# EGEE-III

# SAGA Service Discovery

## Users Guide for C++ Programmers

| | |
|---|---|
| Document identifier: | sagaC++SD |
| Date: | **08/04/2011** |
| Activity: | **SA3: Integration, Testing and Certification** |
| Lead Partner: | **STFC** |
| Document status: | **FINAL** |

Abstract: This document provides the C++ programmer with the information necessary to get started with the SAGA Service Discovery API using the gLite adapter

This work is co-funded by the EC EMI project under the FP7 Collaborative Projects Grant Agreement Nr. INFSO-RI-261611.

## Document Log

| Issue | Date | Comment |
|---|---|---|
| | | |
| | | |
| 0-0 | | |

# TABLE OF CONTENTS



# TABLE OF TABLES

# 1. INTRODUCTION

## 1.1. PURPOSE

This document is intended to get people started with SAGA Service Discovery. It contains instructions on how to build and run an application with reference to the gLite adapter for Service Discovery.

## 1.2. DOCUMENT ORGANISATION

The document starts with an overview of the SAGA Service Discovery API with reference to the GLUE model and the API's classes. Guidance on the required environment variables in Section 3. is followed by descriptions of possible exceptions in Section 4. This is followed by a simple example of how to use the API. Section 6. gives details about the use of the filters that are used by the API to select services. Section 7. gives some notes on GLUE and Section 8. lists the limitations of the gLite adapter.

## 1.3. APPLICATION AREA

This guide is intended for use by middleware developers who's code needs to locate services as well as grid users who are inserted in finding services that match given filters.

## 1.4. DOXYGEN

The doxygen html files for the SAGA C++ API can be found in /usr/`share/doc/glite-saga-adapters-cpp/html/`

## 1.5. REFERENCES

**Table 1: Table of references**

| | |
|---|---|
| R1 | Goodale T. *et al*. (2008). *A Simple API for Grid Applications (SAGA)*. Retrieved July 01, 2009, from Open Grid Forum website: http://www.ogf.org/documents/GFD.90.pdf |
| R2 | Fisher S., Wilson A. and Paventhan A. (2009). *SAGA API Extension: Service Discovery API.* Retrieved July 01, 2009, from Open Grid Forum website: http://www.ogf.org/documents/GFD.144.pdf |
| R3 | Andreozzi S. *et al*. (2007). *GLUE Schema Specification version 1.3*. Retrieved July 01, 2009, from Open Grid Forum website: https://forge.gridforum.org/sf/docman/do/downloadDocument/projects.glue-wg/docman.root.background.specifications/doc14185 |
| R4 | Andreozzi S. *et al*. (2009). *GLUE Schema Specification version 2.0*. Retrieved July 01, 2009, from Open Grid Forum website: http://www.ogf.org/documents/GFD.147.pdf |

# 2. OVERVIEW

## 2.1. SAGA

SAGA is the "Simple API for Grid Applications", a high level, application-oriented API for grid application development. This is an Open Grid Forum (OGF) standard, [R1].

## 2.2. THE SAGA SERVICE DISCOVERY API

The Service Discovery (SD) API is an extension to SAGA, [R2]. The SD API provides a mechanism to locate services. The quality of the information returned will depend upon the quality of the data in the back-end system. The gLite adapter for Service Discovery obtains information from a BDII.

## 2.3. SERVICE MODEL

This API is based upon the GLUE (version 1.3) model of a service [R3] as summarized in Figure 1. This service model is also compatible with GLUE 2.0 [R4]. A *Site* may host many *Services* and a *Service* has multiple *ServiceData* entries associated with it. Each *ServiceData* entry is represented by a key and a value, thus allowing any set of keyword/value pairs to be associated with an instance of a *Service*. In addition, a *Service* has a many-to-many relationship with itself. This allows the model to describe groupings of services.

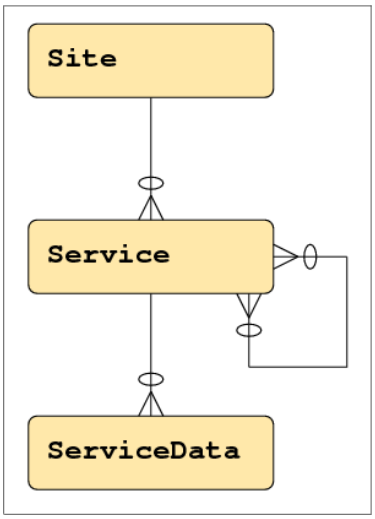


**Figure 1** - ER diagram of Service Model

## 2.4. CLASSES

This API consists of a `discoverer` class with a single method: `list_services`. This method returns a list of objects of the `service_description` class, filtered according to several specified filters, Section 6.

The `service_description` class has three methods. One of these, `get_url`, is all that most people will use to obtain the address registered for the service. In the case of a Web Service, this will be the service endpoint. It also exposes *ReadOnly* properties of the service, such as its type and uid as attributes. There is a method `get_related_services` that returns the set of related `service_description` objects, which represent related services. Finally, there is a method

`get_data` to access the set of further key value pairs. This method returns a `service_data` object, which gives *ReadOnly* access to all the key names and values in the `service_data` object. By referencing a separate `service_data` object holding the key value pairs, potential key name clashes between the sets of pre-defined and free-form attributes are avoided.

### 2.5. DETAILS

This API uses an information system and does not contact the services to check their availability. The user must expect that a service provided by the Service Discovery API may not be available. Even if the API were to contact a service to confirm its availability, by the time the user attempts to use that service, it may have failed. Similarly the API cannot be guaranteed to provide a complete set of matching services.

The API may try to use a BDII but not be able to access it. If no result can be returned because of BDII or other internal problems, it will throw the `NoSuccess` exception.

### 2.6. GLITE ADAPTER

The C++ gLite adapter uses the BDII information system.

# 3. GETTING STARTED WITH SAGA SERVICE DISCOVERY

## 3.1. INSTALLATION

It is recommended that yum is used to install the client:

```
yum install emi.saga-adapter.sd-cpp
```

## 3.2. GRID CERTIFICATE

It is not essential to have a grid certificate. A proxy of your certificate is only needed if you try to select services using the default authorization filter (see Section 6.3.). If you do use a proxy then this must be a VOMS proxy as the extended attributes are required.

## 3.3. ENVIRONMENT VARIABLES

You should ensure the required variables are set as listed below.

### 3.3.1. Building

`SAGA_LOCATION=/usr`

The SAGA_LOCATION is the location of the C++ installation of SAGA.

Boost headers to match the installed boost libraries are needed (typically `boost-devel` in the case of an RPM).

### 3.3.2. Executing

`SAGA_LOCATION=/usr`

The SAGA_LOCATION is the location of the C++ installation of SAGA.

`LD_LIBRARY_PATH=$SAGA_LOCATION/lib/`

The LD_LIBRARY_PATH is required by the C++ adapter. This should also include the path to the `boost` libraries.

`x509_USER_PROXY=/tmp/x509up_uxxx`

The X509_USER_PROXY is the location of the users proxy, where xxx is the users uid.

`BDII_URL=ldap://host.domain:2170`

The BDII_URL is the URL of the LDAP server to be used by the adapter when a URL is not passed in via the API. This takes preference over the LCG_GFAL_INFOSYS and the hard coded value of the CERN bdii.

`LCG_GFAL_INFOSYS=host.domain:2170`

LCG_GFAL_INFOSYS will be used if BDII_URL is not set. This takes preference over the hard coded value of the CERN bdii.

## 4. SAGA EXCEPTIONS

**saga::exception** is the base class for all exceptions in SAGA.

**AuthenticationFailed** is thrown if none of the available session contexts could successfully be used for authentication.

**AuthorizationFailed** is thrown if none of the available contexts of the used session could be used for successful authorization. This error indicates that the resource could not be accessed at all, and not that an operation was not available due to restricted permissions.

**BadParameter** is thrown if any filter has an invalid syntax or if any filter uses invalid keys. However the dataFilter never signals invalid keys as there is no schema with permissible key names.

**DoesNotExist** is thrown if the URL is syntactically valid, but no service can be contacted at that URL.

**IncorrectState** is thrown if the object a method was called on is in a state where that method cannot possibly succeed.

**IncorrectURL** is thrown if an implementation cannot handle the specified protocol, or that access to the specified entity via the given protocol is impossible.

**NoSuccess** is thrown if no result can be returned because of information system or other internal problems.

**NotImplemented** is thrown if the method is not implemented by that SAGA implementation at all.

**PermissionDenied** is thrown when the method failed because the identity used did not have sufficient permissions to perform the operation successfully.

**Timeout** is thrown if a remote operation did not complete successfully because the network communication or the remote service timed out.

# 5. A SIMPLE EXAMPLE

## 5.1. THE CODE

```
1  #include <iostream>
2  #include <vector>
3  #include <string>
4  #include "saga/saga.hpp"
5  #include "saga/saga/sd.hpp"

6  int main(int argc, char *argv[])
7  {
8     std::string svcFilter = "type='org.glite.lb.server'";
9     std::string dataFilter = "glite-info-service_version='1.5'";
10    std::string authzFilter = "";

11    try
12    {
13       // Create a Discoverer
14       saga::sd::discoverer sd;

15       // Use the Discoverer to select services
16       std::vector<saga::sd::service_description> services;
17       services = sd.list_services(svcFilter, dataFilter, authzFilter);

18       if (services.size()> 0) // Take the first url
19       {
20          std::cout << "URL: " << services[0].get_url() << std::endl;
21       }
22       else
23       {
24          std::cout << "No matching services found" << std::endl;
25       }
26    }

27    catch ( saga::exception& e )
28    {
29       std::cerr << "ERROR: " << e.get_message() << std::endl;
30       exit(1);
31    }
```

```
32     return 0;
33 }
```

### 5.2. EXPLANATION OF THE CODE

Lines 1-5 are the various include statements.

Lines 8-10 set the values of the filters.

Lines 14 creates a Discoverer. The underlying adapter will attempt to use the value set in the environment variable BDII_SERVER as the URL to contact the BDII.

Lines 16-17 uses the Discoverer to select services. This is done via the `listServices` method. The `listServices` method returns list of service descriptions, in a random order, matching the filter criteria. The use of filters in is described in Section 6.

Lines 18-25 uses the `getUrl` method to get the URL of the first service.

Lines 27-31 reports any SAGA exceptions.

### 5.3. BUILDING THE EXAMPLE

Ensure that the environment is set up correctly (see 3.3.). The following makefile will build the example.

```
NUM = one
SRC = sd_example_$(NUM).cpp
OBJ = $(SRC:%.cpp=%.o)
BIN = sd_example_$(NUM)
CXX = g++
CXXFLAGS = -g -c
LD = $(CXX)
LDFLAGS = -g

include $(SAGA_LOCATION)/share/saga/make/saga.engine.mk
include $(SAGA_LOCATION)/share/saga/make/saga.package.file.mk
SAGA_CPPFLAGS += $(shell $(SAGA_LOCATION)/bin/saga-config --cppflags)
SAGA_CXXFLAGS += $(shell $(SAGA_LOCATION)/bin/saga-config --cxxflags)
SAGA_LDFLAGS += $(shell $(SAGA_LOCATION)/bin/saga-config --ldflags)

all: $(BIN)

$(OBJ): %.o : %.cpp
      $(CXX) $(CXXFLAGS) $(SAGA_CXXFLAGS) -o $@ $<

$(BIN): $(OBJ)
      $(LD) $(LDFLAGS) $(SAGA_LDFLAGS) -o $@ $<
```

```
clean:
      @$(RM) $(OBJ) $(BIN)
```

Assuming the above is in a file called Makefile, the command 'make' will build the example

### 5.4. RUNNING THE EXAMPLE

Ensure that the environment is set up correctly (see 3.3.). Assuming you are in the directory where sd_example_one is then

```
./sd_example_one
```

will run the test application.

### 5.5. ACCESSING THE DATA FIELDS

In order to examine the contents of some of the data associated with a service replace lines 18– 25 with:

```
1     std::vector<saga::sd::service_description>::iterator servicesIter;
2     std::vector<saga::sd::service_description>::const_iterator
3       servicesEnd = services.end();

4     for ( servicesIter = services.begin();
5         servicesIter != servicesEnd;
6         ++servicesIter )
7     {
8       std::cout << std::endl << "Service Description of "
9         << servicesIter->get_url() << std::endl;
10      std::cout << "Name: "<< servicesIter->get_attribute("Name")
11        << std::endl;
12      saga::sd::service_data data;
13      data = servicesIter->get_data();

14      std::vector<std::string> attribs =
15        data.get_vector_attribute("glite-info-service_hostname");
16      std::vector<std::string>::const_iterator attribsIter;
17      std::vector<std::string>::const_iterator attribsEnd =
18        attribs.end();
19      for ( attribsIter = attribs.begin(); attribsIter != attribsEnd;
20          ++attribsIter )
21      {
22        std::cout << "glite-info-service_hostname: "
23          << *attribsIter << std::endl;
24      }

25    }
```

Lines 1-3

Lines 4-25 loops round all of the service descriptions.

Lines 8-9 gets and displays the URL of the service.

Lines 10-11 gets and displays the name of the service. “Name” is a scalar attribute so the getAttribute method is used.

Lines 12-13 gets the ServiceData object associated with the service.

Lines 14-24 displays the values for the ServiceData attribute with the key “glite-info-service_hostname”. N.B. All ServiceData attributes are vectors so the getVectorAttribute method is used.

# 6. FILTERS

There are three filter strings: `serviceFilter`, `dataFilter` and `authzFilter` which act together to restrict the set of services returned. Each of the filter strings uses SQL92 syntax as if it were part of a `WHERE` clause acting to select from a single table that includes columns as described below for that filter type. SQL92 has been chosen because it is widely known and has the desired expressive power. Multi-valued attributes are treated as a set of values.

Three strings are used, rather than one, as this clarifies the description of the functionality, avoids problems with key values being themselves existing GLUE attributes, and facilitates implementation as it makes it impossible to specify constraints that correlate, for example, service and authz information. Only the following operators are permitted in expressions not involving multi-valued attributes: `IN`, `LIKE`, `AND`, `OR`, `NOT`, `=`, `>=`, `>`, `<=`, `<`, `<>` in addition to column names, parentheses, column values as single quoted strings, numeric values and the comma. For a multi-valued attribute, the name of the attribute MUST have the keyword `ALL` or `ANY` immediately before it (see Limitations), unless comparison with a set literal is intended. For each part of the expression, the attribute name MUST precede the literal value.

The `LIKE` operator matches string patterns:

`'%xyz'` matches all entries with trailing xyz

`'xyz%'` matches all entries with leading xyz

`'%xyz%'` matches all entries with xyz being a substring

The `ESCAPE` keyword can be used with `LIKE` in the normal way.

Column names are not case sensitive but values are.

The operators `>=`, `>`, `<= and <` are not supported when applied to strings.

## 6.1. SERVICE FILTER

Column names in the `serviceFilter` are:

`Capabilities`

identifiable aspects of functionality

`ImplementationVersion`

version of service's implementation

`Implementor`

name of the organisation providing the implementation of the service

`InterfaceVersion`

version of service's interface

`Name`

name of service (not necessarily unique)

`RelatedServices`

the uids of services related to the one being looked for

`Site`

name of site the service is running at

`Type`

type of service. This API does not restrict values of the service type -- it might be a DNS name, a URN or any other string.

`Uid`

unique identifier of service

`Url`

the endpoint to contact the service - will normally be used with the LIKE operator

Some examples are:

```
Site IN ('INFN-CNAF', 'RAL-LCG2')
Type = 'org.glite.ResourceBroker' AND Site LIKE '%.uk' AND Implementor = 'EGEE'
```

### 6.2. DATA FILTER

Column names in the the `dataFilter` string are matched against the service data key/value pairs. No keys are predefined.

If values are specified as numeric values and not in single quotes, the service data will be converted from string to numeric for comparison.

Data attributes may be multi-valued. If a `dataFilter` string does not have the correct syntax to accept multi-valued attributes, and a service has more than one value for an attribute mentioned in the filter, that service MUST be rejected.

Some examples are:

```
source = 'RAL-LCG2' OR destination = 'RAL-LCG2'
RunningJobs >= 1 AND RunningJobs <= 5
```

### 6.3. AUTHZ FILTER

The set of column names in the `authzFilter` is not defined. Instead the list below shows a possible set of names and how they might be interpreted. Each of these column names could reasonably be related to an authorization decision.

Vo

virtual organization - will often be used with the IN operator

Dn

an X.509 ``distinguished name''

Group

a grouping of people within a Virtual Organization

Role

values might include ``Administrator'' or ``ProductionManager''

It is expected that many of the attributes used in the `authzFilter` will be multi-valued.

Some examples, where `VO` is assumed to be multi-valued are:

```
ANY Vo IN ('cms', 'atlas')
Vo = ('dteam')
```

Note the use of the set constructor in both examples. Being a set, ('aaa','bbbb') is of course the same as ('bbb', 'aaa').

The `listServices` method is overloaded: the last parameter the `authzFilter` may be omitted. If it is omitted the authorization filtering is performed on the contexts in the session, see below. This is quite different from including the `authzFilter` parameter with an empty string which means that there is **no** authz filtering.

The gLite adapter uses the users proxy to construct the context. If set it will use the environment variable X509_USER_PROXY, it will then try looking for /tmp/x509up_uxxx, where xxx is the user's uid. An exception will be thrown when the `authzFilter` is omitted and a proxy certificate is not available via one of the above mechanisms.

## 7. GLUE 1 AND GLUE 2

The adapter is capable of using GLUE 1 and the prototype GLUE 2 information from gLite bdiis. The type of data, GLUE 1 and/or GLUE 2, that gets queried is controlled from an ini file, $SAGA_LOCATION/etc/saga-adapter/config-cpp/glite_sd_adaptor.ini. It is possible for the user to provide their own ini files, which should be placed in an ini directory. In order for SAGA to find this directory create the file ~/.saga.ini which should contain the lines:

```
[saga]
ini_path = $[saga.location]/share/saga/:"your ini directory"
```

Make a copy of $SAGA_LOCATION/etc/saga-adapter/config-cpp/glite_sd_adaptor.ini in your ini directory. The values in your file will then override the values in $SAGA_LOCATION/etc/saga-adapter/config-cpp/glite_sd_adaptor.ini.

If both GLUE 1 and GLUE 2 are selected then the results will be combined. This may lead to duplicate UIDs appearing in the returned service descriptions.

## 8. LIMITATIONS

This section lists the limitations of the adapter.

- Cannot use ALL or ANY with the service filter - this affects multi-valued attributes (related services and capabilities)
- The createDiscoverer method does not throw a DoesNotExistException if the information service cannot be contacted via the provided URL. This is because the adapter does not contact the information system until the listServices method is called. As a result a NoSecsessException gets thrown by the listServices method if the information service cannot be contacted via the provided URL.
- When querying GLUE1, the attribute 'implementor' is ignored if it is used in the service filter
- For GLUE1, a service filter query of the form "a = (x, y ,z)" will match services where "x = v, w, x, y, z" and not just "x = x, y, z"
- It is possible that a service will be published in both GLUE 1 and GLUE 2 formats. If the adapter is configured to use GLUE 1 and 2 there maybe multiple service descriptions for the same UID. The GLUE 2 version of the description is likely to have additional fields populated i.e. capabilities and implementor.