Enabling Grids for E-science in Europe

www.eu-egee.org

LCG-2 Tutorial
CERN (Geneva), 29-30 November 2004

Grid Information System: Hands-on session

Patricia Méndez Lorenzo
patricia.mendez@cern.ch
LCG Experiment Integration and Support
CERN IT/GD-EIS

EGEE is a project funded by the European Union under contract IST-2003-508833

Contents

Enabling Grids for E-science in Europe

- Exercises
- lcg-is-search code
  - InfoFromLDAP class
  - dlopen

  } Based on LDAP Protocol
- lcg-is-search-rgma code
  - InfoFromRGMA class
  - dlopen

  } Based on R-GMA Protocol
- lcg-is-add-rgma code
  - InfoToRGMA class
  - dlopen

  } Based on R-GMA Protocol
- lcg-InfoInterface code

  } LDAP + R-GMA + …

Hands-on session

Enabling Grids for E-science in Europe

The Hands-on session includes two types of exercises:

1. Those which will be just shown because they require sgm (lcg-ManageVOTag) or root (lcg-user-configuration) privileges

2. The following needs your work
   - C++ APIs and Perl scripts
   - You just have to work with the C++ APIs and we provide you with the needed Makefile, libraries and headers
   - **Just concentrate on the C++ applications**
   - The Perl scripts are lcg-utilities which use the C++ APIs. Use them to get familiar with the lcg-utilities

3. General Remarks:
   - Work in couples, it will be easier
   - Do not hesitate to ask questions and have a look at the solutions each time you get stack
   - Ask your tutors in case of problems

Generalities for all APIs

Enabling Grids for E-science in Europe

Where to find the sources?

Headers: `/opt/lcg/include`
Libraries: `/opt/lcg/lib`
Executables: `/opt/lcg/bin`
At your home directory you have already installed : `IS_exercises/ldap`
`/rgma`

1. **Makefiles:**
   - `ldap/Makefiles/Makefile_search_ldap`
   - `ldap/Makefiles/Makefile_general_ldap`
   - `rgma/Makefiles/Makefile_search_rgma`
   - `rgma/Makefiles/Makefile_add_rgma`
2. **Templates:**
   - `ldap/Templates/Template_search_ldap.cpp`
   - `ldap/Templates/Template_general_ldap.cpp`
   - `rgma/Templates/Template_search_rgma.cpp`
   - `rgma/Templates/Template_add_rgma.cpp`
3. **Solutions:**
   - `ldap/Solutions/Solution_search_ldap.cpp`
   - `ldap/Solutions/Solution_general_ldap.cpp`
   - `rgma/Solutions/Solution_search_rgma.cpp`
   - `rgma/Solutions/Solution_add_rgma.cpp`

LDAP Exercises

Enabling Grids for E-science in Europe

1. Use the liblcg-info-api-ldap library

We propose you to generate a main program which:

¤ Write an application that requires the following arguments: host, port, filter, attribute(s). It loads dynamically this library (dlopen). Then it invokes the "query" method of InfoFromLDAP.h and prints out the resultof the user query on the screen.

¤ The query method definition is in InfoFromLDAP.h

¤ Have a look at the lcg-infosites script. It uses the executable generated by the solution provided.

¤ If you have time copy this script (placed in /opt/lcg/bin) in a local area and replace the lcg-is-search executable with your executable and try to run it

¤ Compare results with those obtained from lcg-infosites

LDAP Exercises

Enabling Grids for E-science in Europe

2. Use the liblcg-info-search-rgma library

Try to generate a main program which:

1. Passes a file including the query (written in SQL)
2. Makes a dynamic load of this library (dlopen)
3. Invokes the general query method of the InfoFromRGMA.h

Remarks:

¤ The Solutions directory includes some examples of SQL queries

LDAP Exercises

Enabling Grids for E-science in Europe

3. Use the liblcg-info-add-rgma library

You will have to generate a main program which:

1. Passes a file including the request (written in SQL)
2. Makes a dynamical download of this library (dlopen)
3. Invokes the general query method of the InfoToRGMA.h

Remarks:

¤ The Solutions directory includes some examples of SQL queries

To begin with

Enabling Grids for E-science in Europe

If you are not very familiar with the CLI ldapsearch and with the command lcg-infosites, just play a bit before doing APIs

```
% ldapsearch -x -LLL -h grid017.ct.infn.it -p
2170 -b "o=grid"
```

```
% ldapsearch -x -LLL -h grid017.ct.infn.it -p
2170 -b "o=grid" '(objectclass=GlueSE)'
GlueSEName GlueSEPort
```

To begin with

Enabling Grids for E-science in Europe

You can try the same queries you made with ldapsearch:

```
% /opt/lcg/bin/lcg-is-search -f objectclass=GlueSE -a GlueSEName GlueSEPort
```

- You do not have additional information you did not ask for (the DNs)
- The lines are not cut at the end

Compare with ldapsearch

```
lcg-is-search -f objectclass=GlueTop -a'(& (GlueServiceType=edg-local-replica-
catalog) (GlueServiceAccessControlRule))' GlueServiceAccessPointURL
```

First of all you do not care about hosts or ports. Just in the case you want an specific host, otherwise lcg-is-search looks at the one in default

```
ldapsearch -h grid017.ct.infn.it -p 2170 -x -LLL -b "o=grid"
'(objectclass=GlueTop)' '(& (GlueServiceType=edg-local-replica-catalog)
(GlueServiceAccessControlRule))' GlueServiceAccessPointURL
```

- You do not ask for the DN
- The lines are cut at the end of the buffer. It's very difficult to wrap this information into your code

To begin with

Enabling Grids for E-science in Europe

Test some lcg-infosites features:

```
%lcg-infosites -vo gilda ce

%lcg-infosites -vo gilda se

%lcg-infosites -vo gilda all

%lcg-infosites -vo gilda lrc

%lcg-infosites -vo gilda rmc
```

CODE

Enabling Grids for E-science in Europe

The following slides contain the code of the APIs and their implementations

Information System Tools

Enabling Grids for E-science in Europe

```
> lcg-is-search -h <host> -f objectclass=<your_request> -a \
'<your_attributes>'
```

lcg-is-search basic Code

```
#include<dlfcn.h>
#include<iostream> (<strtream>)
#include<vector> (<iterator>, <string>)

#include <lcg-info-api-ldap/InfoFromLDAP.h>
#include<lcg-info-api-ldap/AllInfoLDAP.h>

int main (int argc, char*argv[]){
char *hosttest;
char* first_ptr;
char* last_ptr;
hosttest = getenv("LCG_GFAL_INFOSYS");
first_ptr = hosttest;
last_ptr = strchr(hostest,":");
*last_ptr = '\0';
++last_ptr;
std::string host(first_ptr);
typedef enum {_param_,_host_,_port_,_filter_,_attr_} arg_t;
arg_t expected = _param_;
```

to include DLOPEN

including classes of the package

If not specified, the host will be taken from LCG_GFAL_INFOSYS

including external arguments

Information System Tools

Enabling Grids for E-science in Europe

lcg-is-search basic Code (cont.)

```
for (int i = 1; i<argc: i++){
   string token;
   bool read_token = true;
   istream* in = new istrstream(argv[i]);
   switch (expected){
   case _port_: (*in) >> port; expected = _param_;break;
   case _host_: (*in) >> host; expected = _host_;break;
   case _filter_: (*in) >> filter; expected = _filter_;break;
   case _attr_: (*in) >> attribute;
   if (attribute[0] !='-'){
      attributes.push_back(attribute.c_str());
      break;
   }
   else {
      token = attribute.c_str();
      read_token = false;
   }
     default:
       if(read_token) (*in) >> token;
       if(token =="-p") expected = _port_;
       else if(token == "-h") expected = _host_;
       else if(token == "-f") expected = _filter_;
       else if(token == "-a") expected = _attr_;
```

Part of the code
To include external arguments

EIS
experiment integration and support

Information System Tools

Enabling Grids for E-science in Europe

lcg-is-search basic Code (cont.)

```
  else {
     cout<< "invalid parameter" <<token<<endl;
   }
 }
 delete in;

#ifndef __WINDOWS__;
char* lib_loc = "liblcg-info-api-ldap.so";

void *InfoFromLDAP = dlopen(lib_loc,RTLD_LAZY);

create_t* create_infoldap = (create_t*)dlsym(InfoFromLDAP,"create");
destroy_t* destroy_infoldap = (destroy_t*)dlsym(InfoFromLDAP,"destroy");

AllInfoFromLDAP *ldapinfo = create_infoldap();
ldapinfo-> query(host,filter,attributes,port);

destroy_infoldap(ldapinfo);
dlclose(InfoFromLDAP);
#endif
}
```

including the library to load

loading dynamically the library

instantiating the class

calling its method

destroying the pointer

Information System Tools

Enabling Grids for E-science in Europe

InfoFromLDAP.h

```
#include "AllInfoLDAP.h"
#include "LDAPConnection.h"
```
← one of the LDAP wrappers

```
class InfoFromLDAP: public AllInfoLDAP{
```
← price to pay to include the dlopen package

```
Public:
   InfoFromLDAP();
   ~InfoFromLDAP();

   virtual void query(string, string, vector<string>, int);
```
the method of the class

```
Private:
   LDAPConnection* connection;
```
← pointer to connect the server

```
}
```

AllInfoLDAP.h

```
class AllInfoLDAP{
public:
virtual void query(string, string, vector<string>,int) = 0;
};
typedef AllInfoLDAP* create_t();
typedef void destroy_t(AllInfoLDAP*);
```
← mandatory because of dlopen

Information System Tools

Enabling Grids for E-science in Europe

InfoFromLDAP.cpp

```
#include "LDAPQuery.h"
#include "LDAPSynchConnection.h"
#include "LDAPForwardIterator.h"
#include "InfoFromLDAP.h"

InfoFromLDAP::InfoFromLDAP(){};
InfoFromLDAP::~InfoFromLDAP(){};

void InfoFromLDAP::query(string host, string filter, vector<string> attributes,
     int port){
string information_index = "o=grid";
int timeout = 30;

connection = new LDAPSynchConnection (information_index,host,port,timeout);
copy (attributes.begin(),attributes.end(),ostream_iterator<string>(cout," "));

LDAPQuery query(connection,filter,attributes);

Connection -> open();
Query.execute();

LDAPForwardIterator ldap_it(query.tuples());
```

wrappers of LDAP

instantiating a synch connection

informing of the query to perform

connecting with the server

executing the query

iterating the IS buffer

Information System Tools

Enabling Grids for E-science in Europe

InfoFromLDAP.cpp (cont)

```
ldap_it.first();
while (ldap_it.current() ){
  cout<< (*ldap_it) << endl;
  ldap_it.next();
}

connection->close();
};

extern "C" AllInfoLDAP* create();{
   return new InfoFromLDAP;
}
extern "C" void destroy(AllInfoLDAP*a){
  delete a;
}
```

results printed through the screen

looping through the buffer

closing the connection

price to pay because dlopen

It seems dlopen is quite difficult to use (additional classes and code) but has fundamental advantages

dlopen

Enabling Grids for E-science in Europe

In some situations it can be very useful to load a certain library at runtime

►In many cases you want your code to support multiple technologies: creation of plug-ins

►You want to make your code independent on underlying changes

- **The Solution:**

Plug-ins usage: load the library dynamically at runtime only when needed.

- **But….**

It is quite easy to do in in C but not so easy in C++:

→ Because of name mangling

→ Because you have to expose the symbols of the whole class in C++

- **Solution:**
- Extern “C” (for the name mangling)
- Polymorphism (for the classes)

dlopen in our code

Enabling Grids for E-science in Europe

♠ In our code we want to load a class into the main (`lcg-is-search`); `InfoFromLDAP` to use its method query

♠ We cannot use `"new"` to instantiate the class

Solution:

1. We define a base class: `AllInfoLDAP.h` (pure virtual) and `InfoFromLDAP` will be derived from it (called module)

AllInfoLDAP.h

```
// the types of the class factory
typedef AllInfoLDAP* create_t();
typedef void destroy_t (AllInfoLDAP*)
```

2. Inside the module two helper functions (class factory functions) will be defined as extern "C"

InfoFromLDAP.cpp

```
// the class factories
extern "C" AllInfoLDAP* create() {result new InfoFromLDAP;}
extern "C" void destroy(AllInfoLDAP* a) {delete a;}
```

dlopen in our code

Enabling Grids for E-science in Europe

Finally in the code **lcg_is_search.cpp**

```
void *InfoFromLDAP = dlopen("your lib",RTLD_LAZY);

create_t* create_infoldap = (create_t*)   \
      dlsym(InfoFromLDAP,"create");
destroy_t* destroy_infoldap = (destroy_t*)   \
      dlsym(InfoFromLDAP,"destroy");

AllInfoLDAP* ldapinfo = create_infoldap();
ldapinfo ->query;

destroy_infoldap(ldapinfo);
dlclose(InfoFromLDAP)
```

implementation

Seems similar to new…

Using the method

Seems similar to delete…

LCG APIS from R-GMA

Enabling Grids for E-science in Europe

♠ **InfoFromRGMA:** Parallel development to `InfoFromLDAP`

```
> lcg-is-search-rgma  <your_request>
```

experiment integration and support

InfoFromRGMA.h

```
#include "AllInfoRGMA.h"

class InfoFromRGMA: public AllInfo{
public:
InfoFromRGMA();
~InfoFromRGMA();
virtual void query(char*);
}
```

LCG APIs from R-GMA

Enabling Grids for E-science in Europe

InfoFromRGMA.cpp

```
#include "Consumer.hh"
#include "ResultSet.hh"
#include "InfoFromRGMA.h"

void InfoFromRGMA::query(char* file){

char buff[1024];
std::ifstream sqlFile(file,std::ios::in);
std::ostringstream os;
while (!sqlFile.getline(buff,sizeof(buff)).eof() ){
os << buff << ' ';
}
sqlFile.close();

edg::info::Consumer myConsumer(os.str(),edg::info::Consumer::
              CONTINUOUS);
```

reading the file

Constructing a consumer

LCG APIs from R-GMA

Enabling Grids for E-science in Europe

InfoFromRGMA.cpp

```
edg::info::TimeInterval Timeout(60);
myConsumer.start(Timeout);          initiate streaming with each Producer
while(myConsumer.isExecuting()){
sleep(2);
}
edg::info::ResultSet resultSet = myConsumer.popIfPossible();
std::cout<<ResultSet:\n"<<resultSet.toString().c_str()<<std::endl;
myConsumer.close();   getting results and printing them by screen
};

extern "C" AllInfoRGMA create(){ return new InfoFromRGMA;}

extern "C" void destroy(AllInfoRGMA* a){ delete a;}
```

dlopen

LCG APIS from R-GMA

Enabling Grids for E-science in Europe

♠ **InfoToRGMA:**

You have the power, You create the information

```
> lcg-is-add-rgma  <your_file>
```

experiment integration and support

InfoToRGMA.h

```
#include "AllInfoRGMA.h"

class InfoToRGMA: public AllInfo{
public:
InfoToRGMA();
~InfoToRGMA();
virtual void add(char*);
}
```

LCG APIS from R-GMA

In this package a configuration file should be included with the following data:

1. The name of the table where your info is included
2. Your information

Example of Configuration File

```
theTABLE = userTable

theREQUEST = INSERT INTO userTable (userID, aString,
anInt, MeasurementDate,MeasurementTime) VALUES
('test','producertest',5.18,32,'2004-10-19','18:59:00')
```

LCG APIs from R-GMA

Enabling Grids for E-science in Europe

InfoToRGMA.cpp

```
#include "StreamProducer.hh"
#include "ConfigBuffer.hh"
#include "InfoToRGMA.h"

void InfoFromRGMA::add(char* file){

string thefile = file;
configBuffer *theconfigfile = new ConfigBuffer(thefile);
std::string table = theconfigfile->get_attribute_value("theTABLE");
std::string request = theconfigfile->get_attribute_value("theREQUEST");

edg::info::StreamProducer myProducer;
myProducer.declareTable(table,"");
myProducer.setTerminationInterval(edg::info::TimeInterval(1200));
myProducer.setMinRetentionPeriod(edg::info::TimeInterval(600));
myProducer.insert(request);
```

General Interface Tool

Enabling Grids for E-science in Europe

Your user Application can look like as:

```
#include ''LcgInfoInterface.h''
vector <vector<string> > results;
string input;
LcgInfoInterface iface;
iface.initialize(''config_file'');
Querier* thequerier = iface.connect();
input = ''query performed by the user'';
results = thequerier ->query(input);
iface.disconnect(thequerier);
```

contains the results of the query

written in SQL

read the configuration file

dynamical load of the protocol libraries

the query is performed

the final disconnection

http://grid-deployment.web.cern.ch/grid-deployment/eis/docs/LcgInfoInterface/namespaces.html
http://grid-deployment.web.cern.ch/grid-deployment/eis/docs/LcgInfoInterface/LcgInfoInterface_refman.pdf