# GT 4.0: Information Services: Index

# GT 4.0: Information Services: Index

# Table of Contents

# List of Tables

# Docs that relate to all MDS components: Key Concepts, Migrating Guide and Samples

- Key Concepts[1]
- Migrating Guide[2]
- Samples[3]

[1] ../key-index.html
[2] ../WS_MDS_Migrating_Guide.html
[3] ../WS_MDS_Samples.html

# Chapter 1. GT 4.0 Release Notes: Index Service

## 1. Component Overview

The Index Service collects monitoring and discovery information from Grid resources, and publishes it in a single location; generally, it is expected that a virtual organization will deploy one or more index services which will collect data on all of the Grid resources available within that virtual organization.

## 2. Feature Summary

Features new in release 4.0

- Based on WSRF rather than OGSI
  - Aggregated data is published through WS-Resource Properties mechanisms
  - The *aggregator framework* module (basis of the index service) collects data from monitored resources using WS-Resource Properties collection mechanisms, including the base WS-Resource Properties poll operations and WS-Notification.
- Persistent configuration of aggregations has been refactored. The service no longer has its own config file for specifying aggregations. Instead a separate client is used to make appropriate registrations. This client may be started alongside the container for the equivalent effect. The client may also be deployed elsewhere to support resource-side configuration (providing similar functionality to the GT3.2 RegistryPublishProvider) or at a third location.

Other Supported Features

- The index appears as a ServiceGroup (defined in WSRF) which lists registered resources alongside dynamically collected information from those resources. This information can be examined using (for example) XPath queries to discover resources that match desired constraints.

Deprecated Features

- Support for Xindice and aggregated data persistence has been removed.
- The ability to specify default aggregations in service config file has been removed (see item in 'New Features').
- Non-ServiceGroupRegistration methods for managing aggregations have been removed; the equivalent functionality is available through the ServiceGroupRegistration port type which is available in both GT3.2 and GT 3.9.3.0 versions of the index service.

## 3. Bug Fixes

- Bug 2667: MDS index downstream registrations uses upstream config[1]
- Bug 2409: Index needing user proxy after TLS merge[2]

[1] http://bugzilla.globus.org/globus/show_bug.cgi?id=2667
[2] http://bugzilla.globus.org/globus/show_bug.cgi?id=2409

- Bug 2131: default aggregator service entry EPRs point to the wrong service[3]
- Bug 2104: flatten schema[4]
- Bug 1885: Relative path to ganglia_to_glue.xslt in GangliaGLUE service[5]
- 2141: fixable warn at index startup[6]
- 2925: No deserializer for AggregatorContent[7]
- All fixed Index Service bugs and enhancement requests.[8]

## 4. Known Problems

- All open Index Service bugs and enhancement requests[9]

## 5. Technology Dependencies

The Index Service depends on the following GT components:

- Java WS Core[10]
- WS MDS Aggregator Framework[11]

The Index Service depends on the following 3rd party software:

- None

## 6. Tested Platforms

Tested Platforms for WS-MDS Index Service:

- Linux on i386
- Windows XP

Tested containers for WS-MDS Index Service:

- Java WS Core container

[3] http://bugzilla.globus.org/globus/show_bug.cgi?id=2131
[4] http://bugzilla.globus.org/globus/show_bug.cgi?id=2104
[5] http://bugzilla.globus.org/globus/show_bug.cgi?id=1885
[6] http://bugzilla.globus.org/globus/show_bug.cgi?id=2141
[7] http://bugzilla.globus.org/globus/show_bug.cgi?id=2925
[8] http://bugzilla.globus.org/globus/buglist.cgi?short_desc_type=allwordssubstr&short_desc=&product=MDS&component=wsrf_index&long_desc_type=allwordssubstr&long_desc=&bug_file_loc_type=allwordssubstr&bug_file_loc=&bug_status=RESOLVED&bug_status=VERIFIED&bug_status=CLOSED&emailtype1=substring&email1=&emailtype2=substring&email2=&bugidtype=include&bug_id=&votes=&changedin=&chfieldfrom=&chfieldto=Now&chfieldvalue=&cmdtype=doit&newqueryname=&order=Reuse+same+sort+as+last+time&field0-0-0=noop&type0-0-0=noop&value0-0-0=
[9] http://bugzilla.globus.org/globus/buglist.cgi?short_desc_type=allwordssubstr&short_desc=&product=MDS&component=wsrf_indexr&long_desc_type=allwordssubstr&long_desc=&bug_file_loc_type=allwordssubstr&bug_file_loc=&bug_status=NEW&bug_status=ASSIGNED&bug_status=REOPENED&emailtype1=substring&email1=&emailtype2=substring&email2=&bugidtype=include&bug_id=&votes=&changedin=&chfieldfrom=&chfieldto=Now&chfieldvalue=&cmdtype=doit&newqueryname=&order=Reuse+same+sort+as+last+time&field0-0-0=noop&type0-0-0=noop&value0-0-0=
[10] http://www.globus.org/toolkit/docs/4.0/common/javawscore/
[11] http://www.globus.org/toolkit/docs/4.0/info/aggregator/

- Tomcat 5.0.28

## 7. Backward Compatibility Summary

Protocol changes since GT version 3.2

- Generally incompatible with the GT3.2 index service as the service has been remodelled to use WSRF instead of OGSI.

API changes since GT version 3.2

- The aggregator framework API used internally has retained the general flavor of the GT3.2 aggregator framework API, but is not directly compatible with it.

Schema changes since GT version 3.2

- Schemas used for configuration and publishing are conceptually similar but are not compatible, primarily due to WSRF remodelling.

## 8. For More Information

Click here[12] for more information about this component.

[12] index.html

# Chapter 2. GT 4.0.1 Incremental Release Notes: Index Service

## 1. Introduction

These release notes are for the incremental release 4.0.1. It includes a summary of changes since 4.0.0, bug fixes since 4.0.0 and any known problems that still exist at the time of the 4.0.1 release. This page is in addition to the top-level 4.0.1 release notes at http://www.globus.org/toolkit/releasenotes/4.0.1.

For release notes about 4.0 (including feature summary, technology dependencies, etc) go to the WS MDS Index 4.0 Release Notes[1].

## 2. Changes Summary

Other than bug fixes, no changes have occurred for WS MDS Index service.

## 3. Bug Fixes

The following bug was fixed for the WS MDS Index Service:

- Bug 3570:[2] wsdl error while extending IndexPortType

## 4. Known Problems

No known problems exist for the WS MDS Index service at the time of the 4.0.1 release

## 5. For More Information

Click here[3] for more information about this component.

[1] http://www.globus.org/toolkit/docs/4.0/info/index/WS_MDS_Index_Release_Notes.html
[2] http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=3570
[3] index.html

# Chapter 3. GT 4.0.2 Incremental Release Notes: Index Service

## 1. Introduction

These release notes are for the incremental release 4.0.2. It includes a summary of changes since 4.0.1, bug fixes since 4.0.1 and any known problems that still exist at the time of the 4.0.2 release. This page is in addition to the top-level 4.0.2 release notes at http://www.globus.org/toolkit/releasenotes/4.0.2.

For release notes about 4.0 (including feature summary, technology dependencies, etc) go to the WS MDS Index 4.0 Release Notes[1].

## 2. Changes Summary

No changes have occurred for WS MDS Index service.

## 3. Bug Fixes

No new bugs were fixed for the WS MDS Index Service

## 4. Known Problems

No known problems exist for the WS MDS Index service at the time of the 4.0.2 release.

## 5. For More Information

Click here[2] for more information about this component.

[1] http://www.globus.org/toolkit/docs/4.0/info/index/WS_MDS_Index_Release_Notes.html
[2] index.html

# Chapter 4. GT 4.0.3 Incremental Release Notes: Index Service

## 1. Introduction

These release notes are for the incremental release 4.0.3. It includes a summary of changes since 4.0.2, bug fixes since 4.0.2 and any known problems that still exist at the time of the 4.0.3 release. This page is in addition to the top-level 4.0.3 release notes at http://www.globus.org/toolkit/releasenotes/4.0.3.

For release notes about 4.0 (including feature summary, technology dependencies, etc) go to the WS MDS Index 4.0 Release Notes[1].

## 2. Changes Summary

No changes have occurred for WS MDS Index service since GT 4.0.2.

## 3. Bug Fixes

No new bugs have been fixed for the WS MDS Index Service since GT 4.0.2.

## 4. Known Problems

No known problems exist for the WS MDS Index service at the time of the 4.0.3 release.

## 5. For More Information

Click here[2] for more information about this component.

[1] http://www.globus.org/toolkit/docs/4.0/info/index/WS_MDS_Index_Release_Notes.html
[2] index.html

# Chapter 5. GT 4.0.4 Incremental Release Notes: Index Service

## 1. Introduction

These release notes are for the incremental release 4.0.4. It includes a summary of changes since 4.0.3, bug fixes since 4.0.3 and any known problems that still exist at the time of the 4.0.4 release. This page is in addition to the top-level 4.0.4 release notes at http://www.globus.org/toolkit/releasenotes/4.0.4.

For release notes about 4.0 (including feature summary, technology dependencies, etc) go to the WS MDS Index 4.0 Release Notes[1].

## 2. Changes Summary

No changes have occurred for WS MDS Index service since GT 4.0.3.

## 3. Bug Fixes

- Bug 4803:[2] MDS stub classes are replicated in a number of jar files

## 4. Known Problems

No known problems exist for the WS MDS Index service at the time of the 4.0.4 release.

## 5. For More Information

Click here[3] for more information about this component.

[1] http://www.globus.org/toolkit/docs/4.0/info/index/WS_MDS_Index_Release_Notes.html
[2] http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=4803
[3] index.html

# Chapter 6. GT 4.0 Index Service: System Administrator's Guide

## 1. Introduction

This guide contains advanced configuration information for system administrators working with the WS MDS Index Service. It provides references to information on procedures typically performed by system administrators, including installation, configuring, deploying, and testing the installation.

### Important

This information is in addition to the basic Globus Toolkit prerequisite, overview, installation, security configuration instructions in the GT 4.0 System Administrator's Guide[1]. Read through this guide before continuing!

## 2. Building and Installing

The Index Service is built and installed as part of a default GT 4.0 installation. For basic installation instructions, see the GT 4.0 System Administrator's Guide[2]. No extra installation steps are required for this component.

## 3. Configuring

### 3.1. Configuration overview

For a basic installation, the index service itself does not need any configuration changes from default; a default Index Service is available and automatically "registers" with the following GT web services based resources to allow monitoring and discovery: CAS, RFT[3], and WS GRAM (click the links for information about what data is sent and how to change it).

#### Note

Auto-registration is turned on by default in GT 4.0.1. If you are using GT 4.0.0, we strongly encourage you to update to GT 4.0.1. However, if you must use 4.0.0, this capability can be turned on. See the per service links above for more information.

In order for information to appear in the index, the source of that information must be registered to the Index Service. Information sources are registered using tools like **mds-servicegroup-add**[4]. Each registration has a limited lifetime; **mds-servicegroup-add** should be left running in the background so that it can continue to refresh registrations. Depending on administration preference, it may be run on the same host as the index, on the same host as a member resource, or on any other host(s).

The Index service is built on the WS MDS Aggregator Framework[5] and can use any aggregator source[6] to collect information. In the most common case, the index service uses the `QueryAggregatorSource` to gather resource

[1] http://www.globus.org/toolkit/docs/4.0/admin/docbook/
[2] http://www.globus.org/toolkit/docs/4.0/admin/docbook/
[3] http://www.globus.org/toolkit/docs/4.0/data/rft/admin-index.html#s-rft-admin-autoregistration
[4] http://www.globus.org/toolkit/docs/4.0/info/aggregator/rn01re02.html
[5] http://www.globus.org/toolkit/docs/4.0/info/aggregator/
[6] http://www.globus.org/toolkit/docs/4.0/info/aggregator/WS_MDS_Aggregator_Sources_Reference.html

property values from the registered resource using one of the three WS-Resource Properties operations to poll for information; the polling method used depends on the configuration element supplied in the registration content.

Two other aggregator sources are supplied with the distribution: the `SubscriptionAggregatorSource`, which gathers resource property values through subscription/notification, and the `ExecutionAggregatorSource`, which executes an external program to gather information.

## 3.2. Syntax of the interface

### 3.2.1. Defining the Aggregator Sources

The aggregation source used to collect data can be changed from default by editing the aggregatorSources parameter in the JNDI service configuration. See `$GLOBUS_LOCATION/etc/globus_wsrf_mds_index/jndi-config.xml`:

```
<resource name="configuration"
             type="org.globus.mds.aggregator.impl.AggregatorConfiguration">
  <resourceParams>
    <parameter>
      <name> factory</name>
      <value>org.globus.wsrf.jndi.BeanFactory</value>
    </parameter>
    <parameter>
      <name>aggregatorSource</name>
      <value>org.globus.mds.aggregator.impl.QueryAggregatorSource
             org.globus.mds.aggregator.impl.SubscriptionAggregatorSource
             org.globus.mds.aggregator.impl.ExecutionAggregatorSource
      </value>
    </parameter>
</resourceParams>
```

This parameter specifies one or more Java classes that may be used to collect data for the Index. By default it is set to use the QueryAggregatorSource, SubscriptionAggregatorSource, and ExecutionAggregatorSource. Details of the supplied sources are in the Aggregator Sources Reference[7].

## 3.3. Configuring the Aggregator Framework

The aggregator framework does not have its own service side configuration, although services which are based on the framework have their own service side configuration options (such as MDS Index[8] and MDS Trigger[9]) which are documented in the per-service documentation.

Registrations to a working aggregator framework are configured for the mds-servicegroup-add[10] tool. This tool takes an XML configuration file listing registrations, and causes those registrations to be made.

In general, configuration of aggregator services involves configuring the service to get information from one or more sources in a Grid. The mechanism for doing this is defined by (inherited from) the aggregator framework and described in this section.

[7] http://www.globus.org/toolkit/docs/4.0/info/aggregator/WS_MDS_Aggregator_Sources_Reference.html#aggregator-aggregator_sources-configlist
[8] http://www.globus.org/toolkit/docs/4.0/info/index/
[9] http://www.globus.org/toolkit/docs/4.0/info/trigger/
[10] http://www.globus.org/toolkit/docs/4.0/info/aggregator/re01.html

### 3.3.1. Configuration overview

Configuring an Aggregating Service Group to perform a data aggregation is performed by specifying an AggregatorContent object as the content parameter of a ServiceGroup add method invocation. An AggregatorContent object is composed of two xsd:any arrays: AggregatorConfig and AggregatorData:

- The AggregatorConfig xsd:any array is used to specify parameters that are to be passed to the underlying AggregatorSource when the ServiceGroup *add* method is invoked. These parameters are generally type-specific to the implementation of the AggregatorSource and/or AggregatorSink being used.
- The AggregatorData xsd:any array is used as the storage location for aggregated data that is the result of message deliveries to the AggregatorSink. Generally, the AggregatorData parameter of the AggregatorContent is not populated when the ServiceGroup add method is invoked, but rather is populated by message delivery from the AggregatorSource.

### 3.3.2. Syntax of the interface

#### 3.3.2.1. Configuring the Aggregator Sources

The following links provide information for configuring the three types of aggregator sources provided by the Globus Toolkit:

- Configuring the Execution Aggregator Source[11]
- Configuring the Query Aggregator Source[12]
- Configuring the Subscription Aggregator Source[13]

#### 3.3.2.2. Configuring the Aggregator Sink

An aggregator sink may require sink-specific configuration (the MDS *Trigger service* requires sink-specific configuration; the MDS *Index service* does not). See the documentation for the specific *aggregator service* being used for details on sink-specific documentation.

## 3.4. Changing aggregator source

The aggregation source used to collect data can be changed from default, as detailed in the Defining the Aggregator Sources.

# 4. Deploying

The Index service is deployed into the Globus container by default during the standard toolkit installation[14].

## 4.1. Deploying into Tomcat

The MDS4 Index service has been tested to work without any additional setup when deployed into Tomcat. Please follow these basic instructions[15] to deploy GT4 services into Tomcat. Note: please complete any prerequisite service configuration steps before you deploy into Tomcat.

[11] http://www.globus.org/toolkit/docs/4.0/info/aggregator/Execution_Aggregator_Source.html
[12] http://www.globus.org/toolkit/docs/4.0/info/aggregator/Query_Aggregator_Source.html
[13] http://www.globus.org/toolkit/docs/4.0/info/aggregator/Subscription_Aggregator_Source.html
[14] http://www.globus.org/toolkit/docs/4.0/admin/docbook/
[15] http://www.globus.org/toolkit/docs/4.0/common/javawscore/admin-index.html#javawscore-admin-tomcat-deploying

## 5. Testing

The entire content of the default index service in a deployment can be seen by executing the following command, which will dump the entire RP set of the service:

```
wsrf-query -a -z none -s https://127.0.0.1:8443/wsrf/services/DefaultIndexService/
```

## 6. Security Considerations

The security considerations for the Aggregator Framework[16] also apply to the Index Service:

By default, the *aggregator sources* do not use authentication credentials -- they retrieve information using anonymous SSL authentication or no authentication at all, and thus retrieve only publicly-available information. If a user or administrator changes that configuration so that a service's aggregator source uses credentials to acquire non-privileged data, then that user or administrator must configure the service's aggregator sink to limit access to authorized users.

## 7. Troubleshooting

*Problem: An index service entry has AggregatorConfig data but an empty AggregatorData entry.*

Solution: There is probably something wrong with the registration. For example, a registration that uses QueryAggregatorSource *aggregator source* may have incorrect values for the resource's hostname or port number or a misspelled resource property name, or the remote resource may impose security restrictions that prevent the queries from the index from working. You can use the standard toolkit resource property query tools (such as **wsrf-get-properties**[17]) to verify that the remote resource is responding.

[16] http://www.globus.org/toolkit/docs/4.0/info/aggregator
[17] http://www.globus.org/toolkit/docs/4.0/common/javawscore/rn01re09.html

# Chapter 7. GT 4.0 Index Service: User's Guide

## 1. Introduction

This guide contains information for end-users of the WS MDS Index Service. The Index Service collects information about grid resources and publishes them as service group entries.

## 2. Command-line tools

The index service exposes information via service groups and is accessed using the same command-line tools used to query other WSRF services for information. These tools are part of the Java WS Core[1].

- **wsrf-query**[2] performs a query on a resource property document
- **wsrf-get-property**[3] gets a single resource property
- **wsrf-get-properties**[4] gets multiple resource properties.

## 3. Graphical user interfaces

There is no GUI specifically for the Index Service. The release contains WebMDS[5] which can be used to display monitoring information collected in an Index Service in a normal web browser.

## 4. Troubleshooting

General troubleshooting information can be found in the GT 4.0 Java WS Core User's Guide[6].

[1] http://www.globus.org/toolkit/docs/4.0/common/javawscore/
[2] http://www.globus.org/toolkit/docs/4.0/common/javawscore/rn01re07.html
[3] http://www.globus.org/toolkit/docs/4.0/common/javawscore/rn01re08.html
[4] http://www.globus.org/toolkit/docs/4.0/common/javawscore/rn01re09.html
[5] http://www.globus.org/toolkit/docs/4.0/info/webmds/index.html
[6] ../../common/javawscore/user-index.html#troubleshooting

# Chapter 8. GT 4.0 Index Service: Developer's Guide

## 1. Introduction

The WS MDS Index Service collects information about grid resources and publishes that information as a service group. Client programs use resource property queries or subscription/notification to retrieve information from the index. Information can be added to the index via a number of different mechanisms: since the Index Service is implemented using the *Aggregator Framework*, any *aggregator source* can be used to provide information for the index.

This document describes the programmatic interfaces to the Index Service. See also general Globus Toolkit coding guidelines[1] and GT 4.0 best practices[2].

## 2. Before you begin

### 2.1. Feature summary

Features new in release 4.0

- Based on WSRF rather than OGSI
  - Aggregated data is published through WS-Resource Properties mechanisms
  - The *aggregator framework* module (basis of the index service) collects data from monitored resources using WS-Resource Properties collection mechanisms, including the base WS-Resource Properties poll operations and WS-Notification.
- Persistent configuration of aggregations has been refactored. The service no longer has its own config file for specifying aggregations. Instead a separate client is used to make appropriate registrations. This client may be started alongside the container for the equivalent effect. The client may also be deployed elsewhere to support resource-side configuration (providing similar functionality to the GT3.2 RegistryPublishProvider) or at a third location.

Other Supported Features

- The index appears as a ServiceGroup (defined in WSRF) which lists registered resources alongside dynamically collected information from those resources. This information can be examined using (for example) XPath queries to discover resources that match desired constraints.

Deprecated Features

- Support for Xindice and aggregated data persistence has been removed.
- The ability to specify default aggregations in service config file has been removed (see item in 'New Features').
- Non-ServiceGroupRegistration methods for managing aggregations have been removed; the equivalent functionality is available through the ServiceGroupRegistration port type which is available in both GT3.2 and GT 3.9.3.0 versions of the index service.

[1] http://www.globus.org/toolkit/docs/development/coding_guidelines.html
[2] http://www.globus.org/toolkit/docs/4.0/best_practices.html

## 2.2. Tested platforms

Tested Platforms for WS-MDS Index Service:

- Linux on i386
- Windows XP

Tested containers for WS-MDS Index Service:

- Java WS Core container
- Tomcat 5.0.28

## 2.3. Backward compatibility summary

Protocol changes since GT version 3.2

- Generally incompatible with the GT3.2 index service as the service has been remodelled to use WSRF instead of OGSI.

API changes since GT version 3.2

- The aggregator framework API used internally has retained the general flavor of the GT3.2 aggregator framework API, but is not directly compatible with it.

Schema changes since GT version 3.2

- Schemas used for configuration and publishing are conceptually similar but are not compatible, primarily due to WSRF remodelling.

## 2.4. Technology dependencies

The Index Service depends on the following GT components:

- Java WS Core[3]
- WS MDS Aggregator Framework[4]

The Index Service depends on the following 3rd party software:

- None

## 2.5. Security considerations

The security considerations for the Aggregator Framework[5] also apply to the Index Service:

By default, the *aggregator sources* do not use authentication credentials -- they retrieve information using anonymous SSL authentication or no authentication at all, and thus retrieve only publicly-available information. If a user or administrator changes that configuration so that a service's aggregator source uses credentials to acquire non-privileged data, then that user or administrator must configure the service's aggregator sink to limit access to authorized users.

[3] http://www.globus.org/toolkit/docs/4.0/common/javawscore/
[4] http://www.globus.org/toolkit/docs/4.0/info/aggregator/
[5] http://www.globus.org/toolkit/docs/4.0/info/aggregator

# 3. Architecture and design overview

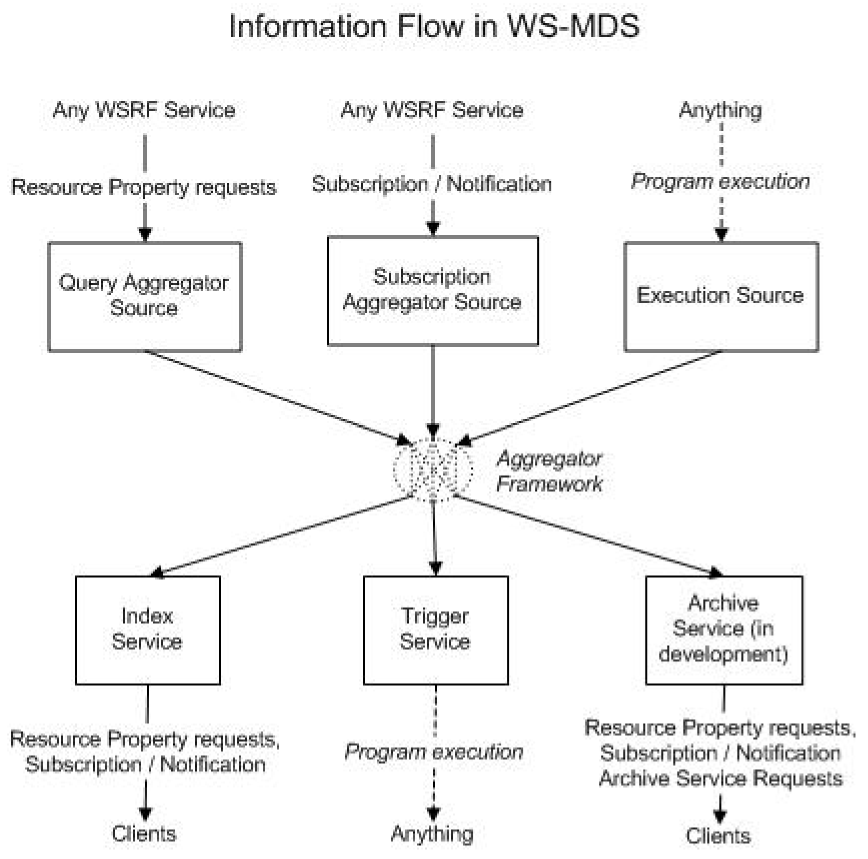
There are essentially two interfaces to the Index Service -- one for getting information into the index, and one for retrieving information from the index.

Information is retrieved from the Index Service as service group entries using the standard WS MDS Core APIs for resource property queries or subscription/notification.

Because the Index is implemented as a WS MDS Aggregator Framework, the programmatic interface for getting information into the index is to create an aggregator source. The Aggregator Framework's architecture is described in the next section.

## 3.1. Aggregator Framework architecture

The WS MDS Aggregator Framework is the software framework on which WS MDS services are built. The aggregator framework collects data from an *aggregator source* and sends that data to an *aggregator sink* for processing. Aggregator sources distributed with the Globus Toolkit include modules that query resource properties, acquire data through subscription/notification, and execute programs to generate data. Another way of describing the Aggregator Framework is that it is designed to facilitate the collecting of information from or about WS-Resources via plugin aggregator sources and the feeding of that information to plugin aggregator sinks, which can then perform actions such as republishing, logging, or archiving the information.


Information Flow in WS-MDS
Any WSRF Service
Resource Property requests
Any WSRF Service
Subscription / Notification
Anything
Program execution
Query Aggregator Source
Subscription Aggregator Source
Execution Source
Aggregator Framework
Index Service
Trigger Service
Archive Service (in development)
Resource Property requests, Subscription / Notification
Program execution
Resource Property requests, Subscription / Notification Archive Service Requests
Clients
Anything
Clients

Aggregators work on a type of service group called an **AggregatorServiceGroupRP**. Resources may be *registered* to an AggregatorServiceGroupRP using the service group `add` operation, which will cause an entry to be added to the service group. The entry will include configuration parameters for the aggregator source; when the registration is made, the appropriate aggregation source and sinks will be informed; the aggregator source will begin collecting data and inserting it into the corresponding service group entry, and the aggregator sink will begin processing the information in the service group entries.

The method of collection by source and processing by the sink is dependent on the particular instantiation of the aggregator framework.

### 3.1.1. Standard aggregator sinks

The aggregator sinks distributed with the toolkit (`org.globus.mds.aggregator.impl.ServiceGroupEntryAggregatorSink` and `org.globus.mds.trigger.impl.TriggerResource`) are described in the following table.

**Table 8.1. Standard aggregator sinks**

| **Aggregator Sink** | **Service Implemented** | **Description** |
|---|---|---|
| `ServiceGroupEntryAggregatorSink` | Index Service | The servicegroup sink (used by the *Index Service*) publishes received data as content in the AggregatingServiceGroup entry used to manage the registration. This data can therefore be retrieved by querying the index for its 'entries' resource property. |
| `TriggerResource` | Trigger Service | The *Trigger Service* provides an aggregator sink which receives data, applies tests to that data, and if the tests match, runs a specified executable. See the trigger service[6] documentation for more information. |

### 3.1.2. Standard aggregator sources

The aggregator sources supplied with the toolkit collect information using resource property queries (query sources), subscription/notification (subscription sources), and execution of external programs (execution sources).

The aggregator sources supplied with the Globus Toolkit are listed in the following table.

**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`

[6] http://www.globus.org/toolkit/docs/4.0/info/aggregator/trigger/

**Table 8.2. Standard aggregator sources**

| Aggregator Source | Description |
|---|---|
| QueryAggregatorSource | The query source collects information from a registered resource by using WS-Resource Properties polling mechanisms: • `GetResourcePropertyPollType`; requests a single Resource Property from the remote resource. • `GetMultipleResourcePropertiesPollType`; requests multiple Resource Properties from the remote resource. • `QueryResourcePropertiesPollType`; requests a query be executed against the Resource Property Set of the remote resource. Polls are made periodically, with both the period and target Resource Properties specified in the registration message. |
| SubscriptionAggregator-Source | The subscription source collects information from a registered resource using WS-Notification mechanisms. Data is delivered when property values change, rather than periodically. |
| ExecutionAggregator-Source | The execution source collects information about (not necessarily from) a registered resource by execution of a local executable, which is passed as input the identity of the registered resource. Details of the interface between the execution source and local executables are in Configuring the Execution Aggregator Source[7]. |

## 4. Public interface

The semantics and syntax of the APIs and WSDL for the component, along with descriptions of domain-specific structured interface data, can be found in the public interface guide[8].

## 5. Usage scenarios

### 5.1. Retrieving information from an index service

Information is retrieved from the index using the standard Java WS Core API calls for getting resource property information:

- `GetResourceProperty` to request a single resource property by name,
- `GetResourceProperties` to request several resource properties by name,
- `QueryResourceProperty` to perform an XPath query on a resource property document, and
- the notification/subscription mechanism.

See the Java WS Core Developer's Guide[9] for API details.

[7] http://www.globus.org/toolkit/docs/4.0/info/aggregator/Execution_Aggregator_Source.html
[8] WS_MDS_Index_Public_Interfaces.html
[9] ../../common/javawscore/developer-index.html

### 5.2. Adding information to an index

Information is added to an index by way of an *aggregator source*.

The Globus Toolkit distribution includes several standard aggregator sources (see the Aggregator Sources Reference[10] for more details).

To create your own custom information source, see the WS MDS Aggregator Framework Developer's Guide[11].

## 6. Tutorials

Use of the index service is covered in the Build a Grid Service Tutorial (GlobusWORLD 2005)[12].

## 7. Debugging

See the Debug section of the Java WS Core Developer's Guide[13] for general information on logging, including which files to edit to set logging properties.

To turn on debug logging for the Index Service, add the line:

```
log4j.category.org.globus.mds.index=DEBUG
```

to the appropriate properties file. Since the Index Service is implemented using the aggregator framework, you may also want to turn on aggregator debugging by adding this line:

```
log4j.category.org.globus.mds.aggregator=DEBUG
```

## 8. Troubleshooting

General troubleshooting information can be found in the GT 4.0 Java WS Core Developer's Guide[14].

## 9. Related Documentation

Specifications for resource properties, service groups, and subscription/notification are available at http://www.globus.org/wsrf/.

[10] http://www.globus.org/toolkit/docs/4.0/info/aggregator/WS_MDS_Aggregator_Sources_Reference.html
[11] ../aggregator/developer-index.html
[12] http://www.globus.org/toolkit/tutorials/BAS/
[13] ../../common/javawscore/developer-index.html#debugging
[14] ../../common/javawscore/developer-index.html#troubleshooting

# Chapter 9. GT 4.0 Index Service: Fact Sheet

## 1. Brief component overview

The Index Service collects monitoring and discovery information from Grid resources, and publishes it in a single location; generally, it is expected that a virtual organization will deploy one or more index services which will collect data on all of the Grid resources available within that virtual organization.

## 2. Summary of features

Features new in release 4.0

- Based on WSRF rather than OGSI
  - Aggregated data is published through WS-Resource Properties mechanisms
  - The *aggregator framework* module (basis of the index service) collects data from monitored resources using WS-Resource Properties collection mechanisms, including the base WS-Resource Properties poll operations and WS-Notification.
- Persistent configuration of aggregations has been refactored. The service no longer has its own config file for specifying aggregations. Instead a separate client is used to make appropriate registrations. This client may be started alongside the container for the equivalent effect. The client may also be deployed elsewhere to support resource-side configuration (providing similar functionality to the GT3.2 RegistryPublishProvider) or at a third location.

Other Supported Features

- The index appears as a ServiceGroup (defined in WSRF) which lists registered resources alongside dynamically collected information from those resources. This information can be examined using (for example) XPath queries to discover resources that match desired constraints.

Deprecated Features

- Support for Xindice and aggregated data persistence has been removed.
- The ability to specify default aggregations in service config file has been removed (see item in 'New Features').
- Non-ServiceGroupRegistration methods for managing aggregations have been removed; the equivalent functionality is available through the ServiceGroupRegistration port type which is available in both GT3.2 and GT 3.9.3.0 versions of the index service.

## 3. Usability summary

Usability improvements for WS MDS Index:

- Auto-registration of select GT4 services: WS-GRAM, WS-RFT, and WS-CAS now automatically register to the DefaultIndexService in a GT4 container at container startup, and select resource properties from those services are aggregated (cached) in the DefaultIndexService "Entry" resource property.

- Simplified hierarchical VO building using DefaultIndexServices: The DefaultIndexService is capable of automatically making upstream or downstream registrations at startup by processing the `$GLOBUS_LOCATION/etc/globus_wsrf_mds_index/hierarchy.xml` file.
- **mds-servicegroup-add** command: The **mds-servicegroup-add** command simplifies the process of registration to WSRF Service Groups via a command line interface.

## 4. Backward compatibility summary

Protocol changes since GT version 3.2

- Generally incompatible with the GT3.2 index service as the service has been remodelled to use WSRF instead of OGSI.

API changes since GT version 3.2

- The aggregator framework API used internally has retained the general flavor of the GT3.2 aggregator framework API, but is not directly compatible with it.

Schema changes since GT version 3.2

- Schemas used for configuration and publishing are conceptually similar but are not compatible, primarily due to WSRF remodelling.

## 5. Technology dependencies

The Index Service depends on the following GT components:

- Java WS Core[1]
- WS MDS Aggregator Framework[2]

The Index Service depends on the following 3rd party software:

- None

## 6. Tested platforms

Tested Platforms for WS-MDS Index Service:

- Linux on i386
- Windows XP

Tested containers for WS-MDS Index Service:

- Java WS Core container
- Tomcat 5.0.28

[1] http://www.globus.org/toolkit/docs/4.0/common/javawscore/
[2] http://www.globus.org/toolkit/docs/4.0/info/aggregator/

## 7. Associated standards

Associated standards for WS MDS Index:

- WS-ResourceProperties (WSRF-RP)
- WS-ResourceLifetime (WSRF-RL)
- WS-ServiceGroup (WSRF-SG)
- WS-BaseFaults (WSRF-BF)
- WS-BaseNotification
- WS-Topics

## 8. For More Information

Click here[3] for more information about this component.

[3] index.html

# Chapter 10. GT 4.0 Index Service: Public Interface Guide

## 1. Semantics and syntax of APIs

### 1.1. Programming Model Overview

Index service queries are performed using resource property requests; consult the Java WS Core documentation[1] for details.

The contents of an index are maintained using the aggregator framework programming model, and can receive data from any *aggregator source*. Information about how to configure existing aggregator sources (such as the aggregator sources distributed with the Globus Toolkit, which include one that polls for resource property information, one that collects resource property information through subscription/notification, and one that collects information by executing an executable program) is found in the Aggregator Sources Reference[2]; information about how to create new aggregator sources can be found in the WS MDS Aggregator Framework Developer's Guide[3].

## 2. Semantics and syntax of the WSDL

The index service inherits its WSDL interface from the *aggregator framework* module, included below:

### 2.1. Protocol overview

The aggregator framework builds on the WS-ServiceGroup[4] and WS-ResourceLifetime[5] specifications. Those specifications should be consulted for details on the syntax of each operation.

Each aggregator framework is represented as a WS-ServiceGroup (specifically, an AggregatorServiceGroup).

Resources may be registered to an AggregatorServiceGroup using the AggregatorServiceGroup Add operation. Each registration will be represented as a ServiceGroupEntry resource. Resources may be *registered* to an AggregatorServiceGroup using the service group `add` operation, which will cause an entry to be added to the service group.

The entry will include configuration parameters for the *aggregator source*; when the registration is made, the following will happen:

1. The appropriate aggregation source and sinks will be informed,
2. the aggregator source will begin collecting data and inserting it into the corresponding service group entry,
3. and the aggregator sink will begin processing the information in the service group entries.

[1] http://www.globus.org/toolkit/docs/4.0/common/javawscore
[2] http://www.globus.org/toolkit/docs/4.0/info/aggregator/WS_MDS_Aggregator_Sources_Reference.html
[3] http://www.globus.org/toolkit/docs/4.0/info/aggregator/developer-index.html
[4] http://viewcvs.globus.org/viewcvs.cgi/wsrf/schema/wsrf/servicegroup/WS-ServiceGroup.wsdl?rev=1.9&only_with_tag=globus_4_0_0
[5] http://viewcvs.globus.org/viewcvs.cgi/wsrf/schema/wsrf/lifetime/WS-ResourceLifetime.wsdl?rev=1.11&only_with_tag=globus_4_0_0

The method of collection by source and processing by the sink is dependent on the particular instantiation of the aggregator framework (see per-source documentation[6] for source information and per-service documentation for sink information - for example the Index Service[7] and the Trigger Service[8].)

## 2.2. Operations

### 2.2.1. AggregatorServiceGroup

- `add:` This operation is used to register a specified resource with the aggregator framework. In addition to the requirements made by the WS-ServiceGroup specification, the Content element of each registration must be an AggregatorContent type, with the AggregatorConfig element containing configuration information specific to each source and sink (documented in the Aggregator Administrator's Guide[9]).

### 2.2.2. AggregatorServiceGroupEntry

- `setTerminationTime:` This operation can be used to set the termination time of the registration, as detailed in WS-ResourceLifetime.

## 2.3. Resource properties

### 2.3.1. AggregatorServiceGroup Resource Properties

- `Entry:` This resource property publishes details of each registered resource, including both an EPR to the resource, the aggregator framework configuration information, and data from the sink.

- `RegistrationCount:` This resource property publishes registration load information (the total number of registrations since service startup and decaying averages)

## 2.4. Faults

The aggregator framework throws standard WS-ServiceGroup, WS-ResourceLifetime, and WS-ResourceProperties faults and does not define any new faults of its own.

## 2.5. WSDL and Schema Definition

- AggregatorServiceGroup[10]

- AggregatorServiceGroupEntry[11]

- common types used by AggregatorServiceGroup and AggregatorServiceGroupEntry[12]

Other relevant source files are the:

[6] http://www.globus.org/toolkit/docs/4.0/info/aggregator/WS_MDS_Aggregator_Sources_Reference.html
[7] http://www.globus.org/toolkit/docs/4.0/info/index/
[8] http://www.globus.org/toolkit/docs/4.0/info/trigger/
[9] http://www.globus.org/toolkit/docs/4.0/info/aggregator/admin-index.html
[10] http://viewcvs.globus.org/viewcvs.cgi/ws-mds/aggregator/schema/mds/aggregator/aggregator_service_group_port_type.wsdl?rev=1.1&only_with_tag=globus_4_0_0&content-type=text/vnd.viewcvs-markup
[11] http://viewcvs.globus.org/viewcvs.cgi/ws-mds/aggregator/schema/mds/aggregator/aggregator_service_group_entry_port_type.wsdl?rev=1.2&only_with_tag=globus_4_0_0&content-type=text/vnd.viewcvs-markup
[12] http://viewcvs.globus.org/viewcvs.cgi/ws-mds/aggregator/schema/mds/aggregator/aggregator-types.xsd?rev=1.1&only_with_tag=globus_4_0_0&content-type=text/vnd.viewcvs-markup

- WSRF service group schema[13]
- WSRF resource lifetime schema[14]
- MDS Usefulrp schema.

## 3. Command-line tools

Please see Index Command Reference.

## 4. Overview of Graphical User Interface

There is no GUI specifically for the Index Service. The release contains WebMDS[15] which can be used to display monitoring information collected in an Index Service in a normal web browser.

## 5. Semantics and syntax of domain-specific interface

### 5.1. Interface introduction

There is no domain-specific interface specific to the Index Service, however, the ExecutionAggregatorSource, which may be used by the Index Service, does have a domain-specific interface (specifically, the inputs provided to and outputs expected from the executable program).

### 5.2. Syntax of the interface

The syntax of the execution source's domain-specific interface is described in the Configuring the executable[16].

## 6. Configuration interface

Please see Section 3, “Configuring”.

## 7. Environment variable interface

There are no Index Service specific environment variables.

[13] http://viewcvs.globus.org/viewcvs.cgi/wsrf/schema/wsrf/servicegroup/WS-ServiceGroup.wsdl?rev=1.9&only_with_tag=globus_4_0_0
[14] http://viewcvs.globus.org/viewcvs.cgi/wsrf/schema/wsrf/lifetime/WS-ResourceLifetime.wsdl?rev=1.11&only_with_tag=globus_4_0_0
[15] http://www.globus.org/toolkit/docs/4.0/info/webmds/index.html
[16] http://www.globus.org/toolkit/docs/4.0/info/aggregator/Execution_Aggregator_Source.html#aggregator-execution_aggregator_source-executable

# Chapter 11. GT 4.0 Index Service: Quality Profile

## 1. Test coverage reports

- None available at this time.

## 2. Code analysis reports

- None available at this time.

## 3. Outstanding bugs

- 2723: loop detection[1]
- 3053: Entries in the upstream index sometimes get dropped because...[2]
- 3054: Client which creates a single resource and registers it ...[3]
- All open Index Service bugs and enhancement requests[4]

## 4. Bug Fixes

- Bug 2667: MDS index downstream registrations uses upstream config[5]
- Bug 2409: Index needing user proxy after TLS merge[6]
- Bug 2131: default aggregator service entry EPRs point to the wrong service[7]
- Bug 2104: flatten schema[8]
- Bug 1885: Relative path to ganglia_to_glue.xslt in GangliaGLUE service[9]
- 2141: fixable warn at index startup[10]
- 2925: No deserializer for AggregatorContent[11]

[1] http://bugzilla.globus.org/globus/show_bug.cgi?id=2723
[2] http://bugzilla.globus.org/globus/show_bug.cgi?id=3053
[3] http://bugzilla.globus.org/globus/show_bug.cgi?id=3054
[4] http://bugzilla.globus.org/globus/buglist.cgi?short_desc_type=allwordssubstr&short_desc=&product=MDS&component=wsrf_index&long_desc_type=allwordssubstr&long_desc=&bug_file_loc_type=allwordssubstr&bug_file_loc=&bug_status=NEW&bug_status=ASSIGNED&bug_status=REOPENED&emailtype1=substring&email1=&emailtype2=substring&email2=&bugidtype=include&bug_id=&votes=&chfieldfrom=&chfieldto=Now&chfieldvalue=&cmdtype=doit&newqueryname=&order=Reuse+same+sort+as+last+time&field0-0-0=noop&type0-0-0=noop&value0-0-0=
[5] http://bugzilla.globus.org/globus/show_bug.cgi?id=2667
[6] http://bugzilla.globus.org/globus/show_bug.cgi?id=2409
[7] http://bugzilla.globus.org/globus/show_bug.cgi?id=2131
[8] http://bugzilla.globus.org/globus/show_bug.cgi?id=2104
[9] http://bugzilla.globus.org/globus/show_bug.cgi?id=1885
[10] http://bugzilla.globus.org/globus/show_bug.cgi?id=2141
[11] http://bugzilla.globus.org/globus/show_bug.cgi?id=2925

- All fixed Index Service bugs and enhancement requests.[12]

## 5. Performance reports

- None available at this time.

[12] http://bugzilla.globus.org/globus/buglist.cgi?short_desc_type=allwordssubstr&short_desc=&product=MDS&component=wsrf_index&long_desc_type=allwordssubstr&long_desc=&bug_file_loc_type=allwordssubstr&bug_file_loc=&bug_status=RESOLVED&bug_status=VERIFIED&bug_status=CLOSED&emailtype1=substring&email1=&emailtype2=substring&email2=&bugidtype=include&bug_id=&votes=&changedin=&chfieldfrom=&chfieldto=Now&chfieldvalue=&cmdtype=doit&newqueryname=&order=Reuse+same+sort+as+last+time&field0-0-0=noop&type0-0-0=noop&value0-0-0=

# Chapter 12. GT 4.0 Index Service: How to Write a Simple Execution Aggregator Information Provider for MDS4

## 1. Introduction

This document is intended to be a starting guide to writing non web-service based *information providers* for the MDS4. It covers the concepts and walks through a simple example of how to get arbitrary information into the MDS4 using the Execution Aggregator Source[1]. This *Aggregator Source* is used for gathering arbitrary XML information about a registered resource by executing an external script. This is mostly useful for scenarios where you would like to publish information into the MDS4 from a *non web-service* based information source. For web-service based information sources that export known Resource Properties, it is much easier to use the Query Aggregator Source[2]. However, that source is outside the scope of this document.

This document covers writing a simple information provider that publishes fortune information at a regular interval into the MDS4's *Index Service*. This example was chosen because it is dynamic and simple, yet it illustrates all the fundamentals of this type of information provider.

## 2. Choosing (or conforming to) a Schema

The first step to getting information into the MDS4 is to decide which information you would like to have published. Since the data is in XML format, you should choose (or pick) the schema that you'd like the data to conform to. This generally means coming up with element names and types and creating some mapping of the data you're about to retrieve from your non web-service based application before putting it in to the MDS4. For this example, I'm going to choose this very simple format for the data:

```
<fortuneInformation>
   <fortuneData>
      ... here is the fortune ...
   </fortuneData>
   <fortuneDateAndTime>
      ... date and time of retrieval ...
   </fortuneDateAndTime>
   <fortuneSourceURL>
      ... the URL of where the fortune was retrieved ...
   </fortuneSourceURL>
</fortuneInformation>
```

As you can see, that format is very simple. An example output will look like this:

```
<fortuneInformation>
   <fortuneData>
     186,282 miles per second: It isn't just a good idea, it's the law!
   </fortuneData>
```

[1] http://globus.org/toolkit/docs/4.0/info/index/admin-index.html#s-aggregator-execution-source
[2] http://globus.org/toolkit/docs/4.0/info/index/admin-index.html#s-aggregator-query-source

```
   <fortuneDateAndTime>
     Thu Jul 14 18:16:01 CDT 2005
   </fortuneDateAndTime>
   <fortuneSourceURL>
     http://anduin.eldar.org/cgi-bin/fortune.pl?text_format=yes
   </fortuneSourceURL>
</fortuneInformation>
```

Once you've chosen how to represent your data in XML format, you can start thinking about how you're going to retrieve and prepare that data for publication.

## 3. The Code

The second step to getting information into the MDS4 is to write a script (or program) that gathers and formats the appropriate data. This can be C code, shell script, perl code, etc, and it doesn't matter what kind of methods it uses behind the scenes, so long as it produces well formatted XML data.

For example, if we wanted to publish a fortune into the Index Service (using the free and charitable online service located at http://anduin.eldar.org/cgi-bin/fortune.pl), we could write a simple shell script to retrieve it and format it into our chosen XML schema.

You can sample the source code for this example implementation here. It is written as a bash shell script due to its simplicity. Tested platforms include GNU/Linux only. For this script to properly publish information, you must have one (or more) of the following programs installed on the system: *wget*, *lynx*, or *fortune*. All of these programs come standard with most GNU/Linux distributions, and it's important to note that only one of them is required (i.e. not ALL are required). [ *NOTE: Windows users must have something like the cygwin*[3] *operating environment for this to work* ]

Download the code: fortune_script.sh.

This file should be saved in your *$GLOBUS_LOCATION/libexec/aggrexec* directory, although the reason will be explained in the next section.

## 4. Enabling The Provider

Now that we have the information provider written, the next step is to enable it so that we can test it. To do this you will need to do three things. First, come up with a short name (i.e. a mapping) that can be used to reference your provider, second, copy your provider to the location where it is expected to be found, and finally, register it to the Index Service with the parameters you'd like.

### 4.1. Establish mapping of your information provider

To establish the mapping of your provider, you need to edit the `$GLOBUS_LOCATION/etc/globus_wsrf_mds_index/jndi-config.xml` file.

You should see an *executableMappings* section that looks something like this:

```
<parameter>
   <name>executableMappings</name>
      <value>
```

[3] http://www.cygwin.com/

```
         aggr-test=aggregator-exec-test.sh,
         pingexec=example-ping-exec
      </value>
</parameter>
```

To add our *fortune_script.sh* file, let's decide that we're call it the *fortuneProvider* as the mapped name. Our entry would then look like this:

```
fortuneProvider=fortune_script.sh
```

With that line added, the entire entry should look like this (note that an extra comma had to be added before our new entry):

```
<parameter>
   <name>executableMappings</name>
      <value>
         aggr-test=aggregator-exec-test.sh,
         pingexec=example-ping-exec,
         fortuneProvider=fortune_script.sh
      </value>
</parameter>
```

**Note**

The reason we are required to establish this mapping in the first place is for security reasons. The execution aggregator source references this mapping when it's registered, rather than a full path name to a script to avoid allowing arbitrary registrations to be made that can execute arbitrary code. Requiring this mapping be configured before starting the globus container guarantees that the system administrator of the deployment has approved of the use of the new provider.

## 4.2. Copy information provider to correct location

To make sure your provider is in the expected place, it *MUST* be copied to the *$GLOBUS_LOCATION/libexec/aggrexec* directory. Notice how the full path of the script was not specified in the above example when making the mapping. That's because the path of *$GLOBUS_LOCATION/libexec/aggrexec* is simply assumed and it will be pre-pended at run-time for you. Make sure your file resides in this directory with proper executable permissions.

Check the listing to make sure:

```
neillm@glob ~ $ ls -al $GLOBUS_LOCATION/libexec/aggrexec/
total 12
drwxr-xr-x  2 neillm wheel 4096 Jul 16 14:01 .
drwxr-xr-x  6 neillm wheel 4096 Jul  8 14:52 ..
-rwxr-xr-x  1 neillm wheel  345 Jul  8 14:52 aggregator-exec-test.sh
-rwxr-xr-x  1 neillm wheel 1947 Jul 16 13:52 fortune_script.sh
```

## 4.3. Configure the registration file

So now that we've completed the first two steps of enabling the provider, we only have left to decide on the final details of how to make the registration to the Index Service.

To do this, you'll need a registration file. There are many types of registrations that can possibly occur, due to the flexibility of the Aggregator Framework[4]. You can view several examples in the `$GLOBUS_LOCATION/etc/globus_wsrf_mds_aggregator/example-aggregator-registration.xml` file.

For this example, we'll simply use the custom fortune registration file provided[5] , which is specific to the fortune provider we've made that uses the Execution Aggregator source. It's relatively simple, and the fields worth mentioning are shown here:

```
<defaultServiceGroupEPR>
   <wsa:Address>https://127.0.0.1:8443/wsrf/services/DefaultIndexService</wsa:Address>
</defaultServiceGroupEPR>

<defaultRegistrantEPR>
   <wsa:Address>https://127.0.0.1:8443/wsrf/services/fortuneProvider</wsa:Address>
</defaultRegistrantEPR>
```

These fields need to be updated to match how you'll be running your container. You'll need to properly address it, that is. For example, if you're running without security enabled on port 8080 and have an IP address of www.xxx.yyy.zzz, you should substitute the "https://127.0.0.1:8443" base part of the address with "http://www.xxx.yyy.zzz:8080".

Next, view or modify this section of the `fortune-provider-registration.xml` file:

```
<ServiceGroupRegistrationParameters
    xmlns="http://mds.globus.org/servicegroup/client" >

  <!-- Renew this registration every 600 seconds (10 minutes) -->
  <RefreshIntervalSecs>600</RefreshIntervalSecs>
  <Content xsi:type="agg:AggregatorContent"
           xmlns:agg="http://mds.globus.org/aggregator/types">
    <agg:AggregatorConfig xsi:type="agg:AggregatorConfig">
      <agg:ExecutionPollType>

        <!-- Run our script every 300,000 milliseconds (5 minutes) -->
        <agg:PollIntervalMillis>300000</agg:PollIntervalMillis>

        <!-- Specify our mapped ProbeName registered in the
             $GLOBUS_LOCATION/etc/globus_wsrf_mds_index/jndi-config.xml
             file -->
        <agg:ProbeName>fortuneProvider</agg:ProbeName>

      </agg:ExecutionPollType>
    </agg:AggregatorConfig>
    <agg:AggregatorData/>
  </Content>
</ServiceGroupRegistrationParameters>
```

The relevant fields here that you can configure are the following:

[4] http://globus.org/toolkit/docs/4.0/info/aggregator/
[5] fortune-provider-registration.xml

*RefreshIntervalSeconds* - the amount of that time that should pass before the registration is renewed for you. 600 seconds (i.e. 10 minutes) is generally sufficient, and certainly is for this example. (Note: the mds-servicegroup-add utility will perform these registrations for you automatically at these time intervals). This parameter's unit is in seconds.

*PollIntervalMillis* - this is the time interval that we execute the specified provider. It's important to not set this value too low, as there's little value in having it execute extremely frequently given the overhead. For our example, we'll set it to 5 minutes (i.e. 300000 milliseconds). This means, the fortune information published in the Index Service will be updated once every 5 minutes. This parameter's unit is in milliseconds.

*ProbeName* - here is where the executable mapping is put to use. It must exactly match the (left-hand side) name you specified in the *$GLOBUS_LOCATION/etc/globus_wsrf_mds_index/jndi-config.xml*. For this example, we chose this name to be *fortuneProvider*, and you can see that's what we've specified.

Download the example registration file, fortune-provider-registration.xml.

## 4.4. Register with Index Service: run mds-servicegroup-add

Finally, to make the registration of our provider to the Index Service, you should run the *mds-servicegroup-add* program in a similar mannner:

```
neillm@glob ~ $ $GLOBUS_LOCATION/bin/mds-servicegroup-add -s \
https://127.0.0.1:8443/wsrf/services/DefaultIndexService \
fortune-provider-registration.xml

Processing configuration file...
Processed 1 registration entries
Successfully registered
https://127.0.0.1:8443/wsrf/services/fortuneProvider to servicegroup at
https://127.0.0.1:8443/wsrf/services/DefaultIndexService
```

Note that you will have to specify the proper URI location of your Index Service on the command line and not the one specified above (unless it's the same, of course).

## 5. An Example Query

```
neillm@glob bin $ ./wsrf-query -s \
https://127.0.0.1:8443/wsrf/services/DefaultIndexService \
"//*[local-name()='fortuneInformation']"

<fortuneInformation xmlns="">
<fortuneData>
They told me you had proven it When they discovered our results About
a month before. Their hair began to curl The proof was valid, more or
less Instead of understanding it But rather less than more. We'd run
the thing through PRL. He sent them word that we would try Don't tell
a soul about all this To pass where they had failed For it must ever
be And after we were done, to them A secret, kept from all the rest
The new proof would be mailed. Between yourself and me. My notion was
to start again Ignoring all they'd done We quickly turned it into code
```

```
To see if it would run.
</fortuneData>
<fortuneDateAndTime>
Wed Jul 20 12:36:36 BST 2005
</fortuneDateAndTime>
<fortuneSourceURL>
http://anduin.eldar.org/cgi-bin/fortune.pl?text_format=yes
</fortuneSourceURL>
</fortuneInformation>
```

This segment of the query output represents the fortune data we've just written and configured for use. As you can see the *fortuneInformation* block was properly published into the Index Service since it's now been properly configured and registered!

## 6. Contact the author

Contact the author at neillm@mcs.anl.gov[6].

[6] mailto:neillm@mcs.anl.gov

# Chapter 13. WS MDS Index Commandline Reference

## 1. End-user command line tools for WS MDS Index Service

The index service exposes information via service groups and is accessed using the same command-line tools used to query other WSRF services for information. These tools are part of the Java WS Core[1].

- **wsrf-query**[2] performs a query on a resource property document
- **wsrf-get-property**[3] gets a single resource property
- **wsrf-get-properties**[4] gets multiple resource properties.

## 2. Administrative command-line tools for WS MDS Index Service

The **mds-servicegroup-add**[5] command in the Aggregator Framework is used to configure the Index Service to query information from various sources.

[1] http://www.globus.org/toolkit/docs/4.0/common/javawscore/
[2] http://www.globus.org/toolkit/docs/4.0/common/javawscore/rn01re07.html
[3] http://www.globus.org/toolkit/docs/4.0/common/javawscore/rn01re08.html
[4] http://www.globus.org/toolkit/docs/4.0/common/javawscore/rn01re09.html
[5] http://www.globus.org/toolkit/docs/4.0/info/aggregator/rn01re02.html

# GT 4.0 WS MDS Glossary

## A

| | |
|---|---|
| Aggregator Framework | A software framework used to build services that collect and aggregate data. MDS4 Services (such as the Index and Trigger services) are built on the Aggregator Framework, and are sometimes called Aggregator Services. |
| aggregator services | Services that are built on the Aggregator Framework, such as the MDS4 Index Service and Trigger Service.<br>See Also Aggregator Framework, Index Service, Trigger Service. |
| aggregator source | A Java class that implements an interface (defined as part of the Aggregator Framework) to collect XML-formatted data. MDS4 contains three aggregator sources: the query aggregator source, the subscription aggregator source, and the execution aggregator source.<br>See Also query aggregator source, subscription aggregator source, execution aggregator source. |

## E

| | |
|---|---|
| execution aggregator source | An Aggregator Source (included in MDS4) that executes an administrator-supplied program to collect information and make it available to an Aggregator Service such as the Index Service.<br>See Also aggregator source. |

## G

| | |
|---|---|
| Ganglia | A cluster monitoring tool. See http://ganglia.sourceforge.net. |

## H

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

## I

| | |
|---|---|
| Index Service | An aggregator service that serves as a registry similar to UDDI, but much more flexible. Indexes collect information and publish that information as WSRF resource properties.<br>See Also aggregator services. |
| information provider | A "helper" software component that collects or formats resource information, for use by an aggregator source or by a WSRF service when creating resource properties. |

## Q

| | |
|---|---|
| query aggregator source | An aggregator source (included in MDS4) that polls a WSRF service for resource property information. See Also aggregator source. |

## S

| | |
|---|---|
| subscription aggregator source | An aggregator source (included in MDS4) that collects data from a WSRF service via WSRF subscription/notification. See Also aggregator source. |

## T

| | |
|---|---|
| Trigger Service | An aggregator service that 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. See Also aggregator services. |

## W

| | |
|---|---|
| WebMDS | A web-based interface to WS-RF resource property information that can be used as a user-friendly front-end to the Index Service or other WS-RF services. |