INFORMATION SOCIETY TECHNOLOGIES
(IST)
PROGRAMME

OpenMolGRID

SPECIFICATION OF THE COMMAND LINE INTERFACE TO UNICORE

Contract Reference:	**IST-2001-37238**
Document identifier:	**OpenMolGRID-4-D4.6a-117-0-1-CLISpecification**
Date:	**25/03/04**
Work package:	**WP 4: Grid Integration**
Partner:	**FZJ, UU**
Lead Partner:	**FZJ**
Document status:	**APPROVED**
Classification:	**PUBLIC**
Deliverable identifier:	**D4.6a**

Abstract: This document contains the specification of the command line client enabling programs to act as UNICORE clients.

Delivery Slip

	Name	Partner	Date
From	B. Schuller	FZJ	19/02/2004
Verified by	M. Romberg	FZJ	26/02/2004
Approved by	G.H.F.Diercksen (TC) R.Ferenczi (QE)	OMC CGX	22/03/2004 29/03/2004

Document Log

Issue	Date	Comment	Author
0-0	19/02/2004	issued for internal review	B. Schuller, M. Romberg, L. Kirtchakova, D. McCourt
0-1	25/03/2004		B. Schuller

Document Change Record

Issue	Item	Reason for Change
0-1	typos fixed, added some background material on the use of the CLI by the data warehousing procedures	comments from TC

Files

Files in this section relate to actual storage locations on the BSCW server located at https://hermes.chem.ut.ee/bscw/bscw.cgi. The URL below describes the location on BSCW from the root OpenMolGRID directory

Software Products	User files / URL
Word 2000/XP	OpenMolGRID/Workpackage 4/Deliverables/ OpenMolGRID-4-D4.6a-117-0-1-CLISpecification

Project information

Project acronym:	OpenMolGRID
Project full title:	Open Computing GRID for Molecular Science and Engineering
Proposal/Contract no.:	IST-2001-37238
European Commission:	
Project Officer:	Annalisa BOGLIOLO
Address:	European Commission - DG Information Society F2 - Grids for Complex Problem Solving B-1049 Brussels Belgium
Office	BU31 4/79
Phone:	+32 2 295 8131
Fax:	+32 2 299 1749
E-mail	annalisa.bogliolo@cec.eu.int
Project Coordinator:	Mathilde ROMBERG
Address:	Forschungszentrum Jülich GmbH ZAM D-52425 Jülich Germany
Phone:	+49 2461 61 3703
Fax:	+49 2461 61 6656
E-mail	m.romberg@fz-juelich.de

Contents

1. INTRODUCTION....................5

1.1. Purpose and Scope....................5
1.2. Document Overview....................5
1.3. Document Structure....................5

2. REQUIREMENTS AND USE CASES....................6

2.1. Functional Requirements....................6
2.2. Non-functional Requirements....................6
2.2.1. Environment....................6
2.2.2. Security....................6
2.3. Use Cases for the OpenMolGRID Data Warehouse (MOLDW)....................6
2.3.1. Fingerprinting....................7
2.3.2. Substructure search....................7
2.3.3. 2D to 3D Conversion....................8
2.3.4. Descriptor calculation....................9

3. SPECIFICATION OF THE COMMAND LINE INTERFACE....................10

3.1. Overview....................10
3.2. Detailed Command Specification....................10
3.2.1. Creation of an AJO....................10
3.2.2. Submission of a job....................10
3.2.3. Querying job status....................11
3.2.4. Retrieving job results....................11
3.2.5. Aborting a job....................12
3.3. Specification of the AJO_ID File....................12
3.4. Specification of XML Workflow Files....................13

4. REFERENCES....................14

5. TERMINOLOGY / GLOSSARY....................15

1. Introduction

1.1. Purpose and Scope

One of UNICORE's main assets is its powerful client, featuring a graphical user interface (GUI) that allows the user to create jobs, monitor them, get their results and abort or delete them. Based on the user input, the GUI client creates an Abstract Job Object (AJO) that is sent to a UNICORE site for execution.

However, access by a program to a UNICORE Grid is currently not well supported by the UNICORE middleware. While the GUI client provides a GUI-less mode that allows to send a pre-built AJO to the Grid, it is not possible to build a new job dynamically without manual user intervention. To fill this gap, a new UNICORE component, a command line interface (CLI) to the UNICORE Grid is needed.

This document contains the specification of this new command line interface. The job generation facility provided by the CLI will be based on the OpenMolGRID MetaPlugin, therefore parts of the MetaPlugin specification (Deliverable D4.2a [1]) will also be relevant to this document.

Within the OpenMolGRID project, the need for such a tool providing access to compute resources on the Grid arises within the data warehousing part of the project. The OpenMolGRID Data Warehouse (MOLDW) [2] needs to perform several computations during its data upload procedures. These computations serve to illustrate the possible usage of the CLI.

1.2. Document Overview

This document details the requirements for the command line interface. Possible usage of the CLI is illustrated by several use cases developed in the data warehousing part of the project.

The syntax and parameters of the operations that have to be provided by the CLI are specified. The format of the input to the CLI and the format of the intermediate files produced by the CLI are specified.

1.3. Document Structure

This document contains the following sections. Section 2 specifies the functional and non-functional requirements for the CLI, and the use cases for CLI usage by the OpenMolGRID data warehouse MOLDW. In Section 3, the CLI command syntax, parameters, and in/output file formats are specified. Finally, section 4 gives the references used in this document, and section 5 lists the terms and abbreviations used.

2. Requirements and Use Cases

2.1. Functional Requirements

The CLI will enable command line access to UNICORE. Since UNICORE is based on the exchange of signed Abstract Job Objects (AJOs), the CLI will need to provide facilities to

- generate AJOs from a job description,
- submit AJOs to UNICORE sites using SSL connections to UNICORE gateways,
- query the status of submitted jobs,
- retrieve results,
- abort running jobs.

The generation of AJOs has to be flexible enough to satisfy the needs of the applications within OpenMolGRID, while still being application-neutral, in order to provide a tool that is of wide use to the UNICORE community.

Since no user interaction is possible during job creation and execution, all information necessary for building the job and submitting it to the UNICORE Grid has to be provided to the CLI.

2.2. Non-functional Requirements

2.2.1. Environment

For reasons of portability and re-use of existing code, the CLI will be written in Java.

The job generation part of the CLI will re-use code from the MetaPlugin that is used in the GUI client to build jobs using XML workflow descriptions, available resources and user defaults. The MetaPlugin and its supporting components are specified in detail in [1].

2.2.2. Security and Authentication

The CLI allows programs or processes to access the UNICORE Grid as a user, i.e. submit jobs, etc. For authentication, the UNICORE security infrastructure uses X.509 certificates (see also [3]). Every program/process that wishes to access the Grid needs such a certificate. This certificate needs to be issued and signed by a certificate authority (CA) that is accepted by the UNICORE server components (Gateway and NJS). Furthermore, the certificate to be used needs to be entered as a user certificate into the user databases at the NJSs that are to be accessed.

The CLI needs access to this user certificate and the signer certificate of the CA in order to connect to the gateway(s).

2.3. Use Cases for the OpenMolGRID Data Warehouse (MOLDW)

Within OpenMolGRID, the CLI will be used extensively by the OpenMolGRID Data Warehouse (MOLDW). Before describing the use cases, some background on MOLDW's need for access to Grid resources is needed here. In a nutshell, the data warehouse will contain a small set of molecular descriptors, which are derived via computationally intensive descriptor calculation programs, from the molecular structure information of the stored compounds. From a data warehousing perspective, these descriptor calculations could be viewed as complex data transformations. As descriptor calculation programs are often specialised in terms of required hardware and software, it is desirable to use these programs as an external resource, accessed through Grid middleware. Essentially, this amounts to virtualisation of parts of the data warehouse's data transformation processes.

Several use cases have been prepared, which serve to illustrate possible uses of the CLI, but do not fully define the scope of the CLI. In fact, the CLI will provide a general client to a UNICORE Grid.

2.3.1. Fingerprinting

Scenario: When MOLDW is carrying out its upload tasks it may come across a number of potentially new chemical structures. In order to help with substructure search MOLDW will pre-compute the molecular fingerprint for the new structure.

Preconditions:
1) Data Upload process is running

Trigger: A potentially new (set of) structure(s) (might be 2D or 3D) is found during the data upload process

Description:
1) If the set of structures is less than X (to be determined)
   a) Compute fingerprints locally
   b) Load results to MOLDW
2) Else
   a) Pass the structures as input to a fingerprinting workflow using the CLI
   b) Receive response (automatically) and load results to MOLDW

Extension:

Variations:

Postconditions: MOLDW will contain the new structures and their fingerprints.

2.3.2. Substructure search

Scenario: During the MOLDW data upload process potentially new structures will be found. In order to prevent duplication, substructure searching against all structures in MOLDW must take place before it is accepted as a "new" chemical or as an alternative representation of an existing one.

Preconditions:
1) Data Upload process is running

Trigger: A potentially new (set of) structure(s) is found during the data upload process

Description:
1) Calculate its fingerprint (Fingerprint use case)

2) Generate a potential fingerprint match list
3) Pass structures as input to a substructure search program using the CLI
4) Retrieve results (automatically)
5) If not duplicate
    a) Process as a new chemical
6) Else
    a) Process as an existing chemical

Extension:

Variations:

Postconditions: A list of structures that match and require further processing.

2.3.3. 2D to 3D Conversion

Scenario: When MOLDW is carrying out its upload tasks it may come across a number of new chemical structures. These structures may be represented in 2D format. MOLDW will also require the 3D format to be generated.

Preconditions:
1) Data Upload process is running
2) The structure is a 2D representation

Trigger: A (set of) 2D structure(s) is found during the data upload process for which 3D structures are not available

Description:
1) Submit the 2D structures as input to a 2D to 3D conversion workflow using the CLI
2) Receive response (automatically) and load results to MOLDW

Extension:

Variations:

Postconditions: MOLDW will contain the new 3D structure(s).

2.3.4. Descriptor calculation

Scenario: During the data upload process of MOLDW a new structure is found and the new structure requires descriptors to be calculated.

Preconditions:
1) Data Upload process is running
2) The structure is a 3D representation

Trigger: Substructure search has found a new structure

Description:
1) Pass structure as input to Descriptor Calculation software via the CLI
2) Retrieve output (automatically)
3) Load Descriptor Calculation results to MOLDW

Extension:

Variations: 2) The structure is a 2D representation and requires conversion to 3D first.

Postconditions: MOLDW will contain Descriptors for the new chemical.

3. Specification of the Command Line Interface

3.1. Overview

The CLI needs to provide the following commands:

- `build_ajo`: create an AJO for submission from a workflow description
- `submit`: submit an AJO
- `get_status`: query job status
- `get_outcome`: retrieve outcome for a job
- `abort_job`: abort job execution

After submission, jobs are referenced by an AJO_ID_FILE, containing the information needed to get job status, retrieve the outcome, or abort the job.

The syntax parameters of the individual CLI commands are given in section 3.2, while the AJO_ID_FILE content and syntax is specified in section 3.3.

All commands need access to the keystore containing the user certificate of the CLI. The password needed to "unlock" the keystore can be supplied on the command line, alternatively, the name of a file containing the password can be given.

3.2. Detailed Command Specification

This section gives syntax and parameters for all CLI commands.

3.2.1. Creation of an AJO

Synopsis:

`build_ajo` - Builds an abstract job object from an XML workflow definition and saves it to file.

Syntax:

```
CLI build_ajo [-defaults <defaults>] -in <xml_file> -out <ajo_file>
-keystore <keystore> [-password <password> | -pwfile <pwfile>]
```

Description of parameters:

`<xml_file>`	XML workflow file
`<ajo_file>`	Relative path to the destination file of the abstract job object
`<keystore>`	Absolute path to the keystore file, plus CA certificate files in the same directory
`<password>`	Password to unlock the keystore
`<pwfile>`	Absolute path to a file containing the keystore password
`<defaults>`	File containing users defaults

Remarks:

Defaults.txt contains user preferences about the execution of the job.

If the keystore password is not given with the -password option, a file containing the password has to be specified using -pwfile.

3.2.2. Submission of a job

Synopsis:

`submit` - Reads an AJO from file and submits it to the Vsite defined in the AJO. Submission can be either *synchronous,* when the command will return only when the job has been finished, or *asynchronous*, when the command returns immediately.

Syntax:

```
CLI submit -in <ajo> -dir <out_dir> -keystore <keystore> [-password
<password> | -pwfile <passwordfile>]
```

or

```
CLI   submit   -in   <ajo>   -mode   async   -out   <AJO_ID_FILE>   -keystore
<keystore> [-password <password> | -pwfile <pwfile>]
```

Description of parameters:

`<ajo>`	Relative path to the abstract job file
`<out_dir>`	Directory for outcome files
`<keystore>`	Absolute path to the keystore, plus CA certificate files in the same directory
`<password>`	Password to unlock the keystore
`<pwfile>`	Absolute path to a file containing the keystore password
`<mode>`	Submission mode {sync| async}. Default: sync
`<AJO_ID_FILE>`	Relative path to the AJO ID file (see remark below)

Remarks:

Depending on the submission mode the result of this command can either consist of outcome files in the defined directory or an AJO_ID file. If submission mode is synchronous, the CLI will wait until the job has finished executing, retrieve the job results, and delete the respective job from the NJS. Otherwise it will write the AJO_ID file as detailed in section 3.3.

If the keystore password is not given with the -password option, a file containing the password has to be specified using -pwfile.

3.2.3. Querying job status

Synopsis:

`get_status` - Get the status of the job identified by the given AJO_ID file. The status is saved to the specified file, or written to standard output.

Syntax:

```
CLI   get_status   [-out   <status   file>]   -in   <AJO_ID_FILE>   -keystore
<keystore> [-password <password> | -pwfile <pwfile>]
```

Description of parameters:

`<AJO_ID_FILE>`	Relative path to the AJO_ID file
`<keystore>`	Absolute path to the keystore, plus CA certificate files in the same directory
`<password>`	Password to unlock the keystore
`<pwfile>`	Absolute path to a file containing the keystore password
`<status_file>`	Relative path to the file to save the job status to (stdout as default)

3.2.4. Retrieving job results

Synopsis:

`get_outcome` - Get outcome (result files) of the job identified by the given AJO_ID file. Results are stored to the directory indicated.

Syntax:

`CLI get_outcome -in <`**`AJO_ID_FILE`**`> -dir <`**`outcome_dir`**`> -keystore <`**`keystore`**`> [-password <`**`password`**`> | -pwfile <`**`pwfile`**`>]`

Description of parameters:

`<AJO_ID_FILE>`	Relative path to the AJO_ID file
`<outcome_dir>`	Directory for outcome files
`<keystore>`	Absolute path to the keystore, plus CA certificate files in the same directory
`<password>`	Password to unlock the keystore
`<pwfile>`	Absolute path to a file containing the keystore password

Remarks:

The outcome can be retrieved only if the job status is DONE. If the job is still executing, the CLI will wait until it is finished and then retrieve results. Finally, the job will be removed from the NJS.

If the keystore password is not given with the -password option, a file containing the password has to be specified using -pwfile.

3.2.5. Aborting a job

Synopsis:

`abort_job` - Abort a job identified by the given AJO_ID file and remove it from the NJS.

Syntax:

`CLI abort_job -in <`**`AJO_ID_FILE`**`> -keystore <`**`keystore`**`> [-password <`**`password`**`> | -pwfile <`**`pwfile`**`>]`

Description of parameters:

`<AJO_ID_FILE>`	Relative path to the AJO_ID_FILE
`<keystore>`	Absolute path to the keystore, plus CA certificate files in the same directory
`<password>`	Password to unlock the keystore
`<pwfile>`	Absolute path to a file containing the keystore password

Remarks:

The CLI will abort the job (if it is still running) and remove it from the NJS.

If the keystore password is not given with the -password option, a file containing the password has to be specified using -pwfile.

3.3. Specification of the AJO_ID File

The AJO_ID file is created and written to file by the `submit` command. It must contain the AJO identifier that uniquely identifies AJOs within the UNICORE system. The AJO identifier is used to identify jobs in subsequent calls to CLI commands. Additionally, the AJO_ID file must contain information enabling the program or human user using the CLI to keep track of running jobs.

To make this information easily accessible, a Java properties file is used as AJO_ID file. Thus, the information is stored as *key=value* pairs.

key	***description***	***format***
ajo_id	AJO identifier (serialized Java object)	String (encoded byte array)

key	*description*	*format*
vsite	Vsite name	free string
gateway	Gateway URL (Usite)	ssl://machine:port
job_name	job name	free string
date	job submission date	hh:mm:ss dd/mm/yyyy

3.4. Specification of XML Workflow Files

The `build_ajo` command expects a workflow description as input. To maximise re-use of existing code within the project, the CLI will accept the same XML workflow files as the OpenMolGRID MetaPlugin. Therefore, the specification of the XML workflow files given in Deliverable D4.2a of OpenMolGRID [1] is used here.

For the sake of completeness, the XML document type specifying the syntax of the XML workflows is reproduced here.

```
<!DOCTYPE workflow [
<!ELEMENT workflow (task*, group*, dependency*)>
    <!ELEMENT task (option*)>
      <!ELEMENT option EMPTY>
        <!ATTLIST option
             name     CDATA #REQUIRED
             value    CDATA #REQUIRED
        >
      <!ATTLIST task
           name        CDATA #REQUIRED
           identifier  CDATA #REQUIRED
           id          CDATA #REQUIRED
           export      (true | false) #REQUIRED
           split       (true | false) #REQUIRED
      >
    <!ELEMENT group (option*, task*, group*, dependency*)>
      <!ATTLIST group
           type        (subjob | repeat | doN | if | then | else) #REQUIRED
           identifier  CDATA #REQUIRED
           id          CDATA #REQUIRED
           split       (true | false) #REQUIRED
      >
    <!ELEMENT dependency EMPTY>
      <!ATTLIST dependency
           pred        CDATA #REQUIRED
           succ        CDATA #REQUIRED
      >
]>
```

4. References

[1] Deliverable D4.2a,
Specification of the Grid interface for classes of applications to support automated workflows

[2] Deliverable D1.1a
The OpenMolGRID Data Warehouse MOLDW

[3] Deliverable D4.5a,
Description of the OpenMolGRID Grid architecture, security architecture, and infrastructure and the deployment of the project's testbed

5. Terminology / Glossary

AJO	Abstract Job Object
CLI	Command line interface
FZJ	Forschungszentrum Jülich
GUI	Graphical user interface
JRE	Java Runtime Environment
NJS	Network Job Supervisor (UNICORE server component)
UNICORE	Uniform Interface to Computer Resources
UU	University of Ulster
WP	Work Package
XML	Extensible Markup Language