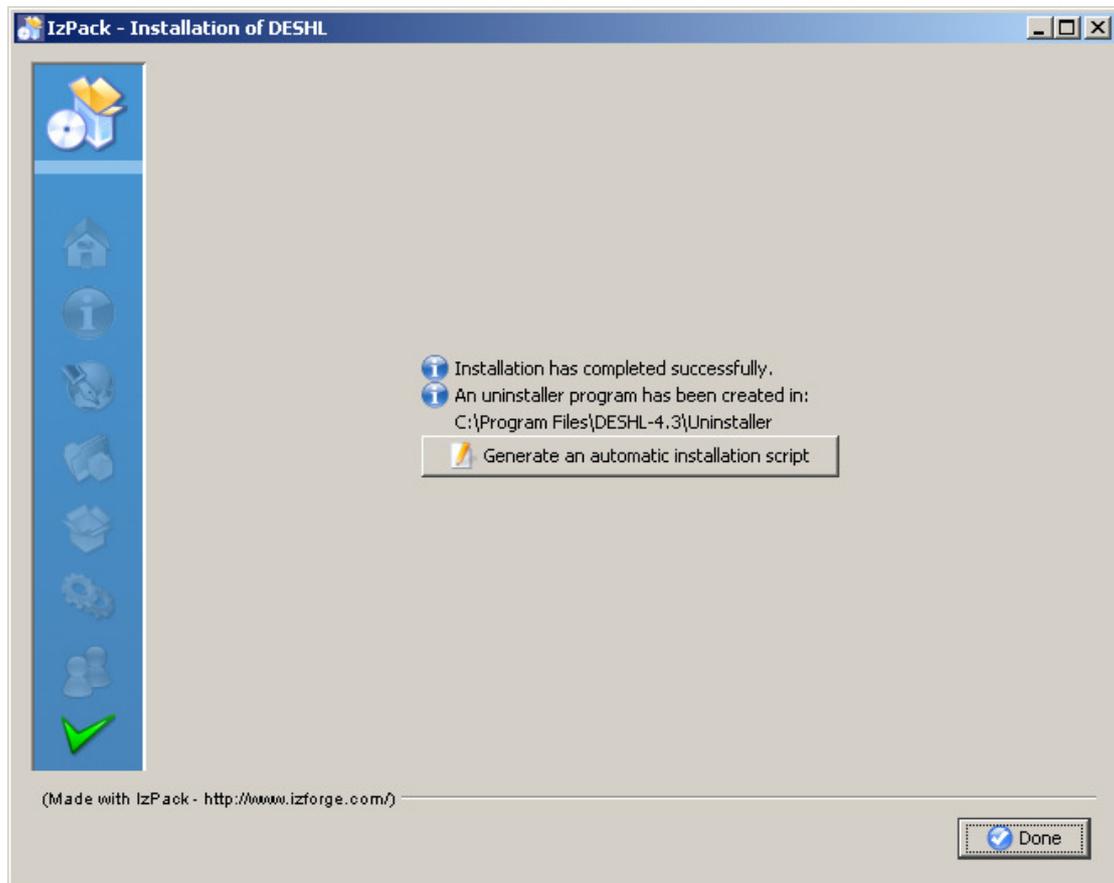


Figure 8: Installation Complete

Click on the “Done” button to exit from the installer.

Note that 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
```

Note that if your shell is `csh` or `tcsh`, you will need to use `setenv` instead of `export`.

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

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

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 “epcc-hpcx”, 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.

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.)

```

$ deshl -h

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

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

With commands:
  fetch
  cp
  isdir
  mv
  status
  jobs
  rm
  mkdir
  ls
  submit
  sites
  terminate
  isfile
  exists

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

```

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:

```
$ deshl ls ssl://admin.hpcx.ac.uk:4433/EPCC-HPCx/deisa_home/
```

Or if using shortcuts:

```
$ deshl ls epcc-hpcx::deisa_home
```

For the site, the DESHL client should display a list of the files and directories in the top level of the DEISA_HOME storage. (The DEISA_HOME storage is an area on the DEISA General Purpose File System available to each user.)

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 = $DEISA_HOME/jobs/hello.sh > hello.sh
#$ SAGA_HostList = epcc-hpcx
#$ 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. Insert the appropriate shortcut for the site where you want to submit the job.

```
#$ SAGA_HostList = epcc-hpcx
```

The `hello-submit.sh` file you have edited now contains a simple 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:

```
$ deshl exists epcc-hpcx::deisa_home/jobs
```

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

```
exists: target epcc-hpcx::deisa_home/jobs exists
```

Otherwise, the following message is displayed:

```
exists: target epcc-hpcx::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 mkdir` command:

```
$ deshl mkdir epcc-hpcx::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.

```
$ desh1 ls epcc-hpcx::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 `desh1 cp` command:

```
$ desh1 cp hello.sh epcc-hpcx::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:

```
$ desh1 cp -f hello.sh epcc-hpcx::deisa_home/jobs/hello.sh
```

4. Submit the job

Now that your script is present at the remote site, you can submit a job to run it.

```
$ desh1 submit hello-submit.sh
Your job: epcc-hpcx%2F957383131, has been successfully submitted.
```

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

5. Get the job status

Get the job's current status. Please note that the job identifier shown in the examples is for illustration only, you will need to use the job identifier returned when you submitted your job.

```
$ desh1 status epcc-hpcx%2F957383131
Job: epcc-hpcx%2F957383131, has status: Running
```

You can obtain extended information about the job using the status command with the `-f` option:

```
$ deshl status -f epcc-hpcx%2F957383131

Job: epcc-hpcx%2F957383131, has status: Done
Site: ssl://admin.hpcx.ac.uk:4433/EPCC-HPCx
Name: hello.sh
Running (Unicore:RUNNING): 10/08/2008 18:24:05
Done (Unicore:FINISHED):10/08/2008 18:26:19
```

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

6. 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 epcc-hpcx%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. The files will be named `<job identifier>.out` and `<job identifier>.err`, in this example `epcc-hpcx%2F957383131.out` and `epcc-hpcx%2F957383131.err`.

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 = $DEISA_HOME/jobs/concat.sh > concat.sh
#$ SAGA_FileTransfer = $DEISA_HOME/tutorial/filea > filea
#$ SAGA_FileTransfer = $DEISA_HOME/tutorial/fileb > fileb
#$ SAGA_FileTransfer = $DEISA_HOME/tutorial/filec < filec
#$ SAGA_HostList = epcc-hpcx
#$ 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 cp` command to import the executable script file, `concat.sh`, to the jobs directory on the remote site.

```
$ desh1 cp concat.sh epcc-hpcx::deisa_home/jobs/concat.sh
$ desh1 ls epcc-hpcx::deisa_home/jobs/concat.sh
```

Use the `deshl mkdir` and `deshl cp` 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 ls` command.

For example:

```
$ desh1 mkdir epcc-hpcx::deisa_home/tutorial
$ desh1 cp localFileA.dat
epcc-hpcx::deisa_home/tutorial/filea
$ desh1 cp localFileB.dat
epcc-hpcx::deisa_home/tutorial/fileb
$ desh1 ls epcc-hpcx::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 ls epcc-hpcx::deisa_home/tutorial/filec
-rw-    63    Mar 01 2007 03:29:57    filec
```

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

```
$ desh1 cp epcc-hpcx::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 = $DEISA_HOME/jobs/listmodules.sh > listmodules.sh
#$ SAGA_HostList = epcc-hpcx
#$ 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 cp` command to import the executable script file, `listmodules.sh`, to the `jobs` directory on the remote site.

```
$ desh1 cp listmodules.sh epcc-hpcx::deisa_home/jobs/listmodules.sh
$ desh1 ls epcc-hpcx::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 epcc-hpcx%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
blassmp/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)  lammmps/01Oct06(default)
cpmd/3.11(default)         torb/1.22(default)      lammmps/12Feb07
cpmd2cube/jan05           torbwsmp/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

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.

Appendix II – Examples of Certificate Validation Failure

