

GT 4.2.1 WS MDS Aggregator Framework: System Administrator's Guide

GT 4.2.1 WS MDS Aggregator Framework: System Administrator's Guide

Introduction

This guide contains advanced configuration and other sysadmin information specific to the WS MDS Aggregator Framework.

Important

The Aggregator Framework is built, installed, and deployed as part of the standard Globus Toolkit installation procedure: [Installing GT 4.2.1](#) and [WS MDS System Administrator's Guide](#).

Table of Contents

Aggregator How-to	5
1. Configuring the Aggregator Framework	1
1. Configuration overview	1
2. Syntax of the interface	1
2. Testing	3
3. Security Considerations	4
1. WS MDS Aggregator Services (Index Service and Trigger Service) Security Considerations	4
4. Debugging	5
1. Logging in Java WS Core	5
5. Troubleshooting	7
1. Error Messages	7
2. Aggregator isn't collecting data	7
3. Aggregator entry disappears	7
Glossary	8
Index	9

List of Tables

5.1. WS MDS Aggregator Error Messages	7
---	---

Aggregator How-to

C

- configuration interface,
 - aggregator sinks,
 - aggregator sources,
 - disable publishing,
 - overview,
- configuring,
 - aggregator sinks,
 - aggregator sources,
 - disabling publishing,
 - overview,

D

- debugging
 - logging,

E

- errors,

L

- logging
 - CEDPS-compliant,
 - debugging,

S

- security considerations,

Chapter 1. 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 `mds-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 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.

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](#).

- [Chapter 6, Configuring Execution Aggregator Source](#)
- [Chapter 4, Configuration file: parameters for the query aggregator source](#)
- [Chapter 5, 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.

Chapter 2. Testing

The Aggregator Framework is a software framework used to create services. To test that the Aggregator Framework is working, deploy and test a service (such as [Index Service](#)).

Chapter 3. Security Considerations

1. WS MDS Aggregator Services (Index Service and Trigger Service) Security Considerations

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.

Chapter 4. Debugging

1. Logging in Java WS Core

The following information applies to Java WS Core and all services built on Java WS Core.

Java WS Core server side has two types of loggers. One logger is used for development logging and by default writes to standard out. The other logger includes system administration information and is [CEDPs best practices](#)¹ compliant.

On client side, only developer logging is available and is configured using `log4j.properties`.

1.1. Development Logging in Java WS Core

The following information applies to Java WS Core and those services built on it.

Logging in the Java WS Core is based on the [Jakarta Commons Logging](#)² API. Commons Logging provides a consistent interface for instrumenting source code while at the same time allowing the user to plug-in a different logging implementation. Currently we use [Log4j](#)³ as a logging implementation. Log4j uses a separate configuration file to configure itself. Please see Log4j documentation for details on the [configuration file format](#)⁴.

1.1.1. Configuring server side developer logs

Server side logging can be configured in `$GLOBUS_LOCATION/container-log4j.properties`, when the container is stand alone container. For tomcat level logging, refer to [Logging for Tomcat](#)⁵. The logger `log4j.appender.A1` is used for developer logging and by default writes output to the system output. By default it is set for all warnings in the Globus Toolkit package to be displayed.

Additional logging can be enabled for a package by adding a new line to the configuration file. Example:

```
#for debug level logging from org.globus.package.FooClass
log4j.category.org.globus.package.name.FooClass=DEBUG
#for warnings from org.some.warn.package
log4j.category.org.some.warn.package=WARN
```

1.1.2. Configuring client side developer logs

Client side logging can be configured in `$GLOBUS_LOCATION/log4j.properties`. The logger `log4j.appender.A1` is used for developer logging and by default writes output to the system output. By default it is set for all warnings in the Globus Toolkit package to be displayed.

¹ <http://cedps.net/index.php/LoggingBestPractices>

² <http://jakarta.apache.org/commons/logging/>

³ <http://logging.apache.org/log4j/>

⁴ [http://logging.apache.org/log4j/docs/api/org/apache/log4j/PropertyConfigurator.html#doConfigure\(java.lang.String,org.apache.log4j.spi.LoggerRepository\)](http://logging.apache.org/log4j/docs/api/org/apache/log4j/PropertyConfigurator.html#doConfigure(java.lang.String,org.apache.log4j.spi.LoggerRepository))

⁵ <http://tomcat.apache.org/tomcat-5.5-doc/logging.html>

1.2. Configuring system administration logs

The specific logger to edit will be `log4j.logger.sysadmin` in `$GLOBUS_LOCATION/container-log4j.properties`. There you can configure the following properties:

```
log4j.appender.infoCategory=org.apache.log4j.RollingFileAppender
log4j.appender.infoCategory.Threshold=INFO
log4j.appender.infoCategory.File=var/containerLog
log4j.appender.infoCategory.MaxFileSize=10MB
log4j.appender.infoCategory.MaxBackupIndex=2
```

Above implies the logging file is rolling with each file size limited to 10MB and the logging information is stored in `$GLOBUS_LOCATION/var/containerLog`.

1.3. Sample log file

The [sample log file](#)⁶ contains many log entries for various scenarios in the Java WS container.

⁶ <http://www.globus.org/toolkit/docs/4.2/4.2.1/common/javawscore/sample-container-log.txt>

Chapter 5. Troubleshooting

For a list of common errors in GT, see [Error Codes](#).

1. Error Messages

Table 5.1. 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.		

2. Aggregator isn't collecting data

Problem: I was able to successfully register an aggregator entry with `mds-servicegroup-add`, but the aggregator isn't collecting data for the registration.

Solution: The fact that the registration was successful does not mean that there are no errors in the registration parameters. Verify that details such as resource EPRs, resource property names, and queries are accurate, and check the container logs for the [aggregator service](#) and (if applicable) the remote service for more information.

3. Aggregator entry disappears

Problem: I was able to successfully register an aggregator entry with `mds-servicegroup-add`, and the aggregator collected information for this entry for a while, but then the entry disappeared.

Solution: make sure that `mds-servicegroup-add` is still running. Registrations time out; `mds-servicegroup-add` refreshes them periodically.

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).
-----------------	--

Index

C

- configuration interface, 1
 - aggregator sinks, 1
 - aggregator sources, 1
 - disable publishing, 2
 - overview, 1
- configuring, 1
 - aggregator sinks, 1
 - aggregator sources, 1
 - disabling publishing, 2
 - overview, 1

D

- debugging
 - logging, 5

E

- errors, 7

L

- logging
 - CEDPS-compliant, 5
 - debugging, 5

S

- security considerations, 4