Using gLite API

Vladimir Dimitrov
IPP-BAS

"gLite middleware Application Developers Course",
Sofia, Bulgaria, 25.11.2008

Acknowledgments

This presentation is based on:

- **EGEE JRA1 documents in category "EGEE gLite User's Guide" which could be found through:**
  `https://edms.cern.ch/cedar/plsql/cedarw.home`

- **GILDA training materials and Wiki pages related to gLite APIs:**
  `https://grid.ct.infn.it/twiki/bin/view/GILDA/APIUsage`

- **gLite API documentation on:**
  http://trinity.datamat.it/projects/EGEE/wiki/wiki.php?n=WMProxyAPI.JobSubmission

- **Preinstalled documentation and examples on a gLite 3.1 WN or UI:**
  /opt/glite/share/doc

Introduction

- **gLite middleware provides several Application Progamming Interfaces (APIs) to the application developers:**
  - **Data** API
  - **Data Catalog** API for C, Java, Perl
  - **Data Delegation** API for C
  - **Data SRM** API for C, Perl
  - **Data Transfer** API for C, Java, Perl
  - **JDL** API for C++
  - **LFC** API for Java, Python
  - **R-GMA** API for C, C++, Java, Python
  - **Security VOMS** API for C, C++
  - **Service Discovery** API for C, Java
  - **WMS WMproxy** API for C++, Java, Python

    etc.

gLite Service Discovery API (1)

- **The API returns a list of service descriptors matching search criteria. Information about the individual services can be obtained from the descriptors if it is desired to rank the services returned or produce a web page of some subset of services.**

- **Prerequisites:**
  - **Service Discovery** client must be installed and correctly configured (*on WNs, UI or VObox*)
  - **Valid grid certificate**
  - **glite-service-discovery-api-* RPMs installed.**

gLite Service Discovery API (2) Using CLI

- **Prepare the environment:**

```
export GLITE_SD_PLUGIN = rgma,bdii,file
```

Obtaining a list of SRMs

Execute the command:

```
glite-sd-query -t srm
```

- **The output should be like this:**

```
Name: httpg://clrlcgse01.in2p3.fr:8443/srm/managerv1
Type: SRM
Endpoint: httpg://clrlcgse01.in2p3.fr:8443/srm/managerv1
Version: 1.1.0

Name: httpg://lpsc-se-dpm-server.in2p3.fr:8446/srm/managerv2
Type: SRM
Endpoint: httpg://lpsc-se-dpm-server.in2p3.fr:8446/srm/managerv2
Version: 2.2.0
    ...
```

gLite Service Discovery API (3) Using CLI

Obtaining a list of services running on a host. *(in XML format)*

- **Execute the command:**

```
glite-sd-query --xml --host  se03.grid.acad.bg
```

- **Output:**

```
<service name="httpg://se03.grid.acad.bg:8443/srm/managerv1">
   <parameters>

  <endpoint>httpg://se03.grid.acad.bg:8443/srm/managerv1</endpoint>
     <type>SRM</type>
     <version>1.1.0</version>
   </parameters>
 </service>
```

gLite Service Discovery API (4) Using C API

- **Some C API functions:**

```
SDService* SD_getService(const char* serviceName,
    SDException *  exception)
 // returns basic details about the requested service.

SDServiceDetails* SD_getServiceDetails(const char* serviceName,
   SDException*  exception)
 // returns full details about the requested service.

SDServiceDataList* SD_getServiceData(const char* serviceName,
   SDException* exception)
 // returns all service keyword/value data for the requested service.

char* SD_getServiceDataItem(const char* serviceName,
  const char* key, SDException* exception)
 // returns the value of the requested service parameter.
```

More functions and structures in the file **/opt/glite/include/ServiceDiscovery.h**

gLite Service Discovery API (5)
A sample C program

```
#include "ServiceDiscovery.h"

int main(int argc, char *argv[])
{
        SDService *service;

        SDException exc;

        service = SD_getService(argv[1], &exc);

        if(exc.status != SDStatus_SUCCESS) {
                error(exc.reason);
                SD_freeService(service);
                SD_freeException(&exc);
        return -1;
        } else {
                 printf("Name: %s\n", service->name);
                 printf("Type: %s\n", service->type);
                 printf("Endpoint: %s\n", service->endpoint);
                 printf("Version: %s\n", service->version);
         return 0;
        }
}
```

WMProxy service

- The **WMProxy** (Workload Manager Proxy) is a simple service providing access to the **WMS** (Workload Management System) functionality through a **Web Services** based interface. WMProxy accepts job submission requests described with the **JDL** and other job management and control requests such as job cancellation, job file perusal, job output retrieval etc.

- This service provides additional functionality such as bulk submission and support for shared and compressed **sandboxes**.

- The WMProxy can be either accessed directly through the published **WSDL** or through the provided client tools that are:
  - a C++ command line interface
  - an **API** providing C++, Java and Python bindings.

WMProxy C++ API Overview

- **This User Interface API supplies the client applications with a set of functions providing an easy access to the WMProxy Web Services.**

- **The users are allowed:**
  - delegating the **credential** ;
  - registering and submitting jobs ;
  - cancelling the job during its life-cycle ;
  - retrieving information on the location where the job input sandbox files can be stored ;
  - retrieving the output sandbox files list ;
  - retrieving a list of possible matching Computer Elements ;
  - getting **JDL** templates ;
  - getting information on the user disk quota on the server .

WMProxy C++ API Overview (contd.)

- **Job requirements are expressed by Job Description Language (JDL). The types of jobs supported by the WM service are:**
  - **Normal** - a simple application
  - **DAG** - a direct acyclic graph of dependent jobs
  - **Collection** - a set of independent jobs
  - **Parametric** - jobs with JDL's containing some parameters

- **The API is divided into two groups of functions:**
  - **`wmproxyapi`** : with the functions that allow calling the server and handle possible fault exceptions;
  - **`wmproxyapiutils`**: with some utility functions that allow handling the **X.509 user proxy** files needed by some functions in **WMProxy API**.

WMProxy C++ API prerequisites

- **Package:**

  glite-wms-wmproxy-api-cpp-*x.x.x*.i386.rpm

  *(**x.x.x** means the recent version)*

- **Include files:**

  $GLITE_LOCATION/include/glite/wms/wmproxyapi/wmproxy_api_utilities.h

  $GLITE_LOCATION/include/glite/wms/wmproxyapi/wmproxy_api.h

- **Libraries:**

  $GLITE_LOCATION/lib/libglite_wms_wmproxy_api_cpp.so.0.0.0

  $GLITE_LOCATION/lib/libglite_wms_wmproxy_api_cpp.so.0

  $GLITE_LOCATION/lib/libglite_wms_wmproxy_api_cpp.a

  $GLITE_LOCATION/lib/libglite_wms_wmproxy_api_cpp.so

WMProxy C++ API details

WMProxy C++ API namespaces:

- **glite**
- **glite::wms**
- **glite::wms::wmproxyapi**
- **glite::wms::wmproxyapiutils**

Class hierarchy:

- **BaseException**
  - AuthenticationException
  - AuthorizationException
  - GenericException
  - GetQuotaManagementException
  - GrstDelegationException
  - InvalidArgumentException
  - JobUnknownException
  - NoSuitableResourcesException
  - OperationNotAllowedException
  - ProxyFileException
- **ConfigContext**
- **JobIdApi**
- **NodeStruct**

WMProxy C++ API Class list

- **`AuthenticationException`** Generic Authentication problem
- **`AuthorizationException`** Client is not authorized to perform the required operation
- **`BaseException`** Base exception wrap
- **`ConfigContext`** Used to configure non-default properties
- **`GenericException`** Generic problem
- **`GetQuotaManagementException`** Quota management is not active on the **WM**
- **`GrstDelegationException`** Error during delegation operations with **Gridsite** methods (grstXXXX) - since 1.2.0
- **`InvalidArgumentException`** One or more of the given input parameters is not valid
- **`JobIdApi`** Used to define the jobid hierarchy of a job or a dag
- **`JobUnknownException`** The provided job has not been registered to the system
- **`NodeStruct`** Used to define the structure of a **DAG**
- **`NoSuitableResourcesException`** No resources matching job requirements have been found
- **`OperationNotAllowedException`** Current job status does not allow requested operation
- **`ProxyFileException`** **Proxy** file errors

WMProxy CLI tools

- **glite-wms-job-delegate-proxy -** delegating a user proxy to the WMProxy service.
- **glite-wms-job-status –** retrieving the current job status
- **glite-wms-job-perusal -** allows handling files perusal functionalities for a submitted job
- **glite-wms-job-output -** to retrieve the output files of a job that has been submitted through the **glite-wms-job-submit** command with a job description file including the OutputSandbox attribute.
- **glite-wms-job-list-match -** displays the list of identifiers of the resources on which the user is authorized and satisfying the job
- requirements
- **glite-wms-job-submit -** submitting simple jobs
- **glite-wms-job-cancel –** cancel the submitted job
- **glite-wms-job-info -** retrieving useful information about the user delegated proxy,
- **glite-wms-job-logging-info** - retrieve the history of a job
- **glite-wms-get-configuration**
- **glite-wms-quota-adjust**
- **glite-wms-job-attach**
- **glite-wms-job-get-chkpt**

WMProxy Java API prerequisites

- **The following packages must be installed on the UI:**
  - glite-wms-wmproxy-api-java-***x.x.x***.rpm
  - glite-wms-ui-api-java-***x.x.x***.i386.rpm
  - glite-jdl-api-java-***x.x.x***.i386.rpm
  - glite-security-util-java-***x.x.x***.rpm
  - glite-security-trustmanager-***x.x.x***.rpm
  - glite-security-delegation-java-***x.x.x***.rpm

    *(**x.x.x** means the recent version of the corresponding package)*

    These packages can be obtained from

    `http://lxb2071.cern.ch:8080/etics/index.jsp`

Using WMProxy Java API

The main Java class is: `org.glite.wms.wmproxy.WMProxyAPI`

A client object can be created using one of these 4 constructors:

- **`public WMProxyAPI(String url, String proxyFile)`** where:
  url: the WMProxy server URL to be contacted (e.g. https://<host>:<port>/glite_wms_wmproxy_server);
  proxyFile: the pathname to a valid user proxy; for the default value set this to NULL;

- **`public WMProxyAPI(String url, String proxyFile, String certsPath)`**
  it is used only if the pathname to the local **CA** directory (***certsPath***) is different from the Linux default one (/etc/grid-security/certificates).

- **`public WMProxyAPI(String url, InputStream proxyStream)`** where:
  url: the WMProxy server URL (e.g. https://<host>:<port>/glite_wms_wmproxy_server);
  proxyStream: a valid proxy passed as an input stream;

- **`public WMProxyAPI(String url, InputStream proxyStream, String certsPath)`**
  it is used only if the pathname to the local CA directory (***certsPath***) is different from the Linux default one (/etc/grid-security/certificates).

Using WMProxy Java API (contd.)

Example:

```
WMProxyAPI client =
   new WMProxyAPI("https://trinity.datamat.it:7443/glite_wms_wmproxy_server",
                  "/tmp/x509up_u500");

string version = getVersion(cfg);
```

In case of failure, one of the following exceptions is thrown:

- `org.glite.wms.wmproxy.AuthenticationFaultType`
  (for generic authentication problems);
- `org.glite.wms.wmproxy.AuthorizationFaultType`
  (when the client is not authorized to perform the required operation);
- `org.glite.wms.wmproxy.JobUnknownFaultType`
  (when the identifier of the job provided as input parameter does not correspond to any job that has registered to the WMProxy);
- etc.

Using WMProxy Java API (contd.)

Delegation of user credential

- Delegation process is needed to transfer client proxy credentials to the server host. Delegated credentials are uniquely identified by the association of the delegation identifier, provided by user, and the **user's DN** within the credentials.
- **First of all, define a delegation identifier:**

```
string delegationId = "myId";
```

- **Request a certificate which includes a public key to the server:**

```
String delegationId = "myId";
String proxy = client.grstGetProxyReq(delegationId);
```

- **Send the signed proxy certificate to the server:**

```
client.grstPutProxy(delegationId, proxy);
```

Using WMProxy Java API (contd.)

Job submission

- If the job does not have any file in the InputSandbox to be transferred from the submitting machine to the WMProxy node, the submission can be performed by calling the **jobSubmit** service:

```
jobIds = client.jobSubmit(jdlString, delegationId);
```

- **Otherwise, these following steps are needed:**
  - A preliminary server registration:
    ```
    jobIds = client.jobRegister(jdlString, delegationId);
    ```
  - Transfer of files in the InputSandbox from the client machine to the WMProxy node;
  - Call the jobStart service to trigger the submission:
    ```
    client.jobStart(id);
    ```
    where "id" is the identifier of the job to be retrieved by the struct that the jobRegister service returns.

Using WMProxy Java API (contd.)

WMProxy versions

- **Different versions could have:**
  - different services!
  - services with different input data!
  - services that return different output data!

- **The WMProxy version is identified with 3 numbers:** `<MajorVersion>.<MinorVersion>.<SubMinorVersion>`.

- **The numbers of the WMProxy node that is being contacted can be retrieved calling the `getVersion` service:**

```
WMProxyAPI client =  new WMProxyAPI("https://trinity.datamat.it:7443/glite_wms_wmproxy_server",
                                    "/tmp/x509up_u8930");
string version = getVersion(cfg);
```

Using WMProxy Java API (contd.)

Transfer Protocols

- Since the WMProxy version 2.2.0 is available the `getTransferProtocols` service. It returns the list of File Transfer Protocols supported by the WMProxy server used for the job submission.

```
org.glite.wms.wmproxy.StringList list =
        client.getTransferProtocols();

String[] protoList = list.getItem();
```

Using WMProxy Java API (contd.)

Destination URI locations

- **The WMProxy provides the URI of each location (destination URIs) where the job input sandbox files can be stored but does not perform any file transfer. Users have to transfer the files located on the client machines to the related WMProxy URI locations. These URI locations are retrieved with the following services:**

  `getSandboxDestURI` *(for a single node)*;

  `getSandboxBulkDestURI` *(for a job with N>1 nodes)*.

- *(The input parameter of these two services are different on the basis of the WMProxy version!)*

Using WMProxy Java API (contd.)

`getSandboxDestURI` service

- **The WMProxy getSandboxDestURI service is used in case of simple jobs and compound jobs that do not have any ISB file to be transferred for the children node.**
- **For WMProxy servers with version earlier than 2.2.0:**

```
org.glite.wms.wmproxy.StringList list =
    client.getSandboxDestURI(jobid);
String[] uriList = list.getItem();
```

- The service returns a list of URIs: one for each File Transfer Protocol supported by the server (i.e. https, gsiftp). Files of the job input sandbox that have been referenced in the JDL as relative or absolute paths are expected to be found in the returned location when the job lands on the CE.

Using WMProxy Java API (contd.)

`getSandboxBulkDestURI` service

- **In case of compound jobs with files that need to be uploaded to different URI locations, the list of URIs for all nodes is retrieved calling this service.**
- **For WMProxy servers with version earlier than 2.2.0:**

```
org.glite.wms.wmproxy.StringList list =
    client.getSandboxBulkDestURI(jobid);
String[] uriList = list.getItem();
```

- The returned vector contains a list of pairs:
  the **jobid** string that identifies a single node;
- a vector with a destination **URI** for each available protocol

WMProxy additional info

JDL attributes specification for gLite's WMProxy service:

https://edms.cern.ch/file/590869/1/EGEE-JRA1-TEC-590869-JDL-Attributes-v0-9.pdf

- This document provides the specification of the **Job Description Language** attributes supported by the gLite software (version 3.0.1 or later). Attributes and features described in this document are fully supported only if the job submission to WMS is performed through the **WMProxy**, i.e. the **Web services** based interface to the gLite **Workload Management System**.

Using LFC CLI

- **Summary of some LFC commands：**

`lfc-chmod`	Change access mode of the LFC file/directory
`lfc-chown`	Change owner and group of the LFC file-directory
`lfc-delcomment`	Delete the comment associated with the file/directory
`lfc-getacl`	Get file/directory access control lists
`lfc-ln`	Make a symbolic link to a file/directory
`lfc-ls`	List file/directory entries in a directory
`lfc-mkdir`	Create a directory
`lfc-rename`	Rename a file/directory
`lfc-rm`	Remove a file/directory
`lfc-setacl`	Set file/directory access control lists
`lfc-setcomment`	Add/replace a comment

Using LFC Python API

- **List of all replicas of a file with a given LFN *logical_filename*:**

```
#!/usr/bin/python

import sys
import lfc

file_name = "logical_filename"

result, list = lfc.lfc_getreplica(file_name, "", "")

print "List is "
if (result == 0):
    print "length of list is " + str(len(list))
    print "Replicas list of " + file_name + " is:"
    for i in list:
        print i.host
        print i.sfn
```

Additional info: https://twiki.cern.ch/twiki/bin/view/LCG/LfcApi

Conclusion

Except the **gLite APIs**, a successful **Grid application developer** must have knowledge and skills in the following areas:

- Modern Grid technologies, especially those used in the EGEE project (gLite)
- gLite command line tools
- Distributed and parallel programming techniques, MPI ...
- Experience in one or more of the following high-level programming languages: C, C++, Java and FORTRAN (*Other languages?*)
- Job Description Language (JDL)
- UNIX Shell scripting and/or other scripting languages (Python, Perl ...)
- Web services and related protocols
- Computer and network security
- Dealing with huge data arrays using Storage Resource Management
- Dealing with complex legacy applications
- Software development process

Firther reading

Silva, V. “Grid Computing For Developers”, 2005