

# Key Concepts for Information Services

DRAFT

## Key Concepts for Information Services

### Overview

*Note:* If you have not done so already, we recommend reading the [GT4 Primer](#) before continuing. Also keep in mind that in this documentation, the concepts of MDS are written for the latest version, *WS MDS* (also known as *WS MDS*). The GT 4.2.0 release has deprecated the Pre-WS component, *MDS2*.

The *Monitoring and Discovery System (MDS)* is a suite of web services to monitor and discover resources and services on Grids. This system allows users to discover what resources are considered part of a *Virtual Organization (VO)* and to monitor those resources. MDS services provide query and subscription interfaces to arbitrarily detailed resource data and a trigger interface that can be configured to take action when pre-configured trouble conditions are met. The services included in WS MDS acquire their information through an extensible interface which can be used to:

- query WSRF services for resource property information,
- execute a program to acquire data, or
- interface with third-party monitoring systems.

Grid computing resources and services can advertise a large amount of data for many different use cases. WS MDS was specifically designed to address the needs of a Grid monitoring system – one that publishes data that is available to multiple people at multiple sites. As such, it is not an event handling system, like NetLogger, or a cluster monitor on its own, but can interface to more detailed monitoring systems and archives, and can publish summary data using standard interfaces.

---

## Table of Contents

1. WS MDS Aggregator Services .....	1
1. Index Service .....	1
2. Trigger Service .....	1
2. WS MDS Information Interfaces and Providers .....	2
1. Aggregator Framework .....	2
2. Information Providers .....	3
3. WebMDS user interface .....	4
3. Putting it all together .....	6
Glossary .....	7

DRAFT

---

## List of Figures

2.1. Table view in WebMDS .....	5
2.2. Detail view in WebMDS .....	5

DRAFT

# Chapter 1. WS MDS Aggregator Services

WS MDS includes two WSRF-based services: an *Index Service*, which collects data from various sources and provides a query/subscription interface to that data, and a *Trigger Service*, which collects data from various sources and can be configured to take action based on that data. An *Archive Service*, which will provide access to historic data, is planned for a future release.

## 1. Index Service

The Index Service is a registry similar to UDDI, but much more flexible. Indexes collect information and publish that information as *resource properties*. Clients use the standard WSRF resource property query and subscription/notification interfaces to retrieve information from an Index. Indexes can register to each other in a hierarchical fashion in order to aggregate data at several levels. Indexes are “self-cleaning”; each Index entry has a lifetime and will be removed from the Index if it is not refreshed before it expires.

Each Globus container that has WS MDS installed will automatically have a default Index Service instance. By default, any GRAM, RFT, or CAS service running in that container will register itself to the container’s default Index Service.

## 2. Trigger Service

The Trigger Service collects information and compares that data against a set of conditions defined in a configuration file. When a condition is met, or triggered, an action takes place, such as emailing a system administrator when the disk space on a server reaches a threshold.

# Chapter 2. WS MDS Information Interfaces and Providers

In addition to the services described above, WS MDS includes several additional software components, including an *Aggregator Framework*, which provides a unified mechanism used by the Index and Trigger services to collect data.

## 1. Aggregator Framework

The Aggregator Framework is a software framework used to build services that collect and aggregate data. Services (such as the Index and Trigger services) built on the Aggregator Framework are sometimes called *aggregator services*, and have the following in common:

- They collect information via *Aggregator Sources*. An Aggregator Source is a Java class that implements an interface (defined as part of the Aggregator Framework) to collect XML-formatted data.
- They use a common configuration mechanism to maintain information about which Aggregator Source to use and its associated parameters (which generally specify what data to get, and from where). The Aggregator Framework WSDL defines an [aggregating service group entry type] that holds both configuration information and data. Administrative client programs use standard [WSRF Service Group registration mechanisms] to register these service group entries to the *aggregator service*.
- They are self-cleaning – each registration has a lifetime; if a registration expires without being refreshed, it and its associated data are removed from the server.

### 1.1. Aggregator Sources

WS MDS includes the following three Aggregator Sources:

- the *Query Aggregator Source*, which polls a WSRF service for resource property information,
- the *Subscription Aggregator Source*, which collect data from a WSRF service via WSRF subscription/notification,
- the *Execution Aggregator Source*, which executes an administrator-supplied program to collect information.

Note: The query and subscription sources are most useful with web-service-based resources, whereas the execution source is useful for non-web-service-based resources.

[graphic with flow of info?]

Note: All aggregator sources listed in this table are in the `org.globus.mds.aggregator.impl` package, so for example the aggregator source listed as `QueryAggregatorSource` is actually `org.globus.mds.aggregator.impl.QueryAggregatorSource`.

#### 1.1.1. Query Source

The query source collects information from a registered resource by using WS-Resource Properties polling mechanisms:

- `GetResourcePropertyPollType`: requests a single Resource Property from the Grid resource.
- `GetMultipleResourcePropertiesPollType`: requests multiple Resource Properties from the Grid resource.

- `QueryResourcePropertiesPollType`: requests a query be executed against the Resource Property Set of the Grid resource.

The query source will attempt to detect when the data source EPR is local to the current container instance, and if so, set the connection properties to use local transport.

Polls are made periodically, with both the period and target Resource Properties specified in the registration message.

### 1.1.2. Subscription Source

The subscription source gathers resource property values from the registered resource using WS-Notification subscriptions. Data is delivered when property values change, rather than periodically.

The subscription source will attempt to detect when the data source EPR is local to the current container instance, and, if so, set the connection properties to use local transport.

### 1.1.3. Execution Source

The execution source provides a way to aggregate data (arbitrary XML information) about a registered resource using an arbitrary local executable (such as an external script). The executable will be passed registration information as parameters and is expected to output the gathered data.

A basic example of the use of this API is described in the [ping test example] for the aggregator execution source.

The execution aggregation source will periodically execute an identified executable. The identity of the executable and the frequency with which it is to run are specified in the registration message.

Details of the interface between the execution source and local executables are in [Configuring Execution Aggregator Source](#).

## 2. Information Providers

Depending on the implementation, an Aggregator Source may use an external software component (for example, the Execution Aggregator Source uses an executable program), or a WSRF service may use an external component to create and update its resource properties (which may then be registered to an Index or other aggregator service, using the Query or Subscription Aggregator Source). We refer to this set of components as *Information Providers*.

Currently, WS MDS includes the following sources of information:

- *Hawkeye Information Provider*: An Information Provider that gathers *Hawkeye* data about Condor pool resources using the XML mapping of the GLUE schema and reports it to a GRAM4 service, which publishes it as resource properties. This information includes:
  - basic host data (name, ID)
  - processor information
  - memory size
  - OS name and version
  - file system data
  - processor load data

- 
- other basic Condor host data.
  - *Ganglia Information Provider*: An Information Provider that gathers cluster data from resources running *Ganglia* using the XML mapping of the GLUE schema and reports it to a GRAM4 service, which publishes it as resource properties. This information includes:
    - basic host data (name, ID)
    - memory size
    - OS name and version
    - file system data
    - processor load data
    - other basic cluster data.
  - *GRAM4*: The job submission service component of GT4. This WSRF service publishes information about the local scheduler, including:
    - queue information
    - number of CPUs available and free
    - job count information
    - some memory statistics.
  - *Reliable File Transfer Service (RFT)*: The file transfer service component of GT4. This WSRF service publishes:
    - status data of the server
    - transfer status for a file or set of files
    - number of active transfers
    - some status information about the resource running the service.
  - *Community Authorization Service (CAS)* This WSRF services publishes information identifying the VO that it serves.
  - Any other WSRF service that publishes resource properties. Note: In addition to any other resource properties, GT4 services publish a basic `ServiceMetaDataInfo` element that includes start time, version, and service type name.

### 3. WebMDS user interface

WebMDS is a web-based interface to WSRF resource property information that is available as a user-friendly front-end to the Index Service. WebMDS uses standard resource property requests to query resource property data and displays the results in various formats.

Figure 2.1. Table view in WebMDS

**ServiceGroup Overview**

This page provides a brief overview of Web Services and/or WS-Resources that are members of a WS-ServiceGroup.

This WS-ServiceGroup has 3 direct entries, 10 in whole hierarchy.

Resource Type	ID	Information
CAS	127.0.0.1	Serving VO: Description of VO of CAS service <a href="#">detail</a>
ServiceGroup	128.9.72.106	This WS-ServiceGroup has 3 direct entries, 3 including descendants. <a href="#">detail</a>
GRAN	128.9.72.106	0 queues, submitting to 0 cluster(s) of 0 host(s). <a href="#">detail</a>
GRAN	128.9.72.106	1 queues, submitting to 0 cluster(s) of 0 host(s). <a href="#">detail</a>
RFT	128.9.72.106	0 active transfer resources, transferring 0 files. 5.5 GB transferred in 3970 files since start of database. <a href="#">detail</a>
ServiceGroup	128.9.72.178	This WS-ServiceGroup has 4 direct entries, 4 including descendants. <a href="#">detail</a>
GRAN	128.9.72.178	0 queues, submitting to 1 cluster(s) of 10 host(s). <a href="#">detail</a>
RFT	128.9.72.178	0 active transfer resources, transferring 0 files. 8.42 GB transferred in 6433 files since start of database. <a href="#">detail</a>
GRAN	128.9.72.178	2 queues, submitting to 1 cluster(s) of 10 host(s). <a href="#">detail</a>
GRAN	128.9.72.178	1 queues, submitting to 1 cluster(s) of 10 host(s). <a href="#">detail</a>

Please report bugs and feature requests into the [Issue Tracker](#).

XSLT transformation provided by servicegrouptable.xsl version 2.5.

Figure 2.2. Detail view in WebMDS

**Service Group Entry Detail**

Service Group EPR

- Address: <https://128.9.72.106:9000/wsr/services/DefaultIndexServiceEntry>
- GroupKey: 3747816
- EntryKey: 23565920

Member Service EPR

- Address: <https://128.9.72.106:9000/wsr/services/ReliableFileTransferFactoryService>

Content

- AggregatorConfig:
  - GetMultipleResourcePropertiesPollType:
    - PollIntervalMills: 60000
    - ResourcePropertyNames: rt.TotalNumberBytesTransferred
    - ResourcePropertyNames: rt.TotalNumberActiveTransfers
    - ResourcePropertyNames: rt.RFTFactoryStartTime
    - ResourcePropertyNames: rt.ActiveResourceInstances
    - ResourcePropertyNames: rt.TotalNumberTransfers
- AggregatorData:
  - TotalNumberBytesTransferred: 5906528711
  - TotalNumberActiveTransfers: 0
  - RFTFactoryStartTime: 2005-04-02T20:00:10.475Z
  - ActiveResourceInstances: 0

## Chapter 3. Putting it all together

A typical deployment of WS MDS will enable a project to:

- discover needed data from services in order to make job submission or replica selection decisions;
- evaluate the status of Grid services for a project testbed;
- be notified when disks are full or other error conditions happen;
- visualize the state of services.

MDS should be deployed in a distributed fashion. Some components should be deployed central to a VO, while others should be deployed on individual resources.

In order to deploy a project or VO-wide WS MDS setup, we recommend the following steps.

*Note:* The services deployed do *not* need to be run on the same host or be at the same location.

1. For clusters (or Condor pools), make sure that Ganglia (or Hawkeye) is configured and running properly to view cluster information in the Index Service. Please see [Cluster Monitoring Information and the GLUE Resource Property](#)<sup>1</sup> for more information. Once properly deployed, you can view the scheduler and cluster information, whose format is defined by XML schema definitions, located at `$GLOBUS_LOCA-TION/share/schema/mds/usefulrp/batchproviders.xsd` and `$GLOBUS_LOCA-TION/share/schema/mds/usefulrp/ce.xsd`.
2. Set up the Index Service:
  - a. If you want to set up a site-wide Index Service with all services and resources for the project at that site registered to it, including those provided by Ganglia or Hawkeye, please refer to [System Administrator's Guide](#).
  - b. If you want to deploy an Index Service to act as the centralized data source for the VO to collect information about all of the resources in the VO, please see [Deploying WS MDS in a Virtual Organization](#)<sup>2</sup>.
3. Install the WebMDS release to view the contents of a central Index Service in a web browser (please see [System Administrator's Guide](#) for more information).
4. Deploy a Trigger Service notification script to alert interested parties about changes in the status of the VO (please see [System Administrator's Guide](#)).

---

<sup>1</sup> [gluerrp.html](#)

<sup>2</sup> [deployment\\_overview.html](#)

# Glossary

## A

aggregator services Services that are built on the Aggregator Framework, such as the WS MDS Index Service and Trigger Service.

aggregator source A Java class that implements an interface (defined as part of the Aggregator Framework) to collect XML-formatted data. WS MDS contains three aggregator sources: the query aggregator source, the subscription aggregator source, and the execution aggregator source.

## E

execution aggregator source An Aggregator Source (included in WS MDS) that executes an administrator-supplied program to collect information and make it available to an Aggregator Service such as the Index Service.

## G

Ganglia A cluster monitoring tool (re: WS MDS). See <http://ganglia.sourceforge.net>.

## H

Hawkeye A monitoring service for Condor Pools. See <http://www.cs.wisc.edu/condor/hawkeye/>.