The DESHL Client

Malcolm Illingworth, EPCC

4th DEISA Training Session
30th May – 1st June 2007
CSC

DESHL and JRA7 Objectives

DESHL is developed by the DEISA JRA7 Research Activity:

"To develop a single way of coordinating and integrating Open Grid Service Architecture services for distributed resource management in a heterogeneous environment, and to use this to integrate a variety of existing user-level tools to provide the necessary high-level services in:

- authentication, authorization and accounting;
- job preparation, submission and monitoring;
- data movement for job input and output;
- other areas to be determined by DEISA user requirements."

What is the DESHL ?

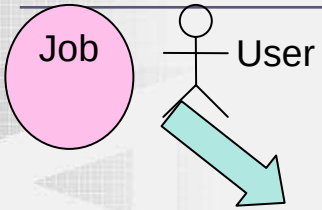
- ***DESHL: DEISA Services for the Heterogeneous management Layer***

- Command line application and application programming interface
- Distributed resource management for UNICORE-based grids
- Submit and manage batch jobs
- Uniform data management across different platforms
- Based on emerging grid standards

How is the DESHL useful?

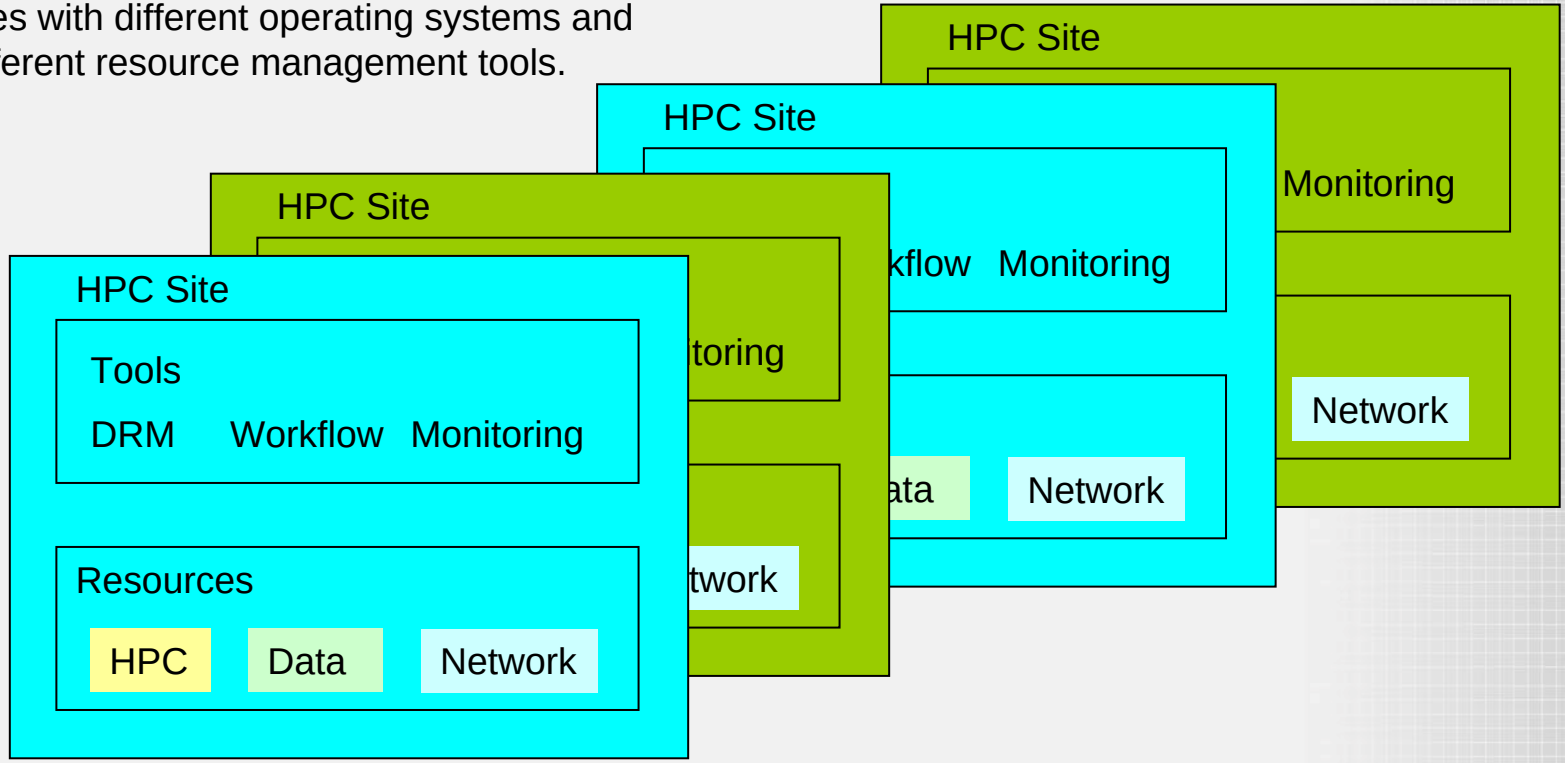
- Under the DEISA access model, direct login to any site other than the user's home site is not permitted
- DESHL provides simplified and seamless access to DEISA resources without needing to log into individual sites
- User can transfer files between any DEISA sites
- User can submit and monitor batch jobs at any DEISA site
- Not dependant on UNICORE graphical client
- Lightweight client application – quick download and install
- Simple API for integration with client applications
- Supporting emerging Grid standards

The Big Picture (1)

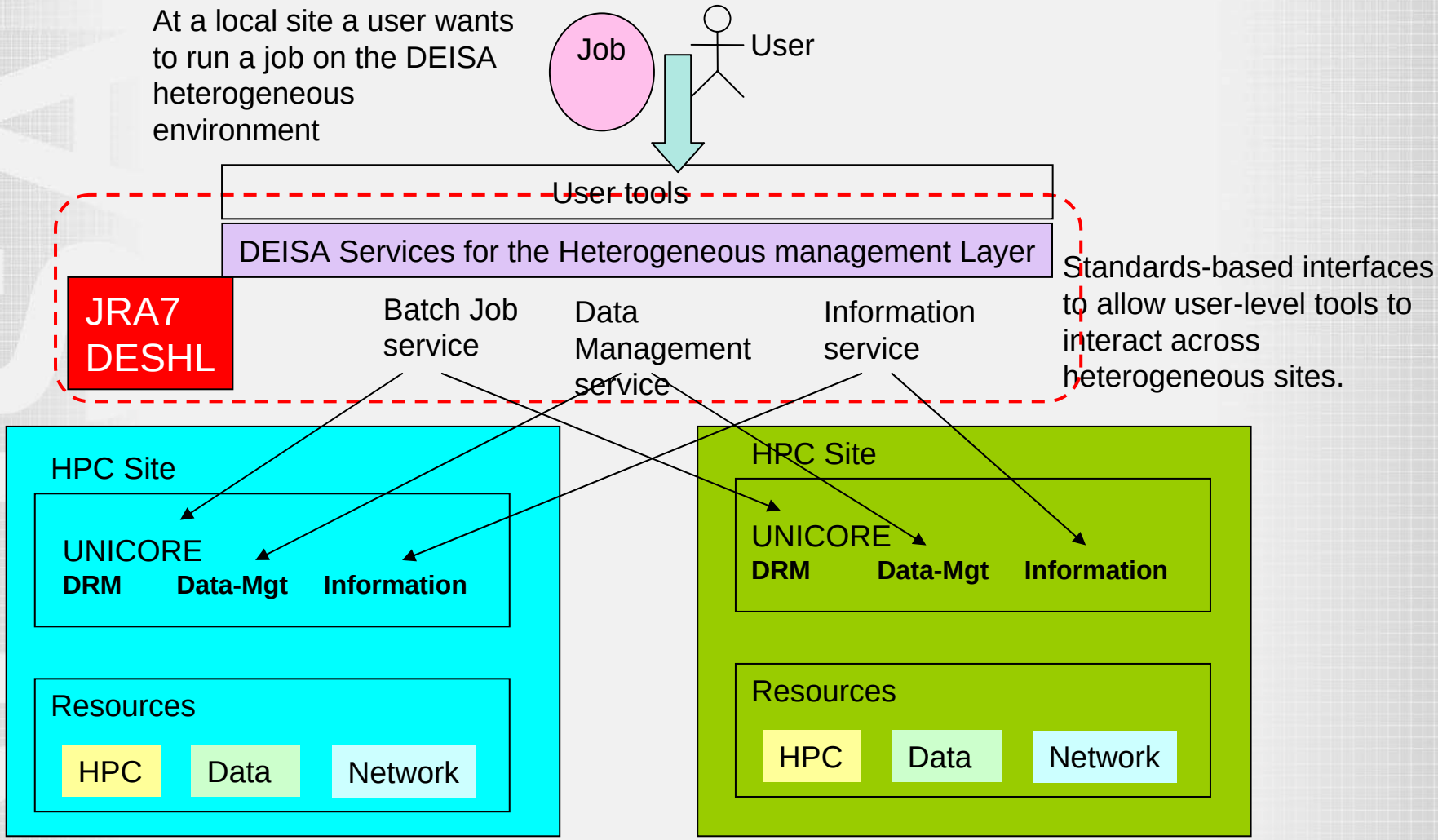


User has a compute job or a network or resource monitoring request,…

DEISA Heterogeneous Environment: HPC sites with different operating systems and different resource management tools.



The Big Picture (2)

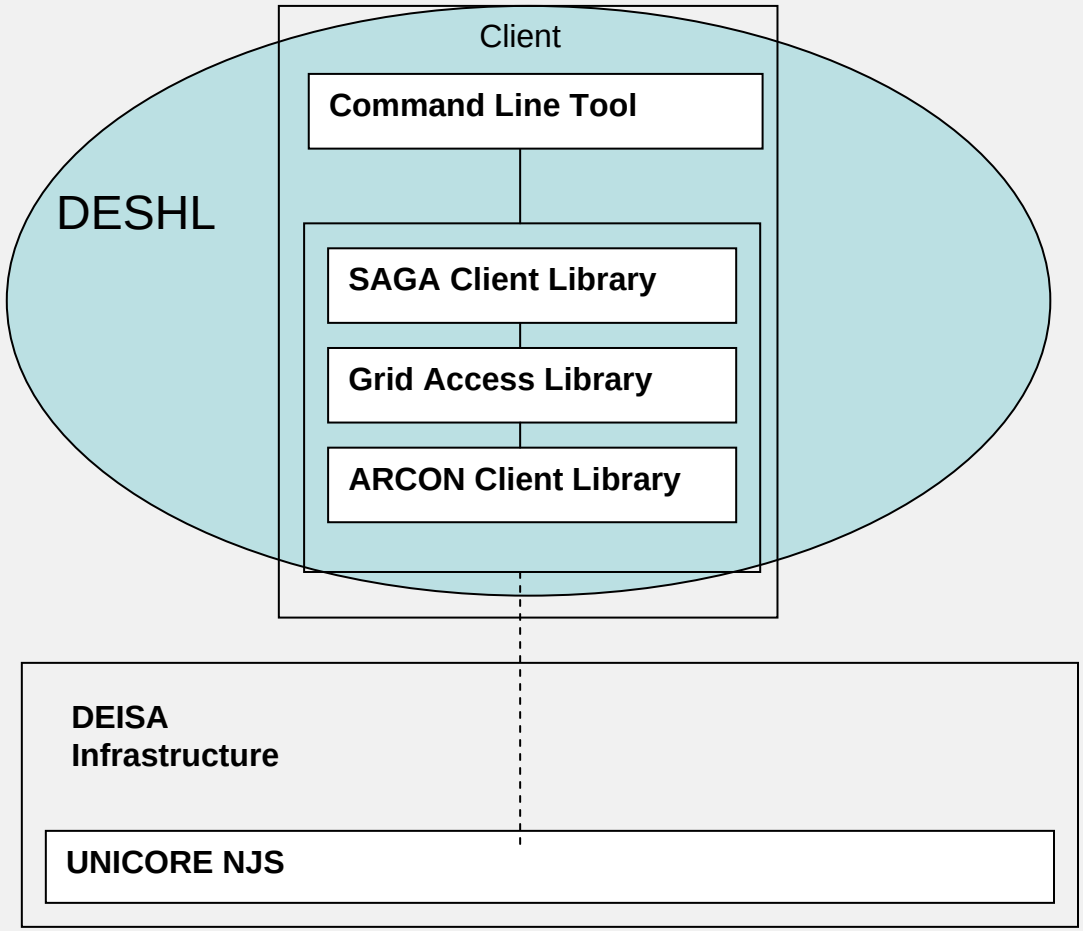


DESHL Client Overview

- Supplied as a Java command-line application
- Follows Open Grid Service Architecture (OGSA) specification for job batch submission and management
- Uses SAGA (Simple API For Grid Applications) directives for job specification
- Where appropriate, DESHL commands follow the Open Group Technical Standard for Batch Environment Systems
- Layered design to protect against future changes in underlying infrastructure
- Sits on top of existing UNICORE infrastructure

DESHL v4.0 Architecture

Layered Design sitting on top of existing Unicore infrastructure



What is SAGA?

- **S**imple **A**PI for **G**rid **A**pps RG
- Open Grid Forum Research Group
- “Provide a simple API that can be used with much less effort compared to the vanilla interfaces of existing grid middleware”
- “Provide a standardized, common interface across various grid middleware systems and their versions”
- “Driven by actual application needs”
- https://forge.gridforum.org/projects/saga-rg/

DESHL Components

- SAGA Client Library
  - Follows SAGA standards
  - Hides implementation details of underlying infrastructure
  - Exposes simple, compact API for job management and data staging
  - API can be used to develop simple standards-based grid applications
- Grid Access Library (roctopus)
  - Provides access to UNICORE resources through ARCON client library, but hides low-level nature of ARCON
  - Provides rich object-oriented interfaces for grid access
  - API can be used to build complex grid applications

DESHL Functionality - Overview

Distributed European Infrastructure for Supercomputing Applications

- File Transfer / Management
  - upload a file from local workstation to a DEISA site
  - download a file from a DEISA site to local workstation
  - delete files at a DEISA site
  - copy/move a file or directory between DEISA sites
  - list file properties / directory contents at a DEISA site
- Job Management
  - submit a batch job to a DEISA site
  - view the status of batch jobs at a DEISA site
  - terminate a batch job at a DEISA site
  - Retrieve output for a completed/terminated job
- Authentication/Authorization is by existing UNICORE mechanisms
  valid X.509 certificates for UNICORE are also valid certificates for DESHL

Client install and configuration

- Installation by GUI installer
- Access to DEISA sites controlled through local configuration file
- Only those DEISA sites which have entries in the host configuration file can be accessed by the DESHL client
- Installer configures access to user's home site
- Minor editing required to enable access to all DEISA sites

Example configuration file

Site Name	*Certificate*	*Password*	*Alias*
ssl://admin.hpcx.ac.uk:4433/EPCC%20HPCx	c:/deisa.p12		hpcx
ssl://admin.hpcx.ac.uk:4433/IDRIS%20ZAHIR	c:/deisa.p12		idris
ssl://admin.hpcx.ac.uk:4433/FZJ%20JUMP	c:/deisa.p12		fzj
ssl://admin.hpcx.ac.uk:4433/RZG%20SP4	c:/deisa.p12		Rzg
#ssl://deisa1.epcc.ed.ac.uk:4010/TEST	c:/test.p12		test

- The DESHL client can support individual user certificates for each site
- For DEISA, the same user certificate is used for all sites.
- Users can define optional “aliases” for DEISA sites for ease of use
- Individual sites can be commented out as required

Data Staging

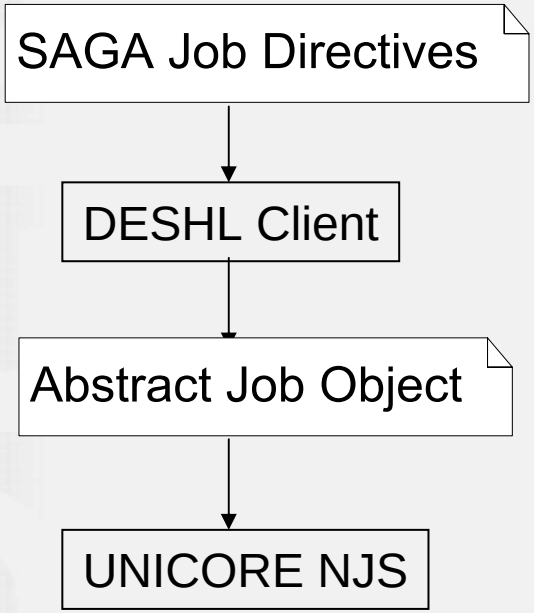
- Uses Unicore “storage” concept - equivalent to a remote mounted filesystem
- Storages for a user configured at the site
- Data staging must be between configured storages, cannot use absolute paths
- Storages at a site can be listed:
  - **`deshl list hpcx`**
- “home” storage for user data and jobs
- “root” storage for system utilities and standard applications
- Core users have access to DEISA GPFS via deisa_home and deisa_data storages

Data Staging Operations

- Upload a file from local workstation to a DEISA site
  - **`deshl copy c:/deshl/test.txt hpcx/home/test.txt`**
- Download a file to local workstation from a DEISA site
  - **`deshl copy hpcx/home/test.txt c:/deshl/test.txt`**
- List a directory or file on a DEISA site
  - **`deshl list hpcx/home/jobs/`**
- List available storages on a DEISA site
  - **`deshl list hpcx -s`**
- Copy a file or directory between DEISA sites
  - **`deshl copy hpcx/home/test.txt fzj/home/test.txt`**
- Rename a file or directory on a DEISA site
  - **`deshl move hpcx/home/test.txt hpcx/home/test2.txt`**
- Move a file or directory between DEISA sites
  - **`deshl move hpcx/home/test.txt fzj/home/test2.txt`**
- Delete a file or directory on a DEISA site
  - **`deshl remove –r fzj/home/test2/txt/`**

Job Definition

- Jobs defined by SAGA directive files
- Directives used to define job properties
- DESHL Client builds job description from directives and submits to UNICORE infrastructure as Abstract Job Object (AJO)



Simple job management scenario

Distributed European Infrastructure for Supercomputing Applications

- Executable application resides on a DEISA supercomputer
- User constructs a SAGA directives file to run the application
- User submits the job
  - deshl submit <script file>
  - returns a unique identifier for the submitted job
- User tracks the status of the job
  - deshl status <job identifier>
- When job has completed, user retrieves any stdout and stderr produced by the job
  - deshl fetch <job identifier>

Simple SAGA example

- Directives file on client, executable on server.
- UNICORE jobs run in a temporary user space (USPACE) created for the job.
- Required data and executables must be staged in and out using SAGA_FileTransfer directives.
- File paths specified using UNICORE notation:
- file:///location#<storage_name>

```
#!/bin/bash
# Test job file for DESHL;
# This is read by the client to specify the job
# Actual executable is hpcx/home/jobs/hello.sh

# SAGA JobDefinition based directives:
#$ SAGA_JobCmd = hello.sh
#$ SAGA_FileTransfer = file:///jobs/hello.sh#HOME > hello.sh
#$ SAGA_FileTransfer = file:///data/out.txt#HOME < out.txt
#$ SAGA_HostList = hpcx
```

Supported SAGA Directives (1)

- SAGA_JobCmd – The job executable path
- SAGA_FileTransfer – a file transfer local to the host where the job is to run, to stage in files required by the job to its uspace or stage out data produced by the job from the uspace.
  - Note: SAGA_FileTransfer and SAGA_JobCmd are the only directives which MUST be specified to run a job
- SAGA_JobArgs – Arguments to be passed to the job to be treated as command line arguments
- SAGA_JobEnv – Set an enviroment variable for the job to use and send information to UNICORE
- SAGA_JobName – a descriptive name for the job

Supported Directives (2)

- SAGA_HostList – The destination host for the job (can also be specified at the command line)
- SAGA_NumTasks – The number of tasks (processors)
- SAGA_NumCpus – The number of threads per task
- SAGA_WallClockSoftLimit – The time the job should run for in seconds
- SAGA_Memory – policy decided by Target System Interface (TSI)

Submitting a batch job

- Submit a job
  - **`deshl submit [options] <directives_script> <job arguments>`**
  - Submission is asynchronous
- Returns a unique job identifier
  - For example, **`hpcx%2F-367296226`**
  - Job identifier contains host site and job identification
  - Returned job identifier subsequently used with
    - **`deshl status <job identifier>`**
    - **`deshl terminate <job identifier>`**
    - **`deshl fetch <job identifier>`**

Job Submission Options

- Supported command line options (can also be specified in SAGA directives file):
  - “n” A descriptive name for the job
    - SAGA equivalent SAGA_JobName
  - “q” The DEISA site to which a job is to be submitted
    - SAGA eqivalent SAGA_HostList
  - “v” Enviroment variables to be passed to the job
    - SAGA equivalent SAGA_JobEnv
- Example:
  - **$ deshl submit –q hpcx –n “My test job” –v inputfile=“data.txt” –v outputfile=“results.txt” jobdirectives.sh**
  - **Your job: hpcx%2F-367296226, has been successfully submitted**

Job Status

- Uses job identifier returned by submit command
  - **deshl status <job identifier>**
- Returned state is one of Running, DoneOk, Failed
- Can return simple job status or extended job details

```
$ deshl status hpcx%2F-367296226
Job: hpcx%2F-367296226, has status: DoneOk

$ deshl status hpcx%2F-367296226 -f
Job: hpcx%2F-367296226, has status: DoneOk
        Site: ssl://admin.hpcx.ac.uk:4433/EPCC%20HPCx
        Name: poe
        Running 10/02 04:55:43
        DoneOk 10/02 04:57:57
```

Tracking submitted jobs

- List all current jobs
  - Looks for submitted jobs on ALL configured sites
  - Lists job identifiers
  - **`deshl jobs`**
- List jobs at a specific site
  - Only looks for jobs on the specified site
  - **`deshl jobs -q hpcx`**
- List extended job detail
  - Lists job name, status, submission time etc.
  - **`deshl jobs -f`**
  - **`deshl jobs -f -q hpcx`**
  - Output for each job is the same as "**`deshl status`**"

Example job listing

```
$ deshl jobs
ssl%3A%2F%2Fadmin.hpcx.ac.uk%3A4433%2FEPCC%2520HPCx%2F677469416
ssl%3A%2F%2Fadmin.hpcx.ac.uk%3A4433%2FEPCC%2520HPCx%2F-584506976

$ deshl jobs –f
ssl%3A%2F%2Fadmin.hpcx.ac.uk%3A4433%2FEPCC%2520HPCx%2F677469416
        Site: ssl://admin.hpcx.ac.uk:4433/EPCC%20HPCx
        Name: example MPI application running on 32 processors
        Running 10/02 05:30:18
        DoneOk 10/02 05:32:31

ssl%3A%2F%2Fadmin.hpcx.ac.uk%3A4433%2FEPCC%2520HPCx%2F-584506976
        Site: ssl://admin.hpcx.ac.uk:4433/EPCC%20HPCx
        Name: example MPI application running on 32 processors
        Running 10/02 05:30:53
        DoneOk 10/02 05:33:05
```

Terminate a running job

- **`deshl terminate <job identifier>`**
- Terminates a running job
- Stops job from executing but does not free resources
- After termination, a job has a status “DoneFailed”
- Any stderr or stdout produced prior to terminate can be subsequently retrieved with a **`deshl fetch`** command

- **`$ deshl terminate hpcx%2F-1210854115`**
  - **`Job: hpcx%2F-1210854115, terminated.`**
- **`$ deshl status hpcx%2F-1210854115`**
  - **`Job: hpcx%2F-1210854115, has status: DoneFail`**

Fetch a job

- **`deshl fetch <job identifier>`**
- Frees any resources for a completed or terminated job and retrieves stdout and stderr produced by the job
- Completed, terminated or failed job and its resources (USPACE etc) remains on host until explicitly fetched by the user
- Can retrieve output files to a specific directory
  - **`deshl fetch –d <directory> <job identifier>`**

Simple scenario (1)

- Server-side script, hpcx/home/jobs/deshlcat.sh takes two filenames, concatenates them to a third specified filename

```
#!/bin/bash

echo $inputfilea
echo $inputfileb
echo $outputfile

cat $inputfilea $inputfileb > $outputfile
```

deshlcat.sh

Simple scenario (2)

- Client-side script to submit job
- Input and output files specified as environment variables
- Assumes input data files in “data” directory on HOME storage
- File staging used to retrieve output file from USPACE
- Output file retrieved to “data” directory on HOME storage

```
#!/bin/sh
# Test job script for DESHL using SAGA.
#
# SAGA JobDefinition based directives:
#$ SAGA_FileTransfer = file:///jobs/deshlcat.sh#HOME > deshlcat.sh
#$ SAGA_JobCmd = deshlcat.sh
#$ SAGA_FileTransfer = file:///data/file1.dat#HOME > file1.dat
#$ SAGA_FileTransfer = file:///data/file2.dat#HOME > file2.dat
#$ SAGA_HostList = hpcx
#$ SAGA_JobEnv = inputfilea=file1.dat
#$ SAGA_JobEnv = inputfileb=file2.dat
#$ SAGA_JobEnv = outputfile=file3.dat
#$ SAGA_FileTransfer = file:///data/output.dat#HOME < file3.dat
```

Load Leveller Image Processing Example

- Edge detection executable, running on 16 processors, 1 thread per process
- Specifies poe as the job, staged in from the ROOT storage
- Executable is imagempi.aix, staged in from the user's HOME storage
- Input data edge192x360.dat, staged in from the user's HOME storage
- Output file image.pgm, staged out to the user's HOME storage
- **`deshl submit –q hpcx improc.sh`**

```
#!/bin/sh
# image processing example
#
#$ SAGA_FileTransfer = file:///data/edge192x360.dat#HOME > edge192x360.dat
#$ SAGA_FileTransfer = file:///jobs/imagempi.aix#HOME > imagempi.aix
#$ SAGA_FileTransfer = file:///bin/poe#ROOT > poe
#$ SAGA_NumTasks = 16
#$ SAGA_NumCpus = 1
#$ SAGA_JobCmd = poe
#$ SAGA_JobArgs = imagempi.aix
#$ SAGA_JobName = image processing example
#$ SAGA_FileTransfer = file:///results/image.pgm#HOME < image192x360.pgm
```

Code example: job submission

```
// get the class factory
JobService js = DESHLSessionFactory.getSharedSAGAFactory().getJobService();

JobDefBuilder jobDefBuilder = new JobDefBuilder();
... // build up job definition from file or arguments
// get the constructed job definition
JobDescription jobDescription = jobDefBuilder.create();

// submit the job, return a job instance
Job submittedJob = js.submitJob( jobDescription );

// get the job identifier, eg to display to the user
String jobID = job.getJobId();

// get the job instance again from the job identifier
Job remoteJob = js.getJob(jobID);

// get the job's status
JobState jobState = remoteJob.getJobStatus();

… // loop until the job has completed or failed

// retrieve the job output to a specified directory
remoteJob.fetch("/home/malcolm/joboutputdir");
```

Code example: File copy

```
int copyFlags[] = {
   NSDirFlags.copyFlags_NoRecursive, NSDirFlags.NoOverwrite };

// DESHL allows shortcutting of path names
// otherwise this would have to be
// ssl://admin.hpcx.ac.uk:4433/EPCC%20HPCx
String source = “hpcx/home/malcolm/test.dat";
String target = “idris/home/malcolm/test.dat";

// get an instance of the factory
NSDir dir = DESHLSessionFactory.getSharedSAGAFactory().getNSDir();

// verify the source file exits
boolean sourceFileExists = dir.exists(source);
// copy the file to the other site
dir.copy(source, target, copyFlags);
// verify the file turned up at the remote site
boolean targetFileExists = dir.exists(target);
assertTrue(targetFileExists);
```

Current status and future plans

Distributed European Infrastructure for Supercomputing Applications

- DESHL 4.1 in beta development
- Local persistent server
  - Secure password handling, passwords entered once per session
  - Passwords not stored in configuration files
  - Lightweight client
- Integration with GridFTP
  - Much faster uploads/downloads from local workstation
- Improved exception handling
- Standards compliance
- Possible inclusion of DESHL client library by eDEISA for lifesciences portal development (integration with EngineFrame)

DESHL Summary

Distributed European Infrastructure for Supercomputing Applications

- The DESHL software has been developed by the seventh Joint Research Activity (JRA7) of the EU-funded DEISA project. EPCC and ECMWF from the UK, FZJ from Germany and CINECA from Italy are the participants in this research activity.
- DESHL allows user and applications to manage batch jobs and data in a uniform manner regardless of underlying differences in hardware and software
- DESHL employs emerging grid standards so that users and applications are not affected by changes in underlying software
- For further information and to download the latest version of the DESHL Client software:
  - http://deisa-jra7.forge.nesc.ac.uk/
- DESHL email address:
  - deisa-jra7-deshl@forge.nesc.ac.uk