



Grid Advanced Information System

Developer's Guide

KMI-R1 Developer Team <kmi@moredream.org>

Document Version: 0.9 Date: June 13, 2005



Copyright 2002-2005 Korea Institute of Scientist and Technology Information.
All rights reserved.

This document is licensed under the terms of the K*Grid Public License.
The details of K*Grid Public License is found at
<http://kmi.moredream.org/downloads/license.html>

Contents

1. Key Concept.....	4
1.1. Introduction	4
1.2. Components.....	4
1.2.1. Datacan Factory Service (DFS).....	4
1.2.2. VO Roster Service (VRS)	5
1.2.3. VO Crawler Factory Service (VCFS)	6
1.2.4. MoreDream Providers.....	7
1.3. Schema	8
1.4. User Interface.....	11
1.5. Features	11
1.6. GAIS in actions	12
2. Querying the GAIS	13
2.1. Querying Service Data.....	13
2.2. Querying the GSH.....	14
3. Writing out the MoreDream Providers.....	15
3.1. Service Data Providers	15
3.2. Core GT3.2 Service Data Providers	15
3.3. Provider interfaces	16
3.4. Creating Custom Service Data Providers	16
3.5. For example, MceScriptProvider.....	16
3.6. Service data provider input	18
3.7. Service data provider output.....	19
3.8. Registering MoreDream information providers to RIPS.....	19
3.8.1. Register MCE Information Provider	19
3.8.2. Register MSE Information Provider.	20
4. Reference.....	22

1. Key Concept

1.1. Introduction

Grid information system is a critical component for Grid computing, by which all types of Grid resources are virtually integrated and their information can be effectually managed and accessed. Furthermore, the efficiency of Grid computing is dependent on the functionalities supported by Grid information system. But the existing information system such as MDS (Monitoring and Discovery System) of GT (Globus Toolkit), which is currently received wide publicity in Grid community, is not appropriate for a production-mode service or a long-lived service because it supports only basic functions. That is why we developed a new Grid information system named Grid Advanced Information System (GAIS), which is a versatile information system that provides information about the available resources on Grids and their status.

GAIS is the information services component of the MoreDream and is composed of a collection of OGSI-compliant services which add and extend the functionalities of GT3 MDS3. It is differentiated from the dynamic management and the flat network of directory servers mentioned below.

1.2. Components

GAIS is composed of three grid services and two information providers. Each service is related to manage and search information in a Grid, whereas two providers play information sources for the GAIS.

1.2.1. Datacan Factory Service (DFS)

Datacan (a compound word of “data” and “can”) is an enhanced version of GT3.x index service. Like an index service, it aggregates *Service Data* from Resource Services such as RIPS (Resource Information Provider Service) or other Grid service instances by means of the *Aggregator* mechanism. It also registers Grid service instances using the *ServiceGroup* mechanism. For these aggregation and registration, it uses the *RegistryPublishPrivder* mechanism. Additionally, it provides the following functionalities:

- 1) It removes stale data to assure data accuracy using the *Data Sweep* mechanism. When a datacan is created, the mechanism in the datacan calls a *ServiceDataSweeper*, which

- periodically checks the available time of registered service data and deletes old service data.
- It has two types. The one is a public datacan (*pubcan*) to announce its information to a VO, and the other is a private datacan (*prican*) to share it only in a domain. A DFS administrator can configure a pubcan suitable for a VO according to her policy and she can also set up a prican to serve some users' purpose in her domain. It operates together with CAS (Community Authority Service) to control access to a pubcan or prican (not implemented).

DFS manages the lifecycle of a datacan using the *Factory* mechanism and maintains the snapshot of DFS status (the list of published datacans) through the *Configuration* mechanism. The snapshot is stored in a configuration file. Figure 1 shows the structure of DFS.

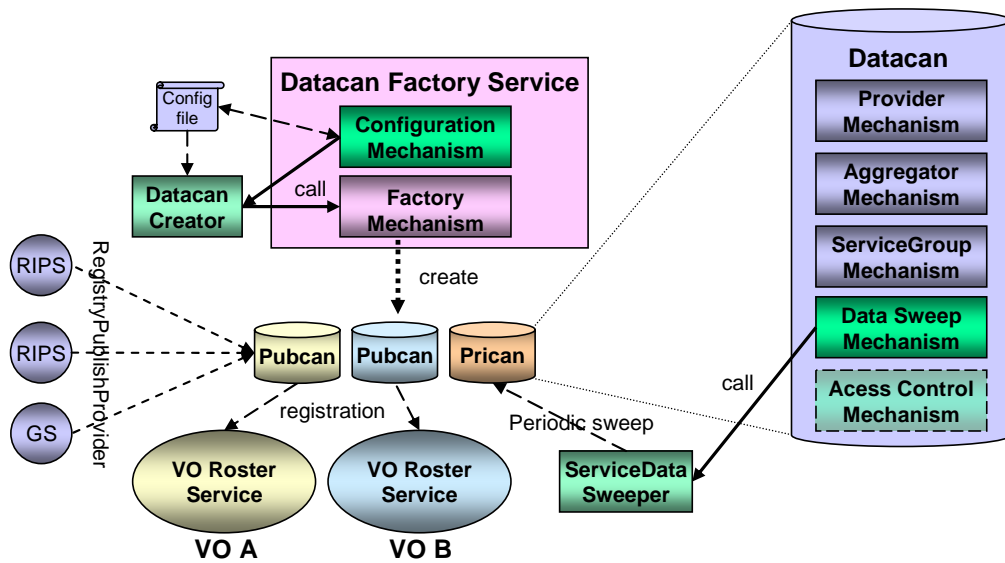


Figure 1. The structure of Datacan Factory Service

1.2.2. VO Roster Service (VRS)

Only one VRS exists in a VO because it typifies a VO. It manages the participants of VO and provides a registration interface to the VO. The structure of VRS is as Figure 2.

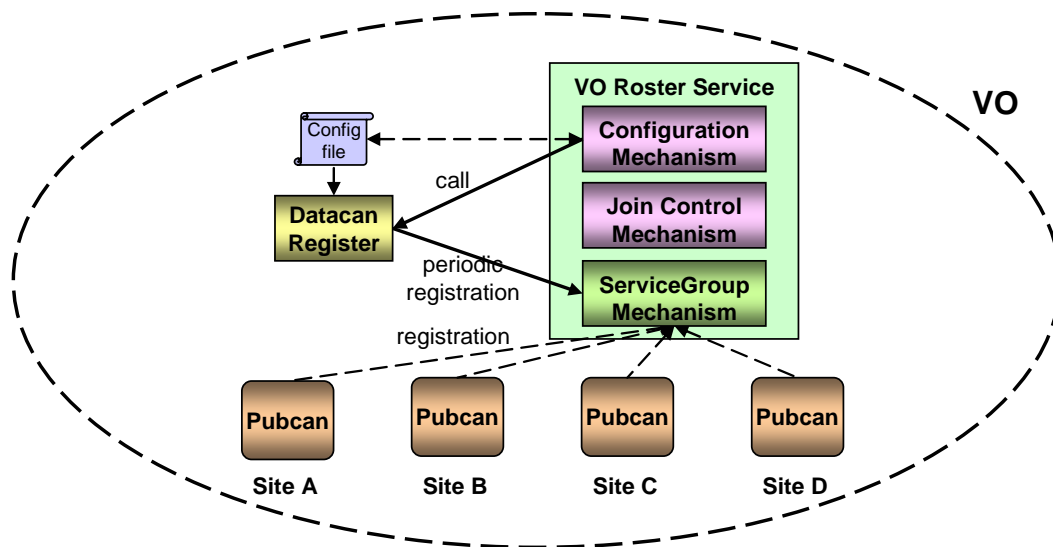


Figure 2. The structure of VO Roster Service

It provides the following functionalities:

- 1) It uses the *ServiceGroup* mechanism to register/unregister a datacan to its own VO.
- 2) It admits only a pubcan. The registration of a prican is rejected. The *Join Control* mechanism does this.
- 3) It makes use of the *Configuration* mechanism to store the snapshot of VO status (the list of registered datacan). But the status of resources in a VO changes dynamically. This may make the maintenance of the VO snapshot difficult. A *DatacanRegister* executes periodic registrations to preserve this.

1.2.3. VO Crawler Factory Service (VCFS)

VCFS provides a user with VO information. Only one VCFS exists in a VO because it is basically a service for a VO like VRS, but we recommend a site-based deployment of this service to avoid heavy load from plenty of users in a VO. This enables the load to be decentralized to each site in the VO. Figure 3 illustrates the structure of DFS.

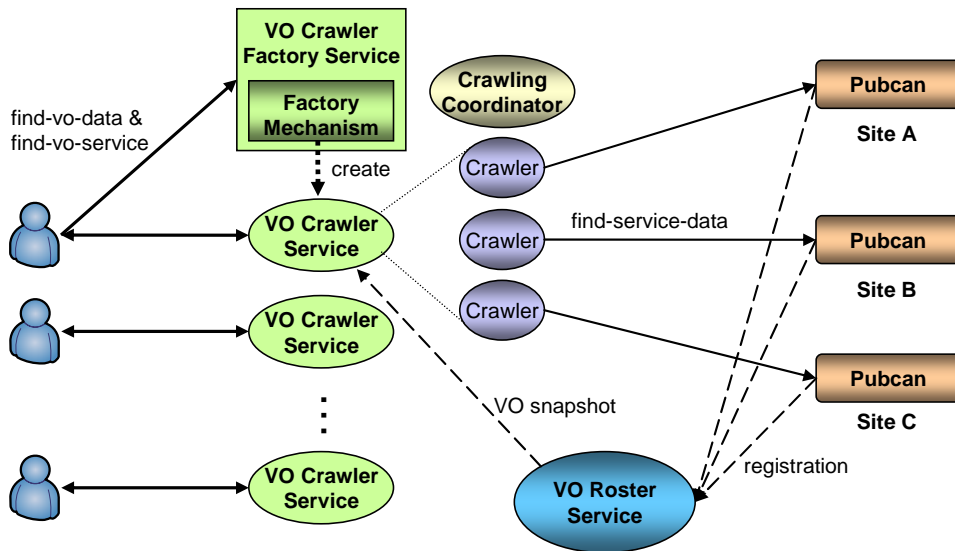


Figure 3. The structure of VO Crawler Factory Service

It provides the following functionalities:

- 1) This service has two query options. The one of two options is the *find-vo-data*, which crawls on VO information for a user. The other is the *find-vo-service*, which provides the location (GSH: Grid Service Handle) of a Grid service in a VO for a user. Actually, the *find-vo-service* is a special form of the *find-vo-data* to serve the convenience of a user.
- 2) It also creates the VO Crawler Service (VCS) using the Factory mechanism to protect a user session. VCS gains a VO snapshot from VRS. To achieve the efficiency of query, VCS creates the *Crawlers* corresponding to each participant (pubcan) using the Thread mechanism. Each crawler uses the OGSi *find-service-data* to query its own pubcan. The *CrawlingCoordinator* orchestrates each crawler's behavior.

1.2.4. MoreDream Providers

MoreDream Information Providers provide lots of information about data replication as well as computing resource. It is based on the use of provider execution mechanism. They are composed of MceScriptProvider and MseScriptProvider. MceScriptProvider provides lots of resource information used in K*Grid. It conforms to Glue schema and extends it. MseScriptProvider provides information about data replications. It defines a new information schema related to storage elements. You can easily obtain information that is produced in MCAT-enabled SRB (Storage Resource Broker) Server. If you'd like to add new information, you have only to create your own provider and register it to RIPS.

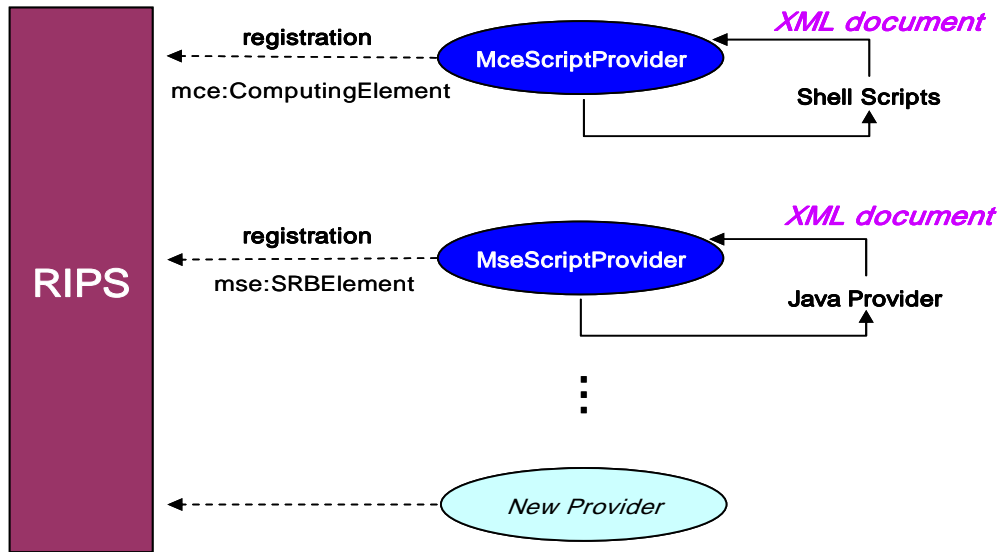


Figure 4. the registration of MoreDream Information Providers

1.3. Schema

GAIS uses the MoreDream schema, which adds and extends GLUE (Grid Laboratory Uniform Environment) schema. The schema is categorized into two element; MoreDream Computing Element (MCE) and MoreDream Storage Element (MSE). MCE enhances the computing element of GLUE schema 1.1 for supporting GRASP (Grid Resource Allocation Services Package), a resource management component of MoreDream. MSE is formed by processing the data replication information of SRB and it will be changed to agree to the storage element of GLUE schema later.

Table 1 shows the content of MCE.

Table 1. MoreDream Computing Element

Category	Object	Description	Unit
ComputingElement	Name	ComputingElement name	
	UniqueID	ComputingElement ID	
Info	LRMSType	Local Resource Manager type	
	LRMSVersion	Local Resource Manager version	
	GRAMVersion	GRAM version	
	HostName	Host name	
	GateKeeperPort	GateKeeper port	
	TotalCPUs	Total CPUs	
State	Staus	Queue status	
	TotalJobs	Total Jobs	

	RunningJobs	Running Jobs	
	WaitingJobs	Waiting Jobs	
	FreeCPUs	Free CPUs	
Policy	HostName	Host name	
UserStorage	LocalID	Local user ID	
	DN	User DN	
	Quota	Local user Quota	MB
	DefaultCapacity	Local user default Quota	MB
Job	GlobalID	Global Job ID	
	LocalID	Local Job ID	
	LocalOwner	Local Owner ID	
	GlobalOwner	Global Owner ID	
	Status	Job status	
Cluster	Name	Cluster name	
	UniqueID	Cluster ID	
SubCluster	Name	SubCluster name	
	UniqueID	SubCluster ID	
Filesystem	Name	File system name	
	Root	File system root	Path
	Size	Total size	MB
	AvailableSpace	Available space	MB
	ReadOnly	Read only or not	T/F
	Type	File system type	eg. NFS
Processor	Vendor	CPU vendor name	
	Model	Model name	
	Version	CPU version	
	Clockspeed	CPU Clock speed	MHz
	OtherProcessorDescription	Other description	
MainMemory	RAMSize	RAM size	MB
	RAMAvailable	Available RAM size	MB
	VirtualSize	Virtual RAM size	MB
	VirtualAvailable	Available virtual RAM size	MB
ProcessorLoad	Last1Min	1-minute average processor availability	%
	Last5Min	5-minute average processor availability	%
	Last15Min	15-minute average processor availability	%

OperatingSystem	Name	OS name	
	Release	OS Release #	
	Version	OS version	
NetworkAdapter	Name	Interface name	
	IPAddress	IP address	IP addr
	MTU	MTU size	Byte
	OutboundIP	OutboundIP or not	T/F
	InboundIP	InboundIP or not	T/F
Host	Name	Host name (Computation Element)	
	UniqueID	Host ID	
ProcessorLoad	Last1Min	1-minute average processor availability	%
	Last5Min	5-minute average processor availability	%
	Last15Min	15-minute average processor availability	%
MainMemory	RAMSize	RAM size	MB
	RAMAvailable	Available RAM size	MB
	VirtualSize	Virtual RAM size	MB
	VirtualAvailable	Available virtual RAM size	MB

Table 2 describes the content of MSE.

Table 2. MoreDream Storage Element

Category	Object	Description	Unit
SRBElement	CollectionName	Collection name	
	UserName	User name	
	ServerLocation	SRB server location	IP addr
SRBResources	CollectionName	Collection name	
	UserName	User name	
	ServerLocation	SRB server location	IP addr
SRBResource	ResourceName	Resource name	
	ResourceLocation	Resource location	IP addr
	ResourceType	Resource type	
	ResourceClassName	Resource class name	
	AdminName	Admin name	
	DomainDesc	Domain description	
	ZoneID	MCAT Zone ID	
SRBReplicas	CollectionName	Collection name	

	UserName	User name	
	ServerLocation	SRB server location	IP addr
SRBReplica	CollectionName	Collection name	
	UserName	User name	
	ServerLocation	SRB server location	IP addr
	FileName	File name	
	FileSize	File size	Byte
	FileType	File type	
ReplicaDetail	FileReplicationID	replica ID	
	ResourceLocation	Resource location	IP addr
	ResourceName	Resource name	

1.4. User Interface

There are now ways in which you can view VO information collected by GAIS or manage GAIS itself: the GAIS portlets and the GAIS PortType panels.

- 1) GAIS portlets: They offer resource information, service information or data replication information of a VO to users.
- 2) GAIS PortType panels: They enable a system administrator to control GAIS services.

The user interfaces will be added continuously as the version of GAIS is up.

1.5. Features

GAIS whose aim is to facilitate the management of information in Grid has the following features.

- 1) Dynamic management of directory server (datacan): GAIS can create a datacan easily whenever needed and can also destroy it freely. This enables a resource owner to share his resource according to his policies. For example, he can publish a datacan, which contains the entire information of his resource, for VO A, whereas the other datacan, which holds half of the information, for VO B.
- 2) Flat network of directory servers: The network of directory servers in GAIS is not configured hierarchically. Instead, it is flat. This has some merit. First, information is not duplicated. In hierarchy, higher level directory overlaps the information of lower level. Second, consistent synchronization of information is guaranteed. Hierarchical structure may pollute the

consistency of information when the fault or the subscription/unsubscription of a directory server.

- 3) Persistent configuration management: For a production-level service and a long-lived service, the configuration of an information system must be preserved persistently. GAIS provides a file-based configuration management.
- 4) Smart query processing: Simultaneous query using *Thread* mechanism alleviates the decline of the query performance which the flat network takes place. GAIS offers the *find-vo-service* operation, which can easily find a service in Grid, as well as the *find-vo-data* operation, which can search the information of a service in detail.
- 5) Rich information providers: GAIS supplies plenty of information about data replication as well as computing resource through providers conforming to MoreDream schema.

1.6. GAIS in actions

Figure 5 shows GAIS which configures 2 VOes (VO-A and VO-B) through the combination of 4 sites. VO-A is composed of site A, B and C, whereas VO-B consists of site B, C and D. A user has two query types. For VO query, he uses the *find-vo-data* and the *find-vo-service*. He also utilizes the *find-service-data* for Site query.

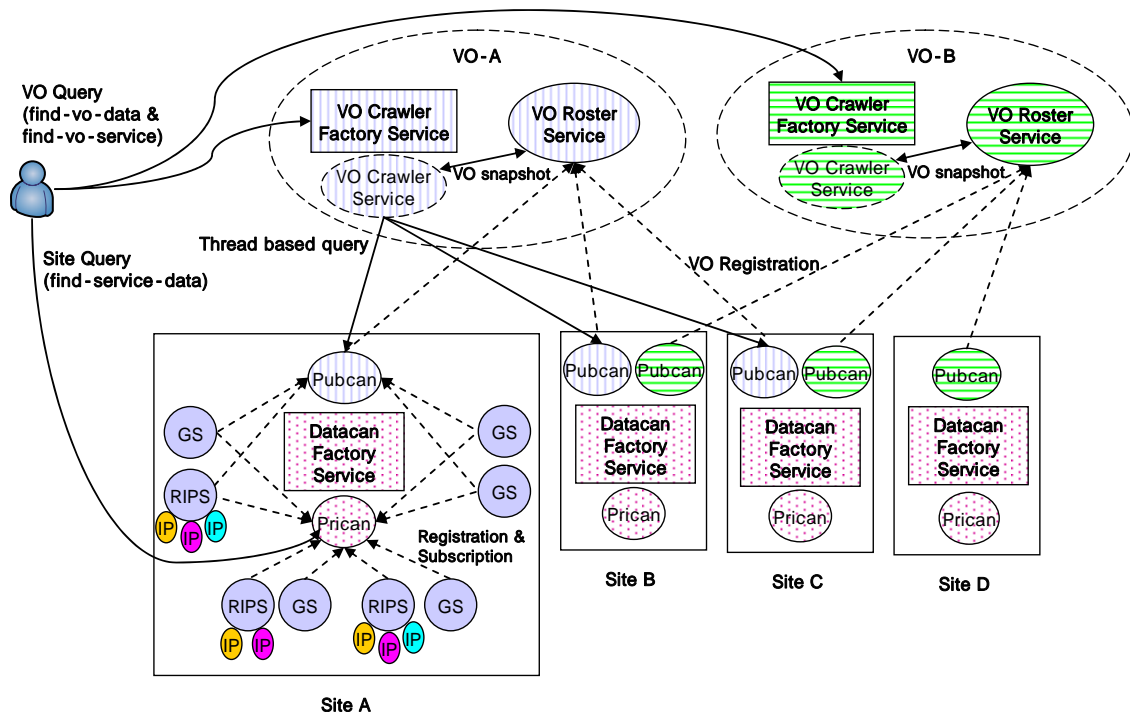


Figure 5. GAIS in actions

2. Querying the GAIS

GAIS has two type of query. One is *find-vo-data* and the other is *find-vo-service*. You can find Service Data in a VO using *find-vo-data*, and get the GSH (Grid Service Handle) of a Grid Service in a VO through *find-vo-service*.

2.1. Querying Service Data

The following code snippet describes the process to obtain the information in a VO. The type of result is the Element class. You can variously translate the format of this result through the AnyHelper class.

```
...
Element result = null;
VoCrawlerPortType voCrawler = null;

try {
    OGSIServiceGridLocator gridLocator = new OGSIServiceGridLocator();
    Factory factory = gridLocator.getFactoryPort(this.defaultEndpoint);
    GridServiceFactory voCrawlerFactory = new
GridServiceFactory(factory);

    LocatorType locator = voCrawlerFactory.createService();
    VoCrawlerServiceGridLocator voCrawlerLocator = new
VoCrawlerServiceGridLocator();
    voCrawler = voCrawlerLocator.getVoCrawlerPort(locator);

    ExtensibilityType queryResult =
voCrawler.findVoData(queryExpression);

    if (queryResult.get_any() == null) {
        // a null any in the case of an xpath query means 0 results
        Object obj = AnyHelper.getAsSingleObject(queryExpression);

        if (obj instanceof ServiceDataXPathQueryExpressionType) {
            throw new Exception("XPath Query: No results found");
        }
    }
}
```

```

    }
}

result = AnyHelper.getAsParentElement(queryResult);

} catch (Exception e) {
    e.printStackTrace();
} finally {
    try {
        if(voCrawler != null) {
            voCrawler.destroy();
        }
    } catch (Exception e1) {
        e1.printStackTrace();
    }
}
}
...

```

2.2. Querying the GSH

find-vo-service is a wrapper of **find-vo-data** to facilitate the discovery of a Grid Service. The type of result is the array of the String class.

```

...
String[] result = null;
VoCrawlerPortType query = null;

try {
    OGSIServiceGridLocator gridLocator = new OGSIServiceGridLocator();
    Factory factory = gridLocator.getFactoryPort(this.defaultEndpoint);
    GridServiceFactory queryFactory = new GridServiceFactory(factory);

    LocatorType locator = queryFactory.createService();
    VoCrawlerServiceGridLocator queryLocator = new VoCrawlerServiceGridLocator();
    query = queryLocator.getVoCrawlerPort(locator);
}

```

```

        result = query.findVoService(pattern);

    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            if(query != null) {
                query.destroy();
            }
        } catch (Exception e1) {
            e1.printStackTrace();
        }
    }
    ...

```

3. Writing out the MoreDream Providers

Before you read this material, visit <http://www-unix.globus.org/toolkit/docs/3.2/infosvcs/ws/developer/servicedataproviders.html>. This section is based on that document.

3.1. Service Data Providers

Service Data Provider components consist of the `ServiceDataProviderManager` Java class and one or more “plug-in” `ServiceDataProvider` classes, which are regularly executed by the Service Data Provider Manager (using Java `TimerTasks`). These provider plug-in programs can be the supplied providers that are part of GT3.2 or user-created, custom providers.

A valid provider is defined as any Java class that implements at least one of three predefined Java interfaces (`SimpleDataProvider`, `DOMDataProvider`, and `AsyncDataProvider`), and generates a well-formed, compatible XML document as the output of its execution.

“Well-formed” above means that the XML document can be parsed in any environment, i.e., any parsing tools written in any programming language can be used. “Compatible” above means a form compatible with the Service Data Provider Manager, i.e., a Java output stream or DOM representation.

3.2. Core GT3.2 Service Data Providers

GT3.2 supplies the following Service Data Providers:

SimpleSystemInformationProvider

HostScriptProvider

AsyncDocumentProvider

ScriptExecutionProvider

3.3. Provider interfaces

The Service Data Provider interfaces are designed to support execution in either a synchronous ("pull") mode or asynchronous ("push") mode. It is up to the developer to choose the appropriate provider interface to implement, based on specific application needs. There are three provider interfaces SimpleDataProvider, DOMDataProvider, and AsyncDataProvider.

3.4. Creating Custom Service Data Providers

Service Data Providers can be as simple or as complicated as the situation requires. The baseline case requires only that the provider developer create a Java class implementing the functions of one interface – SimpleDataProvider – whose purpose is to produce XML output in the form of a Java OutputStream as the result of its execution.

The following steps are the essence of creating a new Service Data Provider:

- 1) Choose the provider interface to be implemented, based on application needs or constraints.
- 2) Write code to produce your dataset as an XML document. This can either be in an OS-specific external program, or native Java code that is executed by the provider class itself.
- 3) Create an entry for the service in which you intend to run the provider in your auxiliary service configuration file. This service is assumed to have incorporated the functionality of the Service Data Provider Manager, which parses the provider configuration file property specified in the service's deployment descriptor entry (in the default server configuration file, server.config.wsdd), loads the provider, and executes it according to parameters specified by a client service.

3.5. For example, MceScriptProvider

New information providers can be made easily by modifying some methods and member variables in sample information providers. For example, MceScriptProvider is very similar to HostScriptProvider that is supplied by GT3.x. But MceScriptProvider provides lots of information

including the basic information about computing resources.

Let's make MceScriptProvider. First, choose AsyncDataProvider interface and implement it. If you intend to modify HostScriptProvider, you have only to change some member variables and method like composeDocument(). Second, write code to produce your dataset as an XML document. There are many script files to make XML document in MceScriptProvider. Then, you need to make a configuration file like gais-mce-providers.conf that defines the sequence of each script execution. Finally, there are some steps to register MceScriptProvider (See 3.8. Registering MoreDream information providers to RIPS).

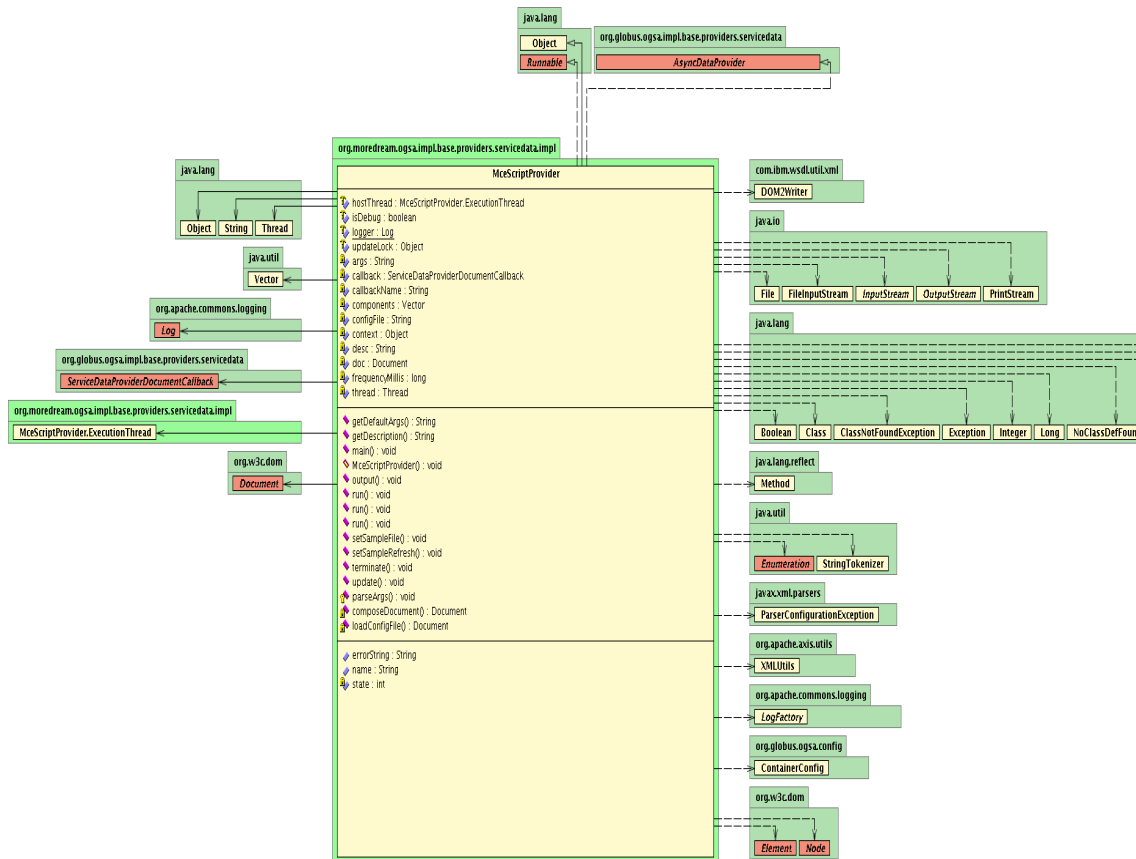


Figure 6. MceScriptProvider UML Diagram

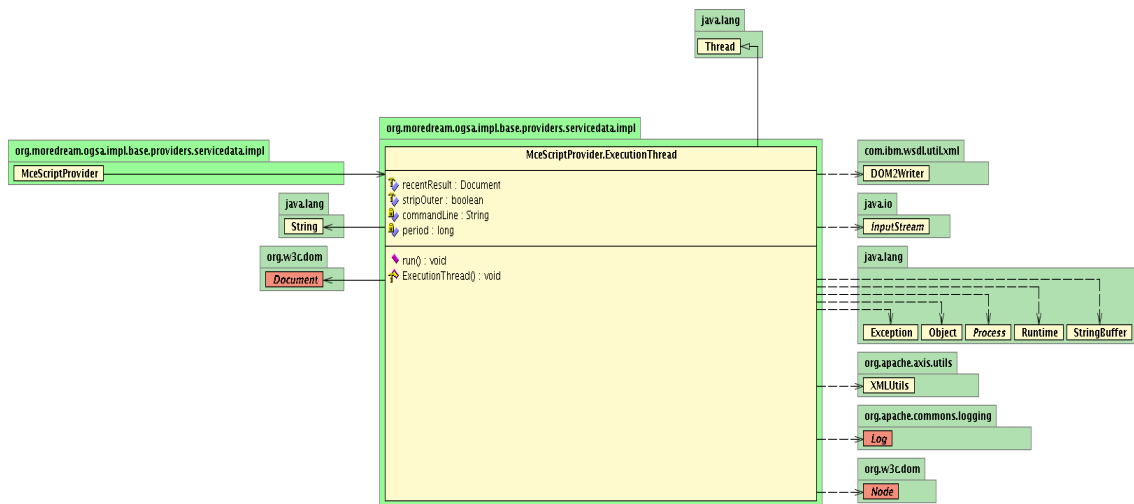


Figure 7. MceScriptProviderExecutionThread UML Diagram

3.6. Service data provider input

Input to Service Data Provider execution is specified via a set of string arguments to the run method. The argument string that gets passed to the provider is the serviceDataProviderArgs member of the ServiceDataProviderExecutionType structure that is passed to the executeProvider port type method. The getDefaultArgs method may be used to retrieve a default argument list for the provider. For Service Data Provider input, the following is an example of an XML serialized form of parameters to executeProvider :

```

<executedProviders>
...
<provider-exec:ServiceDataProviderExecution>
<provider-exec:serviceDataProviderName>MceScriptProvider
</provider-exec:serviceDataProviderName>
<provider-exec:serviceDataProviderImpl>
org.moredream.ogsa.impl.base.providers.servicedata.impl.MceScriptProvi
der</provider-exec:serviceDataProviderImpl>
<provider-exec:serviceDataProviderArgs>
</provider-exec:serviceDataProviderArgs>
<provider-exec:serviceName
xmlns:mce="http://www.moredream.org/ce/1.0"> mce:ComputingElement
</provider-exec:serviceName>
<provider-exec:refreshFrequency>30</provider-exec:refreshFrequency>

```

```

<provider-exec:async>true</provider-exec:async>
</provider-exec:ServiceDataProviderExecution>
...
</executedProviders>

```

3.7. Service data provider output

The output of a Service Data Provider is XML – either in the form of a Java OutputStream or a Java org.w3c.dom document. This output becomes the value of a Service Data Element and hence available as part of the hosting service's Service Data Elements. These Service Data Elements can then be used for the various WS Information Services functions.

3.8. Registering MoreDream information providers to RIPS

GAIS information providers should be registered to RIPS as follows (\$GLOBUS_LOCATION/etc/rips-service-config.xml). They produce {http://www.moredream.org/ce/1.0} ComputingElement and {http://www.moredream.org/se/1.0} SRBElement as service data.

3.8.1. Register MCE Information Provider

```

$ vi $GLOBUS_LOCATION/etc/rips-service-config.xml
...
<installedProviders>
<providerEntry
class="org.globus.ogsa.impl.base.providers.servicedata.impl.
ScriptExecutionProvider" handler="jobDataHandler"/>
<providerEntry
class="org.globus.ogsa.impl.base.providers.servicedata.impl.HostScript
Provider" />
<providerEntry
class="org.moredream.ogsa.impl.base.providers.servicedata.impl.
MceScriptProvider" />
</installedProviders>
...

```

```

<executedProviders>
...
<provider-exec:ServiceDataProviderExecution>
<provider-exec:serviceDataProviderName>MceScriptProvider
</provider-exec:serviceDataProviderName>
<provider-exec:serviceDataProviderImpl>
org.moredream.ogsa.impl.base.providers.servicedata.impl.MceScriptProvi
der</provider-exec:serviceDataProviderImpl>
<provider-exec:serviceDataProviderArgs>
</provider-exec:serviceDataProviderArgs>
<provider-exec:serviceName
xmlns:mce="http://www.moredream.org/ce/1.0"> mce:ComputingElement
</provider-exec:serviceName>
<provider-exec:refreshFrequency>30</provider-exec:refreshFrequency>
<provider-exec:async>true</provider-exec:async>
</provider-exec:ServiceDataProviderExecution>
...
</executedProviders>
...

```

3.8.2. Register MSE Information Provider.

```
$ vi $GLOBUS_LOCATION/etc/rips-service-config.xml
```

```

...
<installedProviders>
<providerEntry
class="org.globus.ogsa.impl.base.providers.servicedata.impl.
ScriptExecutionProvider" handler="jobDataHandler"/>
<providerEntry
class="org.globus.ogsa.impl.base.providers.servicedata.impl.HostScript
Provider" />
<providerEntry
class="org.moredream.ogsa.impl.base.providers.servicedata.impl.SRBscri
ptProvider" />
</installedProviders>
...
<executedProviders>

```

```
...
<provider-exec:ServiceDataProviderExecution>
<provider-exec:serviceName>SRBScriptProvider
</provider-exec:serviceName>
<provider-exec:serviceDataProviderImpl>
org.moredream.ogsa.impl.base.providers.servicedata.impl.SRBScriptProvi
der</provider-exec:serviceDataProviderImpl>
<provider-exec:serviceDataProviderArgs>
</provider-exec:serviceDataProviderArgs>
<provider-exec:serviceName
xmlns:mse="http://www.moredream.org/mse/1.0"> mse:SRB
</provider-exec:serviceName>
<provider-exec:refreshFrequency>30</provider-exec:refreshFrequency>
<provider-exec:async>true</provider-exec:async>
</provider-exec:ServiceDataProviderExecution>
...
</executedProviders>
...
```

4. Reference

[1] <http://www-unix.globus.org/toolkit/docs/3.2/infosvcs/ws/developer/index.html>