# GT 4.2.1 Index Service: System Administrator's Guide

# GT 4.2.1 Index Service: System Administrator's Guide

## 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

The Index Service is built and installed as part of a default GT 4.2.1 installation. For basic installation instructions, see Installing GT 4.2.1. For information about configuring WS MDS in general, see WS MDS System Administrator's Guide. No extra installation steps are required for this component.

# Table of Contents

# List of Tables

# Index Service How-tos

# Chapter 1. Configuring the WS MDS Index Service

### Note

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

# 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], and [GRAM4] (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 4.2.1. See the per service links above for information about configuring this capability.

In order for information to appear in the Index Service, the source of that information must be registered to the Index Service. Information sources are registered using tools like mds-servicegroup-add(1). 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 Aggregator Framework and can use any Aggregator Sources Reference to collect information. In the most common case, the index service uses the `QueryAggregatorSource` to gather resource 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.

# 2. Defining the Aggregator Sources

The aggregation sources 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 these standard sources are in the Aggregator Sources Reference.

# Chapter 2. 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 3. Deploying

The Index Service is deployed into the Globus container by default during the standard toolkit installation.

# 1. Deploying into Tomcat

The WS MDS Index Service has been tested to work without any additional setup when deployed into Tomcat. Please follow these Deploying into Tomcat to deploy GT4 services into Tomcat.

### Note

Note: please complete any prerequisite service configuration steps before you deploy into Tomcat.

# Chapter 4. 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/
```

# Chapter 5. Security Considerations

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

## 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 6. Debugging

Because WS MDS is built on Java WS Core, it uses the same sys admin logging, described below:

# 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)
[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 7. Troubleshooting

You can find frequently asked questions here.

For a list of common errors in GT, see Error Codes.

# 1. Empty AggregatorData entry

**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 the QueryAggregatorSource *aggregator source* may have any of the following wrong with it:

- incorrect values for the resource's hostname or port number
- a misspelled resource property name
- 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) to verify that the remote resource is responding.

# 2. Error Messages

### Table 7.1. WS MDS Index Service 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. | | |

# 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