

GT 4.2.0 WS MDS Aggregator Sources Reference

DRAFT

GT 4.2.0 WS MDS Aggregator Sources Reference

Abstract

This reference describes the aggregator sources provided with the GT 4.2.0, how to register them [with what specifically?], and configure each type.

DRAFT

Table of Contents

Aggregator Sources Howtos	6
1. Introduction	1
2. Types of Aggregator Sources in WS MDS	2
1. QueryAggregatorSource	2
2. SubscriptionAggregatorSource	2
3. ExecutionAggregatorSource	2
3. Registering Aggregator Sources	4
1. Registering resources (general)	4
2. Further configuration	6
4. Configuration file: parameters for the query aggregator source	7
1. GetResourcePropertyPollType	7
2. GetMultipleResourcePropertiesPollType	7
3. QueryResourcePropertiesPollType	8
5. Configuration file: parameters for the subscription aggregator source	9
6. Configuring Execution Aggregator Source	10
1. Configuration file: parameters for the execution aggregator source	10
2. Troubleshooting	11
3. Configuring the executable	11
4. Ping test example	11
Glossary	13
Index	14

List of Figures

1.1. Graphic of Information Services Flow 1

DRAFT

List of Tables

3.1. Aggregator configuration parameters 6

DRAFT

Aggregator Sources Howtos

A

aggregator sources,
 configuring
 executable for the execution aggregator source,
 execution ,
 query ,
 subscription ,
execution,
 query,
 registering,
 subscription,
 troubleshooting,

C

configuration file, registering
 example-aggregator-registration.xml,
 parameters,

E

EPR,

I

information flow (diagram),

M

mds-servicegroup-add,

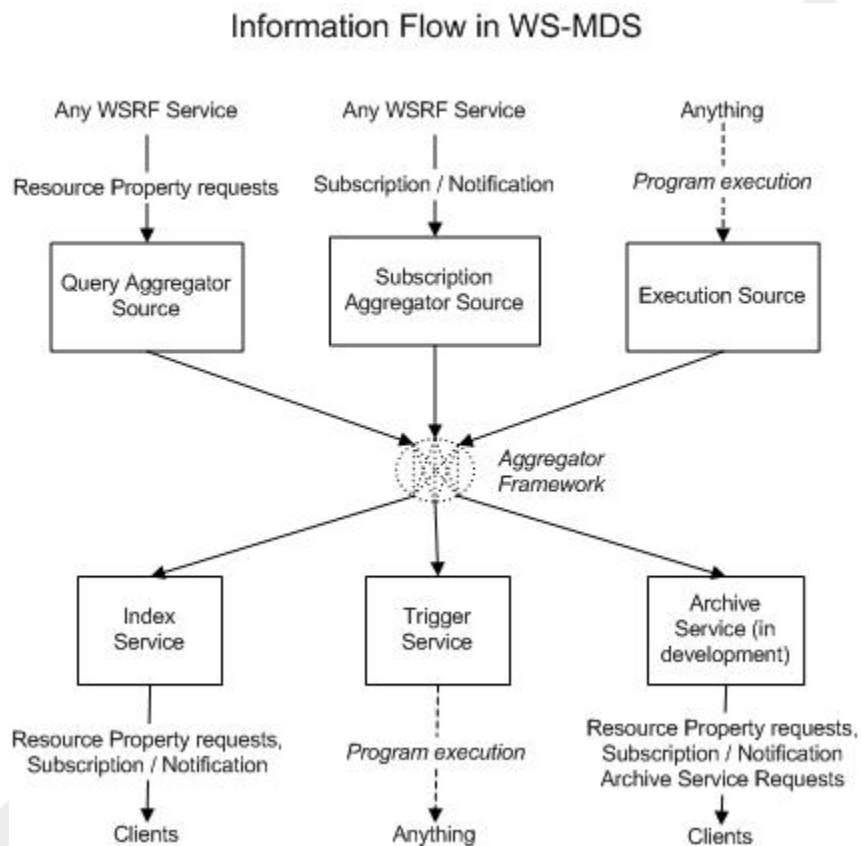
R

registering
 ping test example,
registering aggregator sources,
registering with the default Index Service,

Chapter 1. Introduction

Aggregator sources [glossary] collect information from or about WS-Resources and feed that information to aggregator sinks (such as the Index Service and Trigger Service). The following graphic describes the basic information flow including the three standard aggregator sources: *Query Aggregator Source*, *Subscription Aggregator Source* and Execution Source.

Figure 1.1. Graphic of Information Services Flow



Chapter 2. Types of Aggregator Sources in WS MDS

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).

note? query and subscription agg sources are WSRF, execution source is not.

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

Note

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

1. 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.

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

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

2. SubscriptionAggregatorSource

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

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

3. ExecutionAggregatorSource

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

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

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

Details of the interface between the execution source and local executables are in [Chapter 6, Configuring Execution Aggregator Source](#)

DRAFT

Chapter 3. Registering Aggregator Sources

The following is general configuration information necessary for all aggregator sources (including any custom ones).

1. Registering resources (general)

To register resources with the Index Service:

1. Create a configuration file in XML that specifies registrations. See `$GLOBUS_LOCATION/etc/globus_wsrf_mds_aggregator/example-aggregator-registration.xml` for several specific examples. The configuration file is described in more detail below.
2. Run `mds-servicegroup-add(1)` to perform the registrations specified in that configuration 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
```

- 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 aggregator service, on the same host as a member resource, or on any other host(s).

The configuration file consists of an optional `defaultServiceGroupEPR`, an optional `defaultRegistrantEPR`, and then one or more `ServiceGroupRegistrationParameters` blocks, each of which represents one registration.

You can use the example configuration at `$GLOBUS_LOCATION/etc/globus_wsrf_mds_aggregator/example-aggregator-registration.xml`¹, replacing the EPRs in that file with the EPRs for your resources. It includes many examples of configurations for GRAM, RFT and other situations.

The general syntax of the configuration file is:

```
<?xml version="1.0" encoding="UTF-8" ?>
<ServiceGroupRegistrations
  xmlns="http://mds.globus.org/servicegroup/client">

  // An optional default service group EPR.
  <defaultServiceGroupEPR>
    // Default service group EPR
  </defaultServiceGroupEPR>

  // An optional default registrant EPR.
  <defaultRegistrantEPR>
```

¹ http://viewcvs.globus.org/viewcvs.cgi/*checkout*/ws-mds/aggregator/source/etc/example-aggregator-registration.xml?revision=1.13

```
// Default registrant EPR
</defaultRegistrantEPR>

// An optional default security descriptor file.
<defaultSecurityDescriptorFile>
  // Path name of default security descriptor file
</defaultSecurityDescriptorFile>

// One or more service group registration blocks:

<ServiceGroupRegistrationParameters>
  <ServiceGroupEPR>
    // EPR of the service group to register to
  </ServiceGroupEPR>
  <RegistrantEPR>
    // EPR of the entity to be monitored.
  </RegistrantEPR>
  <InitialTerminationTime>
    // Initial termination time
  </InitialTerminationTime>
  <RefreshIntervalSecs>
    // Refresh interval, in seconds
  </RefreshIntervalSecs>
  <Content type="agg:AggregatorContent">
    // Aggregator-source-specific configuration parameters
  </Content>
</ServiceGroupRegistrationParameters>

</ServiceGroupRegistrations>
```

The following table describes the different blocks of the file and any parameters:

Table 3.1. Aggregator configuration parameters

defaultService-GroupEPR block	Provides a convenient way to register a number of resources to a single service group -- for example, if you wish to register several resources to your default VO index, you can specify that index as the default service group and omit the ServiceGroupEPR blocks from each ServiceGroupRegistrationParameters block.
defaultRegistrantEPR	Provides a convenient way to register a single resource to several service groups -- for example, if you wish to register your local GRAM server to several index servers, you can specify your GRAM server as the default registrant and omit the RegistrantEPR blocks from each ServiceGroupRegistrationParameters block.
defaultSecurity-DescriptorFile	Simply the path to the security descriptor file .
ServiceGroupRegistrationParameters	Each ServiceGroupRegistrationParameters block specifies the parameters used to register a resource to a service group. The parameters specified in this block are:
ServiceGroupEPR	The EPR of the service group to register to. This parameter may be omitted if a defaultServiceGroupEPR block is specified; in this case, the value of defaultServiceGroupEPR will be used instead.
RegistrantEPR	The EPR of the resource to register. This parameter may be omitted if a defaultRegistrantEPR block is specified; in this case, the value of defaultRegistrantEPR will be used instead.
InitialTerminationTime	The initial termination time of this registration (this may be omitted). If the initial termination time is omitted, then the mds-servicegroup-add sets the initial termination time to the current wall time plus 2 times that of the specified RefreshIntervalSecs parameter.
RefreshIntervalSecs	The refresh interval of the registration, in seconds. The mds-servicegroup-add(1) will attempt to refresh the registration according to this interval, by default incrementing the termination time of the registration by 2 times this interval for every successful refresh. If at any point the termination time for the registration expires the registration will be subject to removal within a maximum of 5 minutes.
Content	Aggregator-source-specific registration parameters. The content blocks for the various aggregator sources are described in detail in the following sections.

2. Further configuration

Further configuration information is detailed per source: [QueryAggregatorSource](#), [SubscriptionAggregatorSource](#), and [ExecutionAggregatorSource](#).

Note

Using the ExecutionAggregatorSource also requires configuring the executable itself, which is also covered in the [ExecutionAggregatorSource](#) section.

Chapter 4. Configuration file: parameters for the query aggregator source

The QueryAggregatorSource can use one of the following three configuration blocks in the Content block: GetResourcePropertyPollType, GetMultipleResourcePropertiesPollType and QueryResourcePropertiesPollType.

1. GetResourcePropertyPollType

If a GetResourcePropertyPollType block is used, QueryAggregatorSource will request a single resource property. The block has this form:

```
<Content xsi:type="agg:AggregatorContent"
  xmlns:agg="http://mds.globus.org/aggregator/types">
  <agg:AggregatorConfig xsi:type="agg:AggregatorConfig">
  <agg:GetResourcePropertyPollType>
  <agg:PollIntervalMillis>interval_in_ms</agg:PollIntervalMillis>
  <agg:ResourcePropertyName>rp_namespace:rp_localname</agg:ResourceProp
  </agg:GetResourcePropertyPollType>
  </agg:AggregatorConfig> <agg:AggregatorData/>
</Content>
```

where:

PollIntervalMillis This parameter is the poll refresh period in milliseconds; the ResourcePropertyNames parameter is the QName of the resource property to poll for.

2. GetMultipleResourcePropertiesPollType

If a GetMultipleResourcePropertiesPollType block is used, QueryAggregatorSource will request one or more resource properties. The block has this form:

```
<Content
  xmlns:agg="http://mds.globus.org/aggregator/types"
  xsi:type="agg:AggregatorContent"> <agg:AggregatorConfig
  xsi:type="agg:AggregatorConfig">
  <agg:GetMultipleResourcePropertiesPollType>
  <agg:PollIntervalMillis>interval_in_ms</agg:PollIntervalMillis>
  <agg:ResourcePropertyNames>rp1_namespace:rp1_localname</agg:ResourcePr
  <agg:ResourcePropertyNames>rp2_namespace:rp3_localname</agg:ResourcePr
  <agg:ResourcePropertyNames>rp3_namespace:rp3_localname</agg:ResourcePr
  </agg:GetMultipleResourcePropertiesPollType>
  </agg:AggregatorConfig> <agg:AggregatorData/>
</Content>
```

where:

PollIntervalMillis This parameter is the poll refresh period in milliseconds; the ResourcePropertyNames parameters are the QNames of the resource properties to poll for. There is no limit on the number of ResourcePropertyNames that may be specified.

3. QueryResourcePropertiesPollType

If a `QueryResourcePropertiesPollType` block is used, `QueryAggregatorSource` will request that a query be executed against the Resource Property Set of the remote resource. In the GT4 implementation of WSRF Core, the only query language that is supported is XPath. The block has this form:

```
<Content xmlns:agg="http://mds.globus.org/aggregator/types"
xsi:type="agg:AggregatorContent"> <agg:AggregatorConfig
xsi:type="agg:AggregatorConfig">
<agg:QueryResourcePropertiesPollType>
<agg:PollIntervalMillis>interval_in_ms</agg:PollIntervalMillis>
<agg:QueryExpressionDialect="dialect">
Query Expression </agg:QueryExpression>
</agg:QueryResourcePropertiesPollType>
</agg:AggregatorConfig> <agg:AggregatorData/>
</Content>
```

where:

`PollIntervalMillis`

This parameter is the poll refresh period in milliseconds.

`QueryExpression`

An `xsd:any` element; the `Dialect` attribute specifies the dialect of the query expression.

Chapter 5. Configuration file: parameters for the subscription aggregator source

The configuration block for `SubscriptionAggregatorSource` looks like this:

```
<Content
  xmlns:agg="http://mds.globus.org/aggregator/types"
  xsi:type="agg:AggregatorContent">
  <agg:AggregatorConfig xsi:type="agg:AggregatorConfig">
    <agg:AggregatorSubscriptionType>
      <TopicExpression Dialect="dialect">
        Topic Expression
      </TopicExpression>
      <Precondition Dialect="dialect">
        Precondition
      </Precondition>
      <Selector Dialect="dialect">
        Selector
      </Selector>
      <SubscriptionPolicy>
        Subscription Policy
      </SubscriptionPolicy>
      <InitialTerminationTime>time</InitialTerminationTime>
    </agg:AggregatorSubscriptionType>
  </agg:AggregatorConfig>
  <agg:AggregatorData/>
</Content>
```

where:

`TopicExpression` This is the only required parameter; it specifies the topic expression to use in the subscription request.

For more information, see [this section](#) of the Java WS Core Developer's Guide.

Chapter 6. Configuring Execution Aggregator Source

The ExecutionAggregatorSource requires more configuration than Query and Subscription. In addition to the config file parameters, you must also configure the executable itself. Troubleshooting info and an example are also provided.

1. Configuration file: parameters for the execution aggregator source

The configuration block for ExecutionAggregatorSource (inside the Content block) looks like this:

```
<Content xsi:type="agg:AggregatorContent"
xmlns:agg="http://mds.globus.org/aggregator/types">
<agg:AggregatorConfig xsi:type="agg:AggregatorConfig">
<agg:ExecutionPollType>
<agg:PollIntervalMillis>interval_in_ms</agg:PollIntervalMillis>
<agg:ProbeName>dummy_namespace:probe_name</agg:ProbeName>
</agg:ExecutionPollType>
</agg:AggregatorConfig>
<agg:AggregatorData/>
</Content>
```

where:

PollIntervalMillis This parameter is the poll refresh period in milliseconds.

ProbeName This parameter specifies name of the probe to run. This probe is defined in the `jndi-config.xml` file for the service being configured (for example, the file for the MDS Index Service is `$GLOBUS_LOCATION/etc/globus_wsrf_mds_index_jndi-config.xml`). An `executableMappings` parameter should be defined within this file to map probe names to executable names. For example, this maps the probe names `aggr-test` and `pingexec` to the executables called `aggregator-exec-test.sh` and `example-ping-exec`, respectively. All executables are presumed to be in the directory `$GLOBUS_LOCATION/libexec/aggrexec`.

```
<resource name="configuration"
type="org.globus.mds.aggregator.impl.Aggregator"
<resourceParams>
// ...
<parameter>
<name>executableMappings</name>
<value>aggr-test=aggregator-exec-test.sh,
pingexec=example-ping-exec</value>
</parameter>
</resourceParams>
</resource>
```

2. Troubleshooting

If you've properly configured and registered your script for execution but are getting errors from the container because it cannot find the specified script, there are two likely causes.

First, make sure that your script/program is executable and is located in the `$GLOBUS_LOCATION/libexec/aggrexec` directory. When it's specified in the configuration mentioned above, only specify the name of the script/program, without any qualification or path. For example, using the `ProbeName` as `test-script` will be specifying the file `$GLOBUS_LOCATION/libexec/aggrexec/test-script` script.

Next, make sure that you have correctly created an `executableMappings` definition in the appropriate `jndi-config.xml` file.

3. Configuring the executable

3.1. Name of executable

The executable to run will be `$GLOBUS_LOCATION/libexec/aggrexec/<scriptname>` with `scriptname` supplied by the `ProbeName` parameter in the configuration file.

3.2. Input to executable

Information about the registration will be supplied as command line parameters and on `stdin`.

A single command line parameter will be supplied to the executable. This will be the URL from the EPR of the registered service.

Two XML documents will be sent to `stdin`, in sequence:

1. The first document will be the full EPR to the registered service.
2. The second document will be the `AggregatorConfig` block from the registration message (configuration file).

3.3. Output from executable

The executable must output a well-formed XML document to `stdout`. This output document will be delivered into the Aggregator Framework.

4. Ping test example

4.1. Introduction

This file describes an example of using the Execution Aggregator Source API.

The example provides basic ping information about a registrant. It is intended for illustrative purposes, rather than real deployment use.

The aggregator framework is used by other services to collect information. In this example, it will be shown how to configure the index service to use the execution aggregator source.

4.2. Deploying the example

4.2.1. Install the example script in the correct location.

The example script is installed as: `$GLOBUS_LOCATION/etc/globus_wsrf_mds_aggregator/example-ping-exec`. It is necessary to copy this to the directory where the execution source will look for executables, and ensure that it's executable:

```
$ cp $GLOBUS_LOCATION/etc/globus_wsrf_mds_aggregator/example-ping-  
$GLOBUS_LOCATION/libexec/aggrexec/. $ chmod a+x  
$GLOBUS_LOCATION/libexec/aggrexec/example-ping-exec
```

4.2.2. Configure the index to use the execution source.

By default, the index is configured to use a WS-Resource Properties polling mechanism. It is necessary for this example to change the index configuration to use the execution source instead.

4.2.3. Register some resources.

This can be achieved using the [mds-servicegroup-add tool](#).

Rather than configuring the client to register with parameters for the Resource Property polling source, parameters for the execution source should be supplied in each registration.

Register both resources that you know exist, and also some non-existent resources.

4.2.4. Observe the results.

Start the container hosting the index, start the `mds-servicegroup-add` tool, then query the contents of the index with:

```
$ wsrf-query -s http://localhost:8080/wsrf/services/DefaultIndexService '/'
```

Each registration should be represented as an `Entry` resource property value in the output of `wsrf-query`; embedded in each entry should be an `$examplePingResult` element with the results of ping testing.

Glossary

Q

query aggregator source An aggregator source (included in WS MDS) that polls a WSRF service for resource property information.

S

subscription aggregator source An aggregator source (included in WS MDS) that collects data from a WSRF service via WSRF subscription/notification.

DRAFT

Index

A

- aggregator sources, 1
 - configuring
 - executable for the execution aggregator source, 11
 - execution , 10
 - query , 7
 - subscription , 9
 - execution, 2
 - query, 2
 - registering, 4
 - subscription, 2
 - troubleshooting, 11

C

- configuration file, registering
 - example-aggregator-registration.xml, 4
 - parameters, 5

E

- EPR, 6

I

- information flow (diagram), 1

M

- mds-servicegroup-add, 4

R

- registering
 - ping test example, 11
- registering aggregator sources, 4
- registering with the default Index Service, 4