Table of Contents

Task Force Data Clients Consolidation: Evaluation of Existing Clients

Authors

Introduction

This document is an evaluation of the existing clients for ARC and gLite in the data area.

Consolidation ideas

Ideas for consolidation based on the comparison table below:

- Both components use lfc-tools for catalog handling, lcg_utils discards catalog handling commands
- gLite tries to create an ARC plugin to mate it possible to use SRM code from ARC
- ARC tries to create a GFAL plugin to make it possible to use the protocols supported by GFAL

Comparison table

End-user clients		
gLite	**ARC**	**Comments**
lcg-cp file:// ...	arccp	
lcg-cr	arccp lfc://srm/lfc/[metadata] (inc. space tokens)	
lcg-del	arcrm lfc://srm/lfc/[metadata]	
lcg-rep	arccp lfc://srm/lfc/[metadata]	
lcg-gt (getturls) proto	-	
lcg-sd (set done)	-	
Functional differences		
Infosys interaction (site name / VO / host to end-point resolution)	-	Reuse gLite or ARC job libraries?
Specifying transfer protocol (by end-user)	Configurable through URL option transferprotocol	-
Automatic SURL generation	-	want to have this
- (uses protocol)	chksum on the fly (by internal buffer)	
Libraries		
GFAL2	**ARC**	**Comments**
Protocols		
GridFTP	GridFTP	
SRM	SRM	
LFC	LFC	
file	file	
rfio		
GSIdCAP		
	xrootd	xrootd not in prod. release yet
	HTTPS	
	ARC	

	stdio	
	API	
POSIX	Functionality based	Limited overlap between gLite and ARC APIs
	PrepareReading, PrepareWriting	
	StartReading, StartWriting	
	StopReading, StopWriting	
	FinishReading, FinishWriting	
	Check	
	Remove	
gfal_stat	Stat	
	List	
	Methods for locations and registration	
	Helper methods	
	Set/Get methods	
gfal_chmod		
gfal_rename		
gfal_lstat		
gfal_access		
gfal_readlink		
gfal_creat		
gfal_open		
gfal_lseek		
gfal_close		
gfal_read		
gfal_write		
gfal_getxattr		
gfal_listxattr		
gfal_mkdir		
gfal_opendir		
gfal_closedir		
gfal_readdir		
gfal_rmdir		
gfal_unlink		
gfal_posix_clear_error		
gfal_posix_release_error		
gfal_posix_strerror_r		
gfal_posix_print_error		
gfal_posix_code_error		
gfal_set_verbose		
gfal_set_vo		
gfal_set_nobdii		
gfal_set_timeout_connect		
gfal_get_timeout_connect		
gfal_set_timeout_sendreceive		
gfal_get_timeout_sendreceive		
gfal_set_timeout_bdii		
gfal_get_timeout_bdii		
gfal_set_timeout_srm		

gfal_get_timeout_srm		
gfal_is_nobdii		

Middleware Clients

The middlewares mentioned in the following sections are ordered alphabetically, the order does not express any preferences.

ARC

Classes in ARC

The ARC middleware builds on several sets of classes both utilized by clients and services. These sets cover common, data handling, message handling/communication, module loading, security, info system classes among others. The data, message and security handler classes are general classes, which is extended by specific modules, e.g. for data handling SRM, gsiftp, http, etc., for message handling tls, http, gsi, etc., and for security handling x509, usernametoken etc. These modules are pluggable and can be used on the server as well as on the client side.

ng* data clients

These clients will be phased out in the near future.

arc* data clients and libarcdata2

The libarcdata2 library and the corresponding arc* data clients provides a uniform way to move data from point to point supporting different data transfer protocols. This is a successor of the previous ng* data clients which are still widely used and will be maintained.

The libarcdata2 library has a modular structure to support different data transfer protocols. The core libarcdata2 library does not introduce any additional external dependencies. The plugins (DMCs) for specific data access protocols can however have various external dependencies. This separation of external dependencies from the core library helps reduce the minimum set of requirements for ARC while allowing the support for additional access protocols requiring special dependencies to be installed by those who need it.

Most of these components and the clients are also available on different platforms (Linux, Windows, Mac, Solaris), and the libarcdata2 is also available from Python, and to some extent from Java.

Currently the following transfer protocols and metadata servers are supported:
[http://www.nordugrid.org/documents/arc-ui.pdf]

- ftp: ordinary File Transfer Protocol (FTP)
- gsiftp: GridFTP, the Globus-enhanced FTP protocol with security, encryption, etc.
- http: ordinary Hyper-Text Transfer Protocol (HTTP) with PUT and GET methods using multiple streams
- https: HTTP with SSL v3
- httpg: HTTP with Globus GSI
- ldap: ordinary Lightweight Data Access Protocol (LDAP)
- rls: Globus Replica Location Service (RLS)
- lfc: LFC catalog and indexing service of EGEE gLite
- srm: Storage Resource Manager (SRM) service
- file: local to the host file name with a full path

- arc: for the Chelonia storage service

The arc* data clients are the following:

- arcls is a simple utility that allows to list contents and view some attributes of objects of a specified (by a URL) remote directory.
- arccp is a powerful tool to copy files from a data point to on other data point where both can use different protocols.
- arcrm is a command allows users to erase files at any location specified by a valid URL.

Advantages of using the ARC Data libraries

- It can be used on different platforms (Linux, Windows, Mac Solaris).
- It can be used from different languages (C++, python, java).
- It has a pluggable infrastructure which allows for a very limited set of dependencies from other libraries.
- Plugins exist for several widely used data transfer and management protocols.
- It is light-weight and user-friendly, with easy client-side configuration (the default config would work in most cases).

gLite

The gLite software stack provides both a POSIX-like API for file access (gFAL) and a set of command line tools and higher level APIs (lcg_util).

Both rely on a common set of protocols, widely supported by services on the grid:

Access	gsiftp, rfio, gsidcap, file
Management	SRM
Catalogs	LFC, RLS

The libraries are extensible and have evolved along with grid development, so adding support for additional protocols should not pose significant problems.

In addition to the protocol abstraction, file reference and interaction is also transparent to the user. Both libraries understand LFNs, GUIDs, SURLs and TURLs, and know how to use the grid information system to select the services to contact to convert between them as appropriate.

lcg_util

The lcg_util package provides both a set of command line tools and equivalent APIs for C/C++, Python and Perl.

In most cases there is a one-to-one mapping between the CLI tool and the API function.

Check here for the full list of available commands and man pages.

Replica Management

lcg-cp	Copies a file to/from a SE
lcg-cr	Copies a file to a SE and registers the file in the catalogue
lcg-del	Deletes one file (either one replica or all replicas)
lcg-rep	Copies a file from one SE to another SE and registers it in the catalogue (replicate)
lcg-gt	Gets the TURL for a given SURL and transfer protocol

lcg-sd	Sets file status to "Done" for a given SURL in an SRM's request

Catalog Interaction

lcg-aa	Adds an alias in the catalogue for a given GUID
lcg-ra	Removes an alias in the catalogue for a given GUID
lcg-rf	Registers in the catalogue a file residing on an SE
lcg-uf	Unregisters in the the catalogue a file residing on an SE
lcg-la	Lists the aliases for a given LFN, GUID or SURL
lcg-lg	Gets the GUID for a given LFN or SURL
lcg-lr	Lists the replicas for a given LFN, GUID or SURL
lcg-ls	Lists file information for given SURLs or LFNs

Grid File Access Library (gFAL)

The gFAL API mimics the POSIX calls, preprended with 'gfal_'. The gFAL libraries have also been ported to JAVA.

An example:

```
...
   if((FD = gfal_open ( filename, O_RDONLY,0 )) < 0) {
      perror ("error in gfal_open");
      exit(1);
   }
   cout << "File is successfully opened\n";

   if ((rc=gfal_read (FD, readValues, block_size)) != block_size ) {
      if (rc < 0) perror("error in gfal_read");
      else cerr << "gfal_read returns " << rc << endl;
   }
   cout << "File is successfully read\n";

   for(int i=0; i<array_size; i++)
      cout << "\treadValues[" << i << "] = " << readValues[i] << endl;

   if ((rc= gfal_close (FD)) < 0) {
      perror ("error in gfal_close");
      exit(1);
   }
...
```

Check here for the complete list of available functions and man pages.

Summary

-- JonKerrNilsen - 03-Nov-2010

This topic: EMI > EmiJra1DataClientEvaluation
Topic revision: r8 - 2012-01-09 - JonKerrNilsen