Journal of Physics: Conference Series **513** (2014) 032015 doi:10.1088/1742-6596/513/3/032015

# ARC SDK: A toolbox for distributed computing and data applications

**M Skou Andersen[1], D Cameron[2] and J Lindemann[3]**

[1] University of Copenhagen, NBI, Blegdamsvej 17, DK-2100 Copenhagen Ø, Denmark
[2] Department of Physics, University of Oslo, P.b. 1048 Blindern, N-0316 Oslo, Norway
[3] Lunarc, Lund University, Box 118, 221 00 Lund, Sweden

**Abstract.**

Grid middleware suites provide tools to perform the basic tasks of job submission and retrieval and data access, however these tools tend to be low-level, operating on individual jobs or files and lacking in higher-level concepts. User communities therefore generally develop their own application-layer software catering to their specific communities' needs on top of the Grid middleware. It is thus important for the Grid middleware to provide a friendly, well documented and simple to use interface for the applications to build upon. The Advanced Resource Connector (ARC), developed by NorduGrid, provides a Software Development Kit (SDK) which enables applications to use the middleware for job and data management. This paper presents the architecture and functionality of the ARC SDK along with an example graphical application developed with the SDK. The SDK consists of a set of libraries accessible through Application Programming Interfaces (API) in several languages. It contains extensive documentation and example code and is available on multiple platforms. The libraries provide generic interfaces and rely on plugins to support a given technology or protocol and this modular design makes it easy to add a new plugin if the application requires supporting additional technologies.The ARC Graphical Clients package is a graphical user interface built on top of the ARC SDK and the Qt toolkit and it is presented here as a fully functional example of an application. It provides a graphical interface to enable job submission and management at the click of a button, and allows data on any Grid storage system to be manipulated using a visual file system hierarchy, as if it were a regular file system.

## 1. Introduction

In the classical Grid architecture [1], Grid middleware sits between applications and the distributed resources that application users wish to exploit. Middleware itself typically follows a client-server model, where applications interact with local client software through an API, and these clients communicate with middleware services located at the Grid resources. The ARC middleware [2], developed by NorduGrid, contains both these components: a Computing Element, a service to connect computing resources to the Grid, and a client-side SDK, to allow application developers to interface to the Grid. This focus of this paper is the SDK. The SDK architecture is explained along with some examples of how to use it and then the ARC Graphical Clients, a graphical application built on the SDK, is presented.

## 2. SDK Architecture

The ARC SDK is essentially a set of libraries accessible through an API to develop various utilities and tools. The SDK consists of credential, compute, data, data staging and common libraries, whose API are mostly high-level interfaces. The libraries rely on various plugins dynamically loaded at runtime, making the libraries modular and easily extensible. In order to extend the SDK to support any additional technology (within the modular scope of the SDK), a new plugin can be created to provide a specialised implementation of a given generic interface in a simple manner, without recompilation of the SDK libraries. The functionalities of the various interfaces are: querying of registries and local information systems, matchmaking and ranking, task description handling, task submission, task management, and working with various transfer protocols and meta-protocols. An overview of the ARC SDK architecture showing such interfaces can be seen in Figure 1.

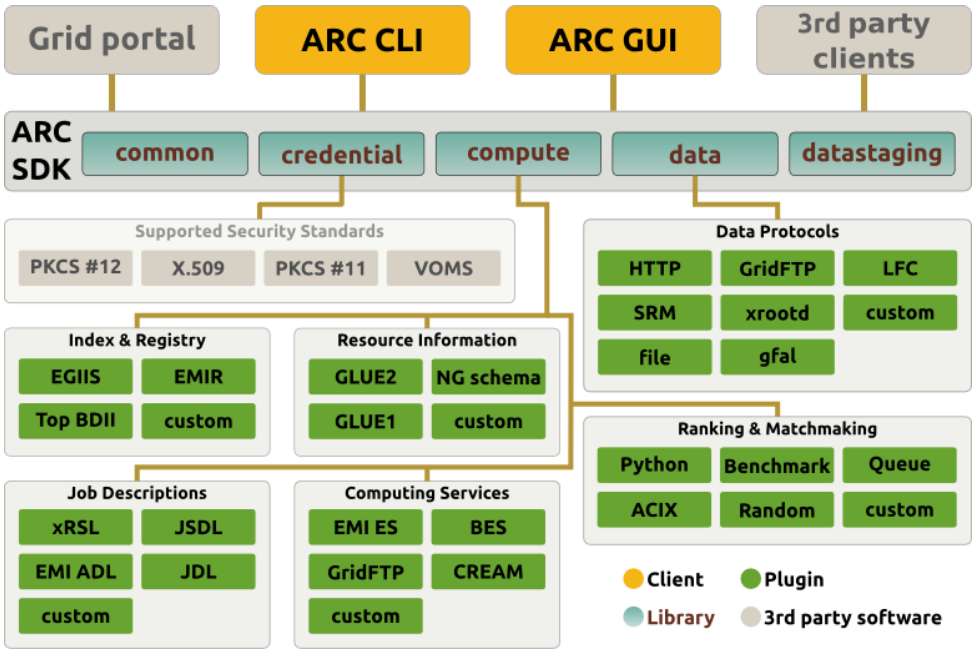


**Figure 1.** Components of the SDK, applications and plugins.

The credential library is able to handle X.509 certificates and associated keys [3], which is vital for Grid authentication and authorisation. The library supports X.509 mainly through the OpenSSL toolkit, and is able to list certificate information, create proxy certificates, create and sign certificate requests, add extensions to certificates, and verify certificate validity. Furthermore, VOMS [4] attribute certificate extensions are supported, and it is possible to create and parse such extensions.

The task of the compute library is to submit and manage Grid jobs, for which a wide range of interfaces exist, with ARC supporting some of the major ones, such as EMI ES [5], GridFTP [6], CREAM [7] and OGSA-BES [8] with or without ARC proprietary extension. When it comes to describing computational tasks, they are formalised in standard terms, i.e. a specialised language, with the ARC libraries currently supporting xRSL [9], JDL [10], EMI ADL [5], JSDL-POSIX [11] and JSDL-HPC [12] languages.

Several different Grid information systems exist, with ARC supporting ARIS [13], EMI ES resource information port-type [5] and Resource BDII [14] local information systems. EGIIS [13] and EMIR [15] information registries are also supported. Using obtained information, resources

are matched against the computational task description, and the matching resources are then ranked according to user requirement, e.g. highest clock frequency, or shortest batch queue. Several broker algorithms are included with ARC and it is possible to supply custom user-defined brokers written in Python at runtime.

Data is a vital element in scientific computing in general and Grid in particular. An abundance of data access protocols exist in Grid, and in addition there are catalogs mapping logical data identifiers to (possibly several) physical replicas. ARC supports many Grid access protocols and catalogs (e.g. GridFTP, SRM [50], HTTP(s) and LHC File Catalog). The data library is capable of carrying out basic file and directory management tasks, such as copying, removing, renaming and listing.

The datastaging library is designed for scheduling of large-scale data transfer to and from the Grid [16]. For low-level data handling it uses the data library and datastaging is used in the ARC Computing Element job execution service to manage data input and output for computational tasks.

## 3. Using the SDK

In order to exploit the functionalities provided by the SDK, either the native C++ API, or the SWIG [17] wrapped Python or Java APIs can be used. Furthermore, the libraries are available and supported on major operating systems and platforms, such as Linux, Mac OS X and Microsoft Windows.

```
import arc
import sys

# UserConfig contains information on credentials and default services to use.
usercfg = arc.UserConfig("", "")

# Simple job description which outputs hostname to stdout
jobdescstring = "&(executable=/bin/hostname)(stdout=stdout)"

# Parse job description
jobdescs = arc.JobDescriptionList()
if not arc.JobDescription_Parse(jobdescstring, jobdescs):
  print "Invalid job description"
  sys.exit(1);

# Use top-level NorduGrid information index to find resources
index = arc.Endpoint("ldap://index1.nordugrid.org:2135/Mds-Vo-name=NorduGrid,o=grid",
                     arc.Endpoint.REGISTRY,
                     "org.nordugrid.ldapegiis")
services = arc.EndpointList(1, index)

# Do the submission
jobs = arc.JobList()
submitter = arc.Submitter(usercfg)
if submitter.BrokeredSubmit(services, jobdescs, jobs) != arc.SubmissionStatus.NONE:
  print "Failed to submit job"
  sys.exit(1)

# Write information on submitted job to local job list (~/.arc/jobs.xml)
jobList = arc.JobInformationStorageXML(usercfg.JobListFile())
if not jobList.Write(jobs):
  print "Failed to write to local job list %s" % usercfg.JobListFile()

# Job submitted ok
print "Job submitted with job id %s" % jobs.front().JobID
```

**Figure 2.** Grid task submission using the ARC Python API.

An example of how to submit a job to the Grid is shown in Figure 2[1]. In this code the top level NorduGrid registry of Grid resources is first queried to obtain possible targets for the job. The default (random) broker is used to choose a suitable resource and then the job is submitted. Information on the job is then added to a local job list, so that the job can be tracked during execution and eventually the result downloaded once execution completes.

## 4. ARC Graphical Clients

The ARC Graphical Clients [18] is an attempt at providing graphical clients for the ARC middleware. To enable a native experience on all platforms, the C++ toolkit Qt [19] was used. C++ is also the native language of the ARC middleware, enabling a well-integrated solution.

The ARC Graphical Clients consist of the following applications:

- Proxy generation: Standalone application for generating a proxy from different sources (arcproxy-ui)
- Job management: Application for managing grid jobs using the ARC job list functionality (arcstat-ui)
- Job submission: Application to handle the submission of parameter sweeps (arcsub-ui)
- Storage Explorer: Application for accessing grid storage (arcstorge-ui)

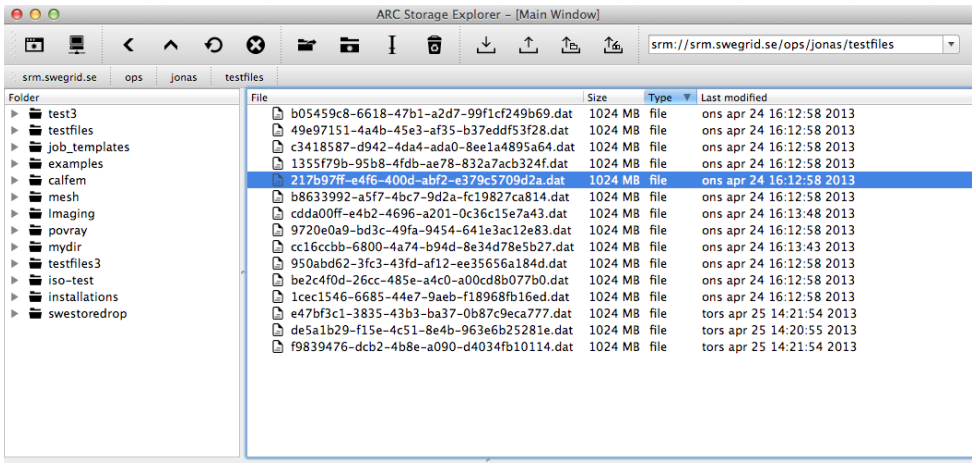


**Figure 3.** Main window of the ARC Storage Explorer.

The ARC Storage Explorer was built to simplify access to the national storage infrastructure in Sweden. By using the ARC SDK, all the details of accessing different storage protocols is hidden and effort can be directed on designing the user interface. An example screenshot of the ARC Storage Explorer browsing data through the SRM protocol is shown in Figure 3. The right pane has the list of files in the current directory along with some attributes of each file and the left pane has an expandable tree-like structure of the directory hierarchy. A selection of buttons at the top allow traversing the hierarchy, uploading, downloading, renaming and deleting files as well as other operations.

The ARC Storage Explorer currently supports the FTP, GridFTP and SRM protocols. It can also support more protocols as these are implemented in the ARC middleware without any changes to the code, due to the ARC SDK's modular nature. The ARC Graphical clients are currently available for Mac OS X and Linux.

[1] This example uses version 3.0.0 of the ARC API and is not guaranteed to be compatible with other versions.

## 5. Conclusion

This paper has presented a brief overview of the ARC SDK, explained its modular structure and available plugins. An application built on the SDK, the ARC Storage Explorer, has been shown as an example of the way that tools can be developed against an implementation-independent interface which allows the freedom to choose between pluggable underlying technologies. The SDK is a mature and well-supported product which is well suited for any application wishing to exploit Grid resources.

## References

[1] Foster I and Kesselman C 1999 *The Grid: Blueprint for a New Computing Infrastructure* (Morgan Kaufmann)
[2] Ellert M *et al.* 2007 *Future Gener. Comput. Syst.* **23** 219–240 ISSN 0167-739X
[3] Public-Key Infrastructure (X.509) (PKI), Proxy Certificate Profile URL `http://rfc.net/rfc3820.html`
[4] Alfieri R *et al.* 2005 *Future Gener. Comput. Syst.* **21** 549–558 ISSN 0167-739X
[5] 2011 European Middleware Initiative (EMI) Execution Service (ES) web site URL `https://twiki.cern.ch/twiki/bin/view/EMI/EmiExecutionService`
[6] Allcock W *et al.* 2002 *Parallel Comput.* **28** 749–771 ISSN 0167-8191
[7] Aiftimiei C *et al.* 2008 Job Submission and Management Through Web Services: the Experience with the CREAM Service *Proc. of CHEP 2007, J. Phys.: Conf. Ser. 119 062004* ed R Sobie R T and Thomson J (IOP) URL `http://dx.doi.org/10.1088/1742-6596/119/6/062004`
[8] Foster I *et al.* 2007 OGSA$^{TM}$ Basic Execution Service Version 1.0 gFD-R-P.108 URL `http://www.ogf.org/documents/GFD.108.pdf`
[9] The Globus Resource Specification Language RSL v1.0. URL `http://www.globus.org/toolkit/docs/2.4/gram/rsl_spec1.html`
[10] Pacini F and Maraschini A 2007 *Job Description Language attributes specification* EGEE-JRA1-TEC-590869-JDL-Attributes-v0-8 URL `https://edms.cern.ch/document/590869/1`
[11] Anjomshoaa A *et al.* 2008 Job Submission Description Language (JSDL) Specification, Version 1.0 (first errata update) gFD-R.136 URL `http://www.ogf.org/documents/GFD.136.pdf`
[12] Humphrey M *et al.* 2007 JSDL HPC Profile Application Extension, Version 1.0 gFD-R.111 URL `http://www.ogf.org/documents/GFD.111.pdf`
[13] Kónya B and Johansson D *The NorduGrid/ARC Information System* The NorduGrid Collaboration NORDUGRID-TECH-4 URL `http://www.nordugrid.org/documents/arc_infosys.pdf`
[14] Field L and Schulz M W 2004 Grid deployment experiences: The path to a production quality ldap based grid information system. *Proceedings of the Conference for Computing in High-Energy and Nuclear Physics* pp 723–726
[15] Field L, Memon S, Marton I and Szigeti G 2013 *Journal of Grid Computing* 1–12
[16] Cameron D, Gholami A, Karpenko D and Konstantinov A 2010 Adaptive data management in the arc grid middleware *J. Phys.: Conf. Ser.*
[17] Swig, simplified wrapper and interface generator URL `http://www.swig.org`
[18] Arc graphical clients URL `http://arc-gui-clients.sourceforge.net/`
[19] Qt URL `https://qt-project.org`