

# UNICORE

## Using the commandline client

Bernd Schuller, FZ Jülich

# Outline

---

- Overview
- Installation
- Configuration overview
- Commands & Demos

# Overview

---

- UCC: Commandline client for UNICORE 6
- Access to many UNICORE 6 features
  - Deal with jobs (submit, check status, get output)
  - Deal with file transfers (remote-local, local-remote, remote-to-remote)
  - Get information (list sites, jobs, applications)
- Batch execution of jobs
- Scripting capability using Groovy
- Extensible

# Prerequisites

---

- Java 5 JRE or SDK (or later)
  - <http://java.sun.com>
- Access to a UNICORE 6 Grid (user certificate!)

# Installation

---

- Available on Sourceforge (right next to the quickstart and GPE client bundles)
- Unzip in any directory
- Contents
  - bin - executable „ucc“ (shell script calling Java code)
  - lib - libraries
  - conf - example config file
  - samples - example files
  - certs - keystore containing „demo user“

# Configuration

---

- None required!
- ...but it is helpful to
  - generate a config file „~/ucc/preferences“
  - add „bin“ dir to path
- Preferences can save lots of typing...

```
$>cat ~/.ucc/preferences
```

```
registry=https://localhost:8080/DEMO-SITE/services/Registry?res=default_registry  
keystore=/home/schuller/certs/bernd-schuller.p12  
storetype=pkcs12  
password=notarealpassword
```

# Getting help

- List available commands using „ucc -h“

```
$>ucc -h
```

```
UCC version 1.0-SNAPSHOT
```

```
Usage: ucc <command> [OPTIONS] <args>
```

```
The following commands are available:
```

```
copy-file          - copy remote files
get-status         - get job status
wsrf               - perform a WSRF operation
resolve           - resolve remote location
list-jobs         - list your jobs
batch             - run ucc on a set of files
get-output        - get output files
get-file          - get remote files
run               - run a job through UNICORE 6
put-file          - put local file to a remote server
list-sites        - list remote sites
list-applications - lists applications on target systems
connect           - connect to UNICORE
copy-file-status  - check status of a copy-file
run-groovy        - run a Groovy script
Enter 'ucc <command> -h' for help on a particular command.
```

# Getting help II

- Help on a particular command using „ucc <command> -h“

```
$>ucc get-file -h
```

```
usage: ucc get-file [OPTIONS]
```

```
Gets a file from remote location '-s' and writes it to the local file
specified by the '-t' option.
```

```
-y,--with-timing           Timing mode
-c,--configuration <Properties> Properties file containing your
                             preferences. By default, a file
                             '<userhome>/ucc/preferences' is checked.
-h,--help                  Print this help message
-k,--keystore <Keystore>   Keystore containing your user
                             credential and trusted certificates
-n,--alias <Alias>        Key alias
-o,--output <Output>      Directory for any output produced
-p,--password <Password>  Keystore password
-r,--registry <Registry>  The URL of a UNICORE registry
-s,--source <Source>      Data source
-t,--target <Target>      Data target
-v,--verbose               Verbose mode
-x,--storetype <jks|pkcs12> Keystore type (default is jks)
```

# Demonstration of basic usage

---

- „ucc connect“ - connect to UNICORE 6
  - checks registry for accessible target system, if none found create one
- „ucc run ...“ - run a job synchronously
- „ucc run -a ...“ - run a job asynchronously
- „ucc get-status ...“ - check job status
- „ucc get-output ...“ - get job output

# Job description

---

- Allows to specify
  - Executable
    - Unicore Application (name+version) or path to executable (e.g. „/bin/date“) on the remote system
    - Arguments
    - Environment variables
  - Local files to stage in
  - Remote files to stage in
  - Result files to stage out
  - Result files to export to local
- Simple ASCII format (JSON)

# Simple examples

---

- Date application

```
{  
  ApplicationName: Date,  
  ApplicationVersion: "1.0",  
}
```

- ls -l

```
{  
  Executable: "/bin/ls",  
  Arguments: ["-l"],  
}
```

- Environment settings

```
{  
  Executable: "/usr/bin/someapp",  
  Environment: ["WIDTH=320", "HEIGHT=200" ],  
}
```

# Data imports/exports

---

- Import local files into the Uspace

```
{  
  Imports: [  
    {File: "./script.sh", To: "input.sh"},  
    {File: "./data",      To: "data"},  
  ],  
}
```

- Export result files

```
{  
  Exports: [  
    {File: "./output", To: "myresult"},  
    {File: "./errorlog", To: "my_errors"},  
  ],  
}
```

- Standard out and error are fetched automatically

# Demo

---

- Importing and running a script
- Povray

# Data movement

---

- Copy files: get-file, put-file, copy-file
  - remote to local, local to remote, remote to remote
- Specify data location
  - Full UNICORE 6 URI (protocol, service address, file)

```
RBYTEIO:https://localhost:8080/  
DEMO-SITE/services/StorageManagement?res=1234567890#test.txt
```

- More user-friendly (Site, Storage, File, Protocol)

```
u6://DEMO-SITE/Home/test.txt?protocol=RBYTEIO
```

- Protocol is optional (default: rbyteio)

# Data movement

---

- Storages
  - Home
  - Job Uspace using the job id

```
u6://DEMO-SITE/cae9f8fb-655b-46bd-a2d8-a10ef356bb87/test.txt
```

- Resolving locations using „ucc resolve“

```
$>ucc resolve u6://DEMO-SITE/cae9f8fb-655b-46bd-a2d8-a10ef356bb87/test.txt  
  
https://localhost:8080/DEMO-SITE/services/StorageManagement?res=c43c2da4-63b5-4b93-8295-5ef91c3d78fe
```

# Demo: data movement

---

- get-file
- put-file
- Locations

# Getting information

---

- `ucc list-jobs`
  - lists all jobs with their status
- `ucc list-sites`
  - available sites
- `ucc list-applications`
  - applications per site
- `ucc wsrp getproperties <resource address>`
  - prints the wsrp resource property document of the given resource

# Batch mode

---

- Run many jobs „at once“ without restarting ucc all the time
- Reads jobs from a specified input directory
- Multithreaded
- Currently only „round robin“ site selection (i.e. no brokering!)
- Basic use

```
$>ucc batch -i input -o output
```

# Demo: batch mode

---

- Create and run some „Date“ jobs

# Scripting

---

- UCC can execute Groovy scripts
  - Object oriented scripting language
  - Tightly integrated with Java
  - It's groovy :-)
- Needs insight into how UNICORE 6 works
- Needs insight into how the uas-core client code works
- `ucc run-groovy -f <scriptfile>`



# Extending UCC

---

- Write your own commands
  - can extend UCC base classes
- Add code to classpath
- Set property when starting UCC
  - `-Ducc.extensions=conf/extensions`
- Extensions file contains command name and name of command class

```
$>cat conf/extensions
```

```
c9m-submit=org.chemomentum.cmdline.SubmitWorkflow  
c9m-get-file=org.chemomentum.cmdline.C9MGetFile
```