EGEE

EGEE User's Guide

SERVICE DISCOVERY

Document identifier:	**EGEE-JRA1-TEC-578147-v1.0**
Date:	**March 16, 2009**
Activity:	**JRA1: Middleware Engineering and Integration (UK Cluster)**
Document status:	**FINAL**
Document link:	**http://edms.cern.ch/document/578147/1**

Abstract: This document describes how to use the Service Discovery API to query services in the information system.

Document Change Log

Issue	Date	Comment	Author
1.0	30/03/05	First release	JRA1-UK

Document Change Record

Issue	Item	Reason for Change

CONTENTS

1 INTRODUCTION

The Service Discovery (SD) API provides a uniform interface to access service details published by information systems. It provides a set of methods to retrieve information about a known service (using its unique name) and a set of list methods to find services based on certain criteria.

2 QUICK-START GUIDE

This section gives you some examples in Java and C of how to use the Service Discovery API and instructions on getting them up and running.

2.1 JAVA

2.1.1 CREATING AN INSTANCE

There are three ways that an implementation of the ServiceDiscovery interface can be instantiated:

1. Use method `create(className)` in ServiceDiscoveryFactory, which creates an instance of the class with the given name.
2. Use method `create()` in ServiceDiscoveryFactory, which creates an instance of the class specified by the `sd.class` system property.
3. Instantiate the required implementation explicitly (i.e. using `new`).

For example, to select R-GMA at run-time, the code should create an instance as:

```
ServiceDiscovery sd = ServiceDiscoveryFactory.create();
```

and should be executed with:

```
java -Dsd.class=org.glite.rgma.discovery.RGMAServiceDiscovery MyServiceDiscoveryClient
```

2.1.2 COMPILING AND RUNNING

To compile and run Java Service Discovery code, your classpath needs to contain:

- The SD `api-java` JAR file.
- An implementation of the SD interfaces for a specific information system.
- Classes and JAR files associated with the information system, e.g. `glite-rgma-api-java` and `glite-rgma-stubs-servlet-java`.

When running the examples, you may also have to pass in system properties to the underlying information system. For example, using R-GMA, you need to specify a value for RGMA_HOME:

```
java -DRGMA_HOME=$RGMA_HOME MyServiceDiscoveryClient
```

2.1.3 EXAMPLE CODE

The following code snippet prints out information about a known service.

```
try {
    ServiceDiscovery sd = ServiceDiscoveryFactory.create();

    String serviceName = "mysrm2.srm.cern.ch";

    ServiceDetails serviceDetails = sd.getServiceDetails(serviceName);

    System.out.println("WSDL: " + serviceDetails.getWSDL());
    System.out.println("Endpoint: " + serviceDetails.getEndpoint());

    ServiceData[] data = serviceDetails.getData();
    for (int i = 0; i < data.length; i++) {
        System.out.println(data[i].getKey() + " = " + data[i].getValue());
    }

} catch (ServiceDiscoveryException e) {
    // Handle the error...
}
```

The following code samples show how to use the list methods to discover services based on various attributes, e.g. type, service data or VO. It assumes you have already instantiated an object `sd` that implements the `ServiceDiscovery` interface.

To get a list of all services...

```
Service[] services = sd.listServices(null, null, null);
```

To search for all SRM services in ATLAS or CMS VOs...

```
Service[] services = sd.listServices("srm", null, new String[] {"atlas", "cms"});
```

To search for all GridFTP services associated with service `mysrm2.srm.cern.ch` that are hosted on site `site01.rl.ac.uk`...

```
Service[] services = sd.listAssociatedServices("mysrm2.srm.cern.ch", "gridftp",
    "site01.rl.ac.uk", null);
```

To search for all available services in the ALICE VO that have service data with `key1 = val1` and `key2 = val2`...

```
Service[] services = sd.listServicesByData(
    new ServiceData[] {
        new ServiceData("key1", "val1"),
        new ServiceData("key2", "val2")
        },
    null, null, new String[] {"alice"});
```

2.2 C

2.2.1 COMPILING AND RUNNING

The Service Discovery C API consists of just two files:

```
$GLITE_LOCATION/include/ServiceDiscovery.h
$GLITE_LOCATION/lib/libglite-sd-c-rgma.a
```

The header file (which you will need to include in your code to use the API) defines the Service Discovery interface, which is independent of the underlying Information System. The library file is an R-GMA implementation of the Service Discovery interface, and is built on the R-GMA C API (libglite-rgma-c), so you will need to install the R-GMA C API in order to use it.

You can compile and link a C program "myprog.c" against the Service Discovery C API using:

```
gcc myprog.c -o myprog -I$RGMA_HOME/include -L$RGMA_HOME/lib \
    -lglite-sd-c-rgma -lglite-rgma-c -lssl
```

Operating systems other than Scientific Linux may require explicit linking with sockets libraries in addition to OpenSSL. RGMA_HOME is usually set to the same location as GLITE_LOCATION (which is usually /opt/glite).

Provided the R-GMA C API is installed correctly and you have access to an R-GMA information system (with valid credentials, if the server requires secure connections), this is all that is required.

2.2.2 EXAMPLE CODE

The following code snippet prints out information about a known service.

```
#include "ServiceDetails.h"

SDServiceDetails *sd;
SDException exception;
int i;

sd = SD_getServiceDetails("mysrm2.srm.cern.ch", &exception);
if (sd != NULL)
{
    if (sd->status == 0) printf("Service available.\n");
    printf("WSDL: %s\n", sd->wsdl);
    printf("Endpoint: %s\n", sd->endpoint);
    for (i = 0; i < sd->data->numItems; ++i)
    {
        printf("%s = %s\n", sd->data->items[i].key;
                            sd->data->items[i].value);
    }
    /* ...see the API documentation for the other fields available... */
    SD_freeServiceDetails(sd);
}
else if (exception.status == SDStatus_SUCCESS)
```

```
{
    printf("No such service.\n");
}
else
{
    printf("Call failed (%s)\n", exception.reason);
    SD_freeException(&exception);
}
```

All the Service Discovery API calls return a status via the SDException struct and since all of the calls (apart from SD_free*) involve a remote call to the information system, it is good practice to check it.

The following code samples show how to use the list methods to discover services based on various attributes, e.g. type, service data and VO.

To get a list of all services (regardless of type, site, VO or status)...

```
SDServiceList *sl;

sl = SD_listServices(null, null, null, 0, &exception);
if (sl)...
SD_freeServiceList(sl);
```

To search for all available SRM services in ATLAS or CMS VOs...

```
SDServiceList *sl;
char *names[] = {"atlas", "cms"};
SDVOList vos = {2, names};

sl = SD_listServices("srm", null, &vos, 1, &exception);
if (sl)...
SD_freeServiceList(sl);
```

To search for all GridFTP services associated with service `mysrm2.srm.cern.ch` that are hosted on site `site01.rl.ac.uk`...

```
SDServiceList *sl;
sl = SD_listAssociatedServices("mysrm2.srm.cern.ch", "gridftp",
                               "site01.rl.ac.uk", null, 0, &exception);
if (sl)...
SD_freeServiceList(sl);
```

To search for all available services in the ALICE VO that have service data with `key1=val1 and key2=val2`...

```
SDServiceList *sl;
SDServiceData items[] = {{"key1", "val1"}, {"key2", "val2"}};
SDServiceDataList data = {2, items};
char *names[] = {"alice"};
SDVOList vos = {1, names};

sl = SD_listServicesByData(&data, null, null, &vos, 1, &exception);
if (sl)...
SD_freeServiceList(sl);
```

3 REFERENCE GUIDE

3.1 DATA TYPES

The methods in the interface make use of three data types:

Service This gives basic information about a service:

- The endpoint used to contact the service.
- The unique service name.
- The type of service.
- The service version.

ServiceDetails This gives extra more detailed information about a service:

- An administration contact e-mail address.
- A list of associated services.
- A list of service data (key/value pairs).
- The name of the site that hosts the service.
- The list of VOs supported by the service.
- The URL of the WSDL for the service (if it is a Web Service).

ServiceData This contains a key/value pair.

3.2 OPERATIONS

3.2.1 GETTING INFORMATION ON A SPECIFIC SERVICE

The methods used to retrieve information about a known service are:

Name	getService
Parameters	service name : string
Return	Service
Description	Gets the service with the given name.

Name	getServiceData
Parameters	service name : string
Return	ServiceData[]
Description	Gets the list of attributes (name/value pairs) associated with the given service.

Name	getServiceDetails
Parameters	service name : string
Return	ServiceDetails
Description	Gets all the available data on the given service.

Name	getServiceDataItem
Parameters	service name : string key : string
Return	string
Description	Gets a parameter value for the service.

Name	getServiceSite
Parameters	service name : string
Return	string
Description	Gets the name of the site where the service runs.

Name	getServiceWSDL
Parameters	service name : string
Return	string
Description	Returns a URL to the service WSDL.

3.2.2 SEARCHING FOR SERVICES

The methods used to search for certain types of service are:

Name	listAssociatedServices
Parameters	service name : string service type : string (use `null` to search on all types) site : string (use `null` to search on all sites) VOs : string (use `null` to search on all VOs)
Return	Service[]
Description	Lists the services that are associated with the specified service and that match the specified type, site and VOs.

Name	listServices
Parameters	service type : string (use `null` to search on all types) site : string (use `null` to search on all sites) VOs : string (use `null` to search on all VOs)
Return	Service[]
Description	Lists all the services matching the specified type, site and VOs.

Name	listServicesByData
Parameters	data : ServiceData[] service type : string (use `null` to search on all types) site : string (use `null` to search on all sites) VOs : string (use `null` to search on all VOs)
Return	Service[]
Description	Lists the services matching the given service data (key/value pairs) and the specified type, site and VOs.

3.3 JAVA AND C API

See the JavaDoc and Doxygen documentation at http://hepunx.rl.ac.uk/egee/jra1-uk/glite-r1/ for more detailed information about each method in the Java and C Service Discovery interfaces.

4 KNOWN PROBLEMS AND CAVEATS

The Service Discovery API is still under development and is liable to change.