

GT 4.2.0 WS MDS Aggregator Framework: Public Interface Guide

DRAFT

GT 4.2.0 WS MDS Aggregator Framework: Public Interface Guide

DRAFT

Table of Contents

1. APIs	1
1. Programming Model Overview	1
2. The Aggregating ServiceGroup framework	1
3. The standard plugins	1
4. Component API	1
2. WS and WSDL	2
1. Protocol overview	2
2. Operations	2
3. WS MDS Aggregator Framework Resource Properties	3
4. Faults	3
5. WSDL and Schema Definition	3
I. WS MDS Commands	4
mds-servicegroup-add	5
mds-set-multiple-termination-time	8
3. Configuring the Aggregator Framework	9
1. Configuration overview	9
2. Syntax of the interface	9
4. Overview of Graphical User Interface	11
A. Errors	12
Glossary	13

List of Tables

A.1. WS MDS Aggregator Error Messages 12

DRAFT

Chapter 1. APIs

1. Programming Model Overview

The Aggregator Framework module consists of an Aggregating ServiceGroup framework which supports plugins as detailed below, as well as a number of standard plugins.

2. The Aggregating ServiceGroup framework

The aggregating servicegroup framework 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.

The framework provides for over-the-wire management of the list of registered resources (through a WS-ServiceGroup interface) and a Java API for connecting sources and sinks together.

In general (although this is not a hard requirement), aggregator sinks will be tied into a specific service implementation, while aggregator sources are more independent. (For example, the trigger and index services act as sinks)

3. The standard plugins

A number of standard aggregator sources are provided, which implement the aggregator source API. These provide for collecting information from/about a WS-Resource by:

- WS-ResourceProperties poll operations
- WS-Notification subscription
- Execution of arbitrary executables

See [Aggregator Sources Reference](#) for more information about standard aggregator sources for GT 4.2.0.

4. Component API

There are two main Java interfaces in the aggregator framework.

- [AggregatorSink](#)¹ - which is implemented by sinks that can receive data from the Aggregator Framework.
- [AggregatorSource](#)² - which is implemented by sources that can feed data into the Aggregator Framework.

In addition, the AggregatorContent class is used when configuring an aggregator service programmatically, and to represent the data published in the aggregator's `Entry` resource property. All aggregator classes and interfaces are documented in the [aggregator Java API documentation](#)³

¹ http://www.globus.org/api/javadoc-4.2.0/globus_wsrf_mds_aggregator/org/globus/mds/aggregator/impl/AggregatorSink.html

² http://www.globus.org/api/javadoc-4.2.0/globus_wsrf_mds_aggregator/org/globus/mds/aggregator/impl/AggregatorSource.html

³ http://www.globus.org/api/javadoc-4.2.0/globus_wsrf_mds_aggregator/

Chapter 2. WS and WSDL

1. Protocol overview

The Aggregator Framework builds on the [WS-ServiceGroup](#)¹ and [WS-ResourceLifetime](#)² 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.

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](#) for source information and [per-service documentation](#) for sink information - for example the [Index Service](#) and the [Trigger Service](#).)

2. Operations

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 [System Administrator's Guide](#)).

2.2. AggregatorServiceGroupEntry

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

¹ http://viewcvs.globus.org/viewcvs.cgi/wsrf/schema/wsrf/servicegroup/sgw-2.wsdl?revision=1.2&view=markup&pathrev=globus_4_2_branch

² http://viewcvs.globus.org/viewcvs.cgi/wsrf/schema/wsrf/lifetime/rlw-2.wsdl?revision=1.2&view=markup&pathrev=globus_4_2_branch

3. WS MDS Aggregator Framework Resource Properties

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)

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.

5. WSDL and Schema Definition

- [AggregatorServiceGroup](#)³
- [AggregatorServiceGroupEntry](#)⁴
- [common types used by AggregatorServiceGroup and AggregatorServiceGroupEntry](#)⁵

Other relevant source files are the:

- [WSRF service group schema](#)⁶
- [WSRF resource lifetime schema](#)⁷
- MDS Usefulrp schema.

³ http://viewcvs.globus.org/viewcvs.cgi/ws-mds/aggregator/schema/mds/aggregator/aggregator_service_group_port_type.wsdl?revision=1.5&view=markup&pathrev=globus_4_2_branch

⁴ http://viewcvs.globus.org/viewcvs.cgi/ws-mds/aggregator/schema/mds/aggregator/aggregator_service_group_entry_port_type.wsdl?revision=1.6&view=markup&pathrev=globus_4_2_branch

⁵ http://viewcvs.globus.org/viewcvs.cgi/ws-mds/aggregator/schema/mds/aggregator/aggregator-types.xsd?revision=1.6&view=markup&pathrev=globus_4_2_branch

⁶ http://viewcvs.globus.org/viewcvs.cgi/wsrf/schema/wsrf/servicegroup/sgw-2.wsdl?revision=1.2&view=markup&pathrev=globus_4_2_branch

⁷ http://viewcvs.globus.org/viewcvs.cgi/wsrf/schema/wsrf/lifetime/rlw-2.wsdl?revision=1.2&view=markup&pathrev=globus_4_2_branch

WS MDS Commands

DRAFT

Name

`mds-servicegroup-add` -- Registering grid resources to aggregating MDS services such as the Index, Archive and Trigger services

`mds-servicegroup-add`

Tool description

mds-servicegroup-add creates a set of registrations to a WS-ServiceGroup and periodically renews those registrations. It is intended primarily for registering grid resources to aggregating MDS services such as the Index and Trigger services.

The tool can be deployed at the aggregating service, at resource services, or at any other location.

This allows registrations to be configured by the administrator of the aggregating service, or by the administrator of resources, by a third party, or by some combination of those.

Registrations are defined in an XML configuration file, which is documented here: [Registering Aggregator Sources](#).

For an example using an Index Service, see [Simple usage for the Index Service](#).

And remember to note the section on [Limitations](#).

Command syntax

The basic syntax for **mds-servicegroup-add** is:

```
mds-servicegroup-add [options] config.xml
```

where:

-s ht-tp(s)://host:port/service-group-address	A URL to the service group against which the <code>mds-servicegroup-add</code> request will be executed. This command line argument is an optional argument, it is only necessary that this URL argument be specified in the case that there are no suitable target service group EPRs present in the configuration file . Any end point references found in the configuration file will automatically override the EPR specified by this argument on the command-line. If this argument is not specified and no suitable service group EPR is present in the configuration file, the target EPR defaults to the <code>DefaultIndexService</code> on the local host using the default TLS port of 8443.
-o outputFile	If this argument is specified, mds-servicegroup-add will write the EPRs of all successfully created service group entries from the target resource to this file. This file can then be used as input to the mds-set-multiple-termination-time command.
-q seconds	By default, mds-servicegroup-add will continue to run, refreshing the lifetimes for the service group entry resources it creates. Use this option to cause mds-servicegroup-add to terminate itself after the specified number of seconds has elapsed. This can be helpful when using long-lifetime registrations or when updating entry lifetimes via a different mechanism.
-a	By default, mds-servicegroup-add will attempt to make an authenticated connection to each service group. This option is used to specify anonymous connections (and to prevent mds-servicegroup-add from failing if you don't have a valid Grid credential).
-z auth_type	Specify an authorization type:

	<p><code>self</code> Fail if the server's identity is different from the user's identity.</p> <p><code>host</code> Fail if the server does not have a valid server certificate.</p> <p><code>none</code> Continue regardless of the server's identity.</p>
<code>config.xml</code>	<p>Path to the registration configuration file (see Registering Aggregator Sources).</p> <p>The Globus Toolkit distribution includes an example configuration file: <code>\$GLOBUS_LOCATION/etc/globus_wsrf_mds_aggregator/example-aggregator-registration.xml</code>.</p>

The [common java client options](#) are also supported.

Registering a resource manually

Prerequisites

You need the following before you register a resource with an Index Service:

Note

Beginning with GT 4.0.1, the CAS, RFT and GRAM4 services are automatically registered with the default Index Service.

- Have a working Index Service, as documented in the [Index Service System Administrator's Guide](#).
- Know the EPR to the resource.
- Know the EPR to the Index Service. This can be seen in the container output at startup of the container on the index host, and will look something like this: `https://myhost:8443/wsrf/services/DefaultIndexService`

Simple usage for the Index Service

The simplest way to register resources to an index is to:

1. Edit the example configuration file (`$GLOBUS_LOCATION/etc/globus_wsrf_mds_aggregator/example-aggregator-registration.xml`), replacing the EPRs in that file with the EPRs for your resources
2. Run **mds-servicegroup-add** to perform the registrations specified in that file.

For example, to register to the `DefaultIndexService` with a modified `example-aggregator-registration.xml` file, you could run a command similar to the following:

```
$GLOBUS_LOCATION/bin/mds-servicegroup-add -s \
https://127.0.0.1:8443/wsrf/services/DefaultIndexService \
$GLOBUS_LOCATION/etc/globus_wsrf_mds_aggregator/example-aggregator-registration.xml
```

Limitations

It may be necessary for the tool to continue to run in order for the registrations that it maintains to be kept alive, as registrations will otherwise time out.

DRAFT

Name

`mds-set-multiple-termination-time` -- Administering the termination time of grid resources created by aggregating MDS services such as the Index and Trigger services

`mds-set-multiple-termination-time`

Tool description

mds-set-multiple-termination-time sets the termination time of multiple service group entries. It is intended primarily for working with groups of service group entry resources created by aggregating MDS services such as the Index and Trigger services.

The tool can be deployed at the aggregating service, at resource services, or at any other location.

This allows the lifetime of registrations to be configured by the administrator of the aggregating service, or by the administrator of resources, by a third party, or by some combination of those.

Command syntax

The basic syntax for **mds-set-multiple-termination-time** is:

`mds-set-multiple-termination-time [options]`

where:

<code>-i inputFile</code>	file containing an XML array of Endpoint References, such as one output by the mds-servicegroup-add command when used with the <code>-o</code> option.
<code>-w delay</code>	integer wait delay in seconds that will be added to the current time at the remote resource to generate the resource termination time. If not specified the termination time by defaults is set to the current time at the remote resource.
<code>-n date string</code>	ISO-8601 formatted date string representing an exact date and time, e.g. 2016-06-28T01:06:430Z If not specified the termination time by default is set to the current time at the remote resource.

The [common java client options](#) are also supported.

Chapter 3. Configuring the Aggregator Framework

WS MDS aggregator services (such as MDS Index, MDS Trigger and MDS Archive Tech Preview) inherit their configuration system from the *Aggregator Framework* module.

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* and *MDS Trigger*) which are documented in the per-service documentation.

Registrations to a working Aggregator Framework are configured for the `mdu-servicegroup-add(1)` 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.

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 ServiceGroupadd 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.

2. Syntax of the interface

2.1. Configuring the Aggregator Sources

For detailed information on configuring the three types of aggregator sources provided by the Globus Toolkit, see [Aggregator Sources Reference](#).

- [Configuring Execution Aggregator Source](#)
- [Configuration file: parameters for the query aggregator source](#)
- [Configuration file: parameters for the subscription aggregator source](#)

2.2. Configuring the Aggregator Sink

An aggregator sink may require sink-specific configuration (for example, 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.

2.2.1. Disabling the publishing of the aggregator configuration on the server side

It is now possible to disable the publishing of the aggregator configuration along with the aggregated data. The following optional parameter can be added to the *AggregatorConfiguration* section of the service `jndi-config.xml` file:

```
<parameter>
  <name>publishAggregatorConfiguration</name>
  <value>>false</value> </parameter>
```

By default, this option is disabled and the aggregator configuration information is published.

DRAFT

Chapter 4. Overview of Graphical User Interface

There is no GUI specifically for the Aggregator Framework. The release contains [WebMDS](#) which can be used to display monitoring information in a web browser. Specifically, it can be directed at services based on the Aggregator Framework to display information about resources registered to the Aggregator Framework.

DRAFT

Appendix A. Errors

Table A.1. WS MDS Aggregator Error Messages

Error Code	Definition	Possible Solutions
error	what causes this	possible solutions
WS MDS is built on Java WS Core, please see Java WS Core Error Codes for more error code documentation.		

DRAFT

Glossary

A

Aggregator Framework	A software framework used to build services that collect and aggregate data. WS MDS 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 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.

I

Index Service	An aggregator service in WS MDS that serves as a registry similar to UDDI, but much more flexible. Indexes collect information and publish that information as WSRF resource properties.
---------------	--

T

Trigger Service	An aggregator service (in WS MDS) that collects information and compares that data against a set of conditions defined in a configuration file. When a condition is met, or triggered, the specified action takes place (for example, an email is sent to a system administrator when the disk space on a server reaches a threshold).
-----------------	--