

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

Index Service How-tos	5
1. Configuring the WS MDS Index Service	1
1. Configuration overview	1
2. Defining the Aggregator Sources	1
2. Configuring the Aggregator Framework	3
1. Configuration overview	3
2. Syntax of the interface	3
3. Deploying	5
1. Deploying into Tomcat	5
4. Testing	6
5. Security Considerations	7
1. WS MDS Aggregator Services (Index Service and Trigger Service) Security Considerations	7
6. Debugging	8
1. Logging in Java WS Core	8
7. Troubleshooting	10
1. Empty AggregatorData entry	10
2. Error Messages	10
Glossary	11
Index	12

List of Tables

7.1. WS MDS Index Service Error Messages	10
--	----

Index Service How-tos

C

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

- configuring,
 - aggregator sinks,
 - aggregator sources,
 - defining aggregator sources,
 - disabling publishing,
 - overview,

D

- debugging
 - logging,
- deploying into Tomcat,

E

- errors,

L

- logging
 - CEDPS-compliant,
 - debugging,

S

- security considerations,

T

- testing,
- testing your installation,
- troubleshooting,

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:

```
wsrp-query -a -z none -s https://127.0.0.1:8443/wsrp/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\(java.lang.String,org.apache.log4j.spi.LoggerRepository\)](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

C

- configuration interface, 1, 3
 - aggregator sinks, 3
 - aggregator sources, 3
 - disable publishing, 4
 - overview, 1, 3
- configuring, 1, 3
 - aggregator sinks, 3
 - aggregator sources, 3
 - defining aggregator sources, 1
 - disabling publishing, 4
 - overview, 1, 3

D

- debugging
 - logging, 8
- deploying into Tomcat, 5

E

- errors, 10

L

- logging
 - CEDPS-compliant, 8
 - debugging, 8

S

- security considerations, 7

T

- testing, 6
- testing your installation, 6
- troubleshooting, 10

GT 4.2.1 Index Service: User's Guide

GT 4.2.1 Index Service: User's Guide

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.

Table of Contents

Index Service How-tos	5
1. Getting Information from the MDS Index Service	1
1. Simple usage	1
I. WS MDS Index User Commands	?
wsrf-query	3
wsrf-get-property	5
wsrf-get-properties	7
globus-wsrf-query	9
globus-wsrf-get-property	12
globus-wsrf-get-properties	14
2. Graphical User Interface	16
3. Troubleshooting	17
1. Error Messages	17
2. General troubleshooting information	17
Glossary	18
Index	19

List of Tables

- 1. Common options 4
- 2. Common options 6
- 3. Common options 8
- 4. Application-specific options 9
- 5. Common options 10
- 6. Common options 12
- 7. Common options 14
- 3.1. WS MDS Index Service Error Messages 17

Index Service How-tos

E

errors,

G

getting information from the Index Service (end users),

R

resource

- globus-wsrf-query,
- querying resource properties,

resource properties

- getting a single resource property from a resource,
- getting multiple resource properties from a resource,
- getting multiple values,
- getting value,
- globus-wsrf-get-properties,
- globus-wsrf-get-property,
- querying,
- querying the resource property document of a resource,
- wsrf-get-properties,
- wsrf-get-property,
- wsrf-query,

T

troubleshooting,

U

usage

- simple,

using,

- basic,

W

wsrf-query,

Chapter 1. Getting Information from the MDS Index Service

To view the information contained in an Index Service, you can use either Java WS Core commands (outlined below) or [WebMDS](#).

1. Simple usage

A typical example of using the default Index Service is with the [wsrf-query](#) Java WS Core command. For example:

```
$GLOBUS_LOCATION/bin/wsrf-query -s https://localhost:8443/wsrf/services/DefaultIndexService
```

displays all the resource properties collected by the default Index Service on your local host.

You can also use an XPath query to drill down your search as well as other Java WS Core commands such as [wsrf-get-property](#) and [wsrf-get-properties](#). For more information, review the [User's Guide](#).

WS MDS Index User Commands

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 [Java WS Core](#) .

- [wsrf-query](#)
- [wsrf-get-property](#)
- [wsrf-get-properties](#)

A set of functionally equivalent tools exist written using WS C core. They tend to be faster alternatives to the above java programs. These tools are part of [C WS Core](#) .

- [globus-wsrf-query\(1\)](#)
- [globus-wsrf-get-property\(1\)](#)
- [globus-wsrf-get-properties\(1\)](#)

The following commands are originally documented under their respective component guides, but are reproduced here for convenience.

Name

wsrf-query -- Performs query on a resource property document

wsrf-query

Tool description

Queries the resource property document of a resource. By default, a simple XPath query is assumed that returns the entire resource property document.

Command syntax

```
wsrf-query [options] [query expression] [dialect]
```

Table 1. Common options

-h, --help	Displays help information about the command.
-d, --debug	Enables debug mode. For example, full stack traces of errors will be displayed.
-e, --eprFile <file>	Specifies an <i>XML</i> file that contains the <i>WS-Addressing</i> endpoint reference.
-s, --service <url>	Specifies the service URL.
-k, --key <name value>	Specifies the resource key. The name is the QName of the resource key in the string form: {namespaceURI}localPart , while the value is the simple value of the key. For complex keys, use the --eprFile option. Example: -k "{http://www.globus.org}MyKey" 123
-f, --descriptor <file>	Specifies a client security descriptor. Overrides all other security settings.
-a, --anonymous	Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.
-g, --delegation <mode>	Enables delegation. mode can be either 'limited' or 'full' . Only supported with the GSI Secure Conversation authentication mechanism.
-l, --contextLifetime <value>	Sets the lifetime of the client security context. value is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.
-m, --securityMech <type>	Specifies the authentication mechanism. type can be 'msg' for GSI Secure Message, or 'conv' for GSI Secure Conversation.
-c, --serverCertificate <file>	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
-p, --protection <type>	Specifies the protection level. type can be 'sig' for signature or 'enc' for encryption.
-x, --proxyFilename <value>	Sets the proxy file to use as client credential.
-z, --authorization <type>	Specifies authorization type. type can be 'self' , 'host' , 'none' , or a string specifying the expected identity of the remote party.
-t, --timeout <timeout>	Specifies client timeout (in seconds). The client will wait maximum of the timeout value for a response from the server before returning an error. By default the timeout value is 10 minutes.

Examples:

```
$ wsrif-query -s https://127.0.0.1:8443/wsrif/services/DefaultIndexService \
  "count(//*[local-name()='Entry'])"
```

```
$ wsrif-query -s https://127.0.0.1:8443/wsrif/services/DefaultIndexService \
  "number(//*[local-name()='GLUECE']/glue:ComputingElement/glue:State/@glue:FreeCPUs)=0"
```

```
$ wsrif-query -s http://localhost:8080/wsrif/services/ContainerRegistryService \
  "/*//*//*/*[local-name()='Address']"
```

Name

`wsrp-get-property --` Gets values of a single resource property

`wsrp-get-property`

Tool description

Gets a single resource property from a resource.

Command syntax

`wsrp-get-property [options] <property>`

The `<property>` is a QName of the resource property in the string form: `{namespaceURI}localPart`.

Table 2. Common options

-h, --help	Displays help information about the command.
-d, --debug	Enables debug mode. For example, full stack traces of errors will be displayed.
-e, --eprFile <file>	Specifies an <i>XML</i> file that contains the <i>WS-Addressing</i> endpoint reference.
-s, --service <url>	Specifies the service URL.
-k, --key <name value>	Specifies the resource key. The name is the QName of the resource key in the string form: {namespaceURI}localPart , while the value is the simple value of the key. For complex keys, use the --eprFile option. Example: -k "{http://www.globus.org}MyKey" 123
-f, --descriptor <file>	Specifies a client security descriptor. Overrides all other security settings.
-a, --anonymous	Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.
-g, --delegation <mode>	Enables delegation. mode can be either ' limited ' or ' full '. Only supported with the GSI Secure Conversation authentication mechanism.
-l, --contextLifetime <value>	Sets the lifetime of the client security context. value is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.
-m, --securityMech <type>	Specifies the authentication mechanism. type can be ' msg ' for GSI Secure Message, or ' conv ' for GSI Secure Conversation.
-c, --serverCertificate <file>	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
-p, --protection <type>	Specifies the protection level. type can be ' sig ' for signature or ' enc ' for encryption.
-x, --proxyFilename <value>	Sets the proxy file to use as client credential.
-z, --authorization <type>	Specifies authorization type. type can be ' self ', ' host ', ' none ', or a string specifying the expected identity of the remote party.
-t, --timeout <timeout>	Specifies client timeout (in seconds). The client will wait maximum of the timeout value for a response from the server before returning an error. By default the timeout value is 10 minutes.

Example:

```
$ wsrif-get-property -s http://localhost:8080/wsrif/services/CounterService \ -k
  "{http://counter.com}CounterKey" 123 \
  "{http://docs.oasis-open.org/wsrif/2004/06/wsrif-WS-ResourceLifetime-1.2-draft-01.xsd}Cu
```

Name

`wsrf-get-properties --` Gets values of multiple resource properties

`wsrf-get-properties`

Tool description

Gets multiple resource properties from a resource.

Command syntax

```
wsrf-get-properties [options] <property1> [<property2>...  
  <propertyN>]
```

Each **<propertyN>** is a QName of the resource property in the string form: **{namespaceURI}localPart**.

Table 3. Common options

-h, --help	Displays help information about the command.
-d, --debug	Enables debug mode. For example, full stack traces of errors will be displayed.
-e, --eprFile <file>	Specifies an <i>XML</i> file that contains the <i>WS-Addressing</i> endpoint reference.
-s, --service <url>	Specifies the service URL.
-k, --key <name value>	Specifies the resource key. The name is the QName of the resource key in the string form: {namespaceURI}localPart , while the value is the simple value of the key. For complex keys, use the --eprFile option. Example: -k "{http://www.globus.org}MyKey" 123
-f, --descriptor <file>	Specifies a client security descriptor. Overrides all other security settings.
-a, --anonymous	Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.
-g, --delegation <mode>	Enables delegation. mode can be either 'limited' or 'full' . Only supported with the GSI Secure Conversation authentication mechanism.
-l, --contextLifetime <value>	Sets the lifetime of the client security context. value is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.
-m, --securityMech <type>	Specifies the authentication mechanism. type can be 'msg' for GSI Secure Message, or 'conv' for GSI Secure Conversation.
-c, --serverCertificate <file>	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
-p, --protection <type>	Specifies the protection level. type can be 'sig' for signature or 'enc' for encryption.
-x, --proxyFilename <value>	Sets the proxy file to use as client credential.
-z, --authorization <type>	Specifies authorization type. type can be 'self' , 'host' , 'none' , or a string specifying the expected identity of the remote party.
-t, --timeout <timeout>	Specifies client timeout (in seconds). The client will wait maximum of the timeout value for a response from the server before returning an error. By default the timeout value is 10 minutes.

Example:

```
$ wsrfg-get-properties -s http://localhost:8080/wsrfg/services/CounterService \ -k
  "{http://counter.com}CounterKey" 123 \
  "{http://docs.oasis-open.org/wsrfg/2004/06/wsrfg-WS-ResourceLifetime-1.2-draft-01.xsd}Cu
  \
  "{http://docs.oasis-open.org/wsrfg/2004/06/wsrfg-WS-ResourceLifetime-1.2-draft-01.xsd}Te
```

Name

globus-wsrf-query -- Query a WSRF resource's Resource Property document

globus-wsrf-query [OPTIONS]... SERVICE-SPECIFIER QUERY-EXPRESSION

Tool description

Perform an XPATH query on a resource property document.

Command syntax

globus-wsrf-query [OPTIONS]... SERVICE-SPECIFIER QUERY-EXPRESSION

Table 4. Application-specific options

-n ---nsMapFile FILENAME.	Use the namespace map entries in <i>FILENAME</i> in the XPATH context.
-N --namespace PREFIX=NAMESPACE-URI	Create a namespace mapping in the XPATH context for the <i>PREFIX</i> string to resolve to the <i>NAMESPACE-URI</i> namespace.
-D --dialect DIALECT-URI	Set query dialect to <i>DIALECT-URI</i> . The value targeted will be interpreted as http://wsrf.globus.org/core/query/targetedXPath (default: http://www.w3.org/TR/1999/REC-xpath-19991116).

Table 5. Common options

-a --anonymous	Use anonymous authentication. Requires either -m 'conv' or transport (https) security.
-d, --debug	Enables debug mode. In debug mode, all SOAP messages will be displayed to stderr and full WSRF Fault messages will be displayed.
-e --eprFile FILENAME	Load service EPR from FILENAME. This EPR is used to contact the WSRF service.
-h --help	Displays help information about the command.
-k --key KEYNAME VALUE	Set resource key in the service EPR to be named KEYNAME with VALUE as its value. This can be combined with -s to construct an EPR without having an xml file on hand. The KEYNAME is a QName string in the format {namespaceURI}localPart. while the VALUE is a literal string to place in the element. For example, the option -k '{http://www.globus.org}MyKey' 128 would be rendered as <MyKey xmlns="http://www.globus.org">128</MyKey>
-m, --securityMech TYPE	Set authentication mechanism. TYPE is one of msg for WS-SecureMessage or conv for WS-SecureConversation.
-p, --protection LEVEL	Set message protection level. LEVEL is one of sig for digital signature or enc for encryption. The default is 'sig'.
-s --service ENDPOINT	Set ENDPOINT the service URL to use. Will be composed with the -k parameter if present to add ReferenceProperties to the ENDPOINT
-t --timeout SECONDS	Set client timeout to SECONDS.
-u --usage	Print short usage message.
-V --version	Show version information and exit.
-v --certKeyFiles CERTIFICATE-FILENAME KEY-FILENAME	Use credentials located in CERTIFICATE-FILENAME and KEY-FILENAME. The key file must be unencrypted.
-x --proxyFilename FILENAME	Use proxy credentials located in FILENAME.
-z --authorization TYPE	Set authorization mode. TYPE can be self , host , none , or a string specifying the identity of the remote party. The default is self .
--versions	Show version information for all loaded modules and exit.

SERVICE-SPECIFIER: [-s URI [-k KEY VALUE] | -e FILENAME]

QUERY-EXPRESSION: XPath-Expression-String

Examples:

```
% globus-wsrf-query -e widget.epr "//*[local-name() = 'CurrentTime']"
<ns0:CurrentTime
  xmlns:ns0="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ns1="http://www.w3.org/2001/XMLSchema"
  xmlns:ns2="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-draft
  ns0:type="ns1:dateTime">2006-05-30T13:53:15Z</ns0:CurrentTime>
```

```
% globus-wsrf-query -e widget.epr "//*[local-name() = 'CurrentTime']/text()"
2006-05-30T13:53:35Z
```

```
% globus-wsrf-query -e widget.epr \
  -N wsrl=http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-draft-
  "//*[wsrl:CurrentTime/text()]"
2006-05-30T13:54:36Z
```

Contents of *widget.epr*:

```
<ns01:EndpointReference xmlns:ns01="http://schemas.xmlsoap.org/ws/2004/03/addressing">
  <ns01:Address>http://globus.my.org:8080/wsrf/services/WidgetService</ns01:Address>
  <ns01:ReferenceProperties>
    <ResourceID xmlns:ns02="http://www.w3.org/2001/XMLSchema-instance" xmlns:ns03="http://
    </ns01:ReferenceProperties>
</ns01:EndpointReference>
```

Limitations

- The namespace mapping option and use of namespace prefixes in the *XPath-Expression-String* does not work when communicating with the Java container unless the *http://wsrf.globus.org/core/query/targetedXPath* dialect is used.

Output and Exit Code

If the query is successful, the program displays the output of the query to *stdout* and terminates with exit code 0. In the case of an error, the type of error will be displayed to *stderr* and the program will terminate with a non-0 exit code.

Name

globus-wsrf-get-property -- Get a resource property's value

globus-wsrf-get-property [OPTIONS]... SERVICE-SPECIFIER PROPERTY-NAME

Tool description

Get the value of a resource property from a WSRF resource.

Command syntax

globus-wsrf-get-property [OPTIONS]... SERVICE-SPECIFIER PROPERTY-NAME

Table 6. Common options

-a --anonymous	Use anonymous authentication. Requires either -m 'conv' or transport (https) security.
-d, --debug	Enables debug mode. In debug mode, all SOAP messages will be displayed to stderr and full WSRF Fault messages will be displayed.
-e --eprFile FILENAME	Load service EPR from FILENAME. This EPR is used to contact the WSRF service.
-h --help	Displays help information about the command.
-k --key KEYNAME VALUE	Set resource key in the service EPR to be named KEYNAME with VALUE as its value. This can be combined with -s to construct an EPR without having an xml file on hand. The KEYNAME is a QName string in the format {namespaceURI}localPart . while the VALUE is a literal string to place in the element. For example, the option -k '{http://www.globus.org}MyKey' 128 would be rendered as <MyKey xmlns="http://www.globus.org">128</MyKey>
-m, --securityMech TYPE	Set authentication mechanism. TYPE is one of msg for WS-SecureMessage or conv for WS-SecureConversation.
-p, --protection LEVEL	Set message protection level. LEVEL is one of sig for digital signature or enc for encryption. The default is 'sig'.
-s --service ENDPOINT	Set ENDPOINT the service URL to use. Will be composed with the -k parameter if present to add ReferenceProperties to the ENDPOINT
-t --timeout SECONDS	Set client timeout to SECONDS.
-u --usage	Print short usage message.
-V --version	Show version information and exit.
-v --certKeyFiles CERTIFICATE-FILENAME KEY-FILENAME	Use credentials located in CERTIFICATE-FILENAME and KEY-FILENAME . The key file must be unencrypted.
-x --proxyFilename FILENAME	Use proxy credentials located in FILENAME .
-z --authorization TYPE	Set authorization mode. TYPE can be self , host , none , or a string specifying the identity of the remote party. The default is self .
--versions	Show version information for all loaded modules and exit.

SERVICE-SPECIFIER: [-s URI [-k KEY VALUE] | -e FILENAME]

PROPERTY-NAME: [{Namespace-URI}]Property-Name

Example:

```
% globus-wsrf-get-property -e widget.epr \  
    '{http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-draft-01.xsd  
  
<ns02:CurrentTime  
    xmlns:ns00="http://www.w3.org/2001/XMLSchema-instance"  
    xmlns:ns01="http://www.w3.org/2001/XMLSchema"  
    xmlns:ns02="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-draft  
    ns00:type="ns01:dateTime">2006-05-30T14:26:35Z</ns02:CurrentTime>
```

Output and Exit Code

If the property exists, its values (if any) are displayed to *stdout* and the program terminates with exit code 0. In the case of an error, the type of error will be displayed to *stderr* and the program will terminate with a non-0 exit code.

Name

globus-wsrf-get-properties -- Get multiple resource property value

globus-wsrf-get-properties [OPTIONS]... SERVICE-SPECIFIER PROPERTY-NAME...

Tool description

Get the value of multiple resource properties from a WSRF resource.

Command syntax

globus-wsrf-get-properties [OPTIONS]... SERVICE-SPECIFIER PROPERTY-NAME...

Table 7. Common options

-a --anonymous	Use anonymous authentication. Requires either -m 'conv' or transport (https) security.
-d, --debug	Enables debug mode. In debug mode, all SOAP messages will be displayed to stderr and full WSRF Fault messages will be displayed.
-e --eprFile FILENAME	Load service EPR from FILENAME. This EPR is used to contact the WSRF service.
-h --help	Displays help information about the command.
-k --key KEYNAME VALUE	Set resource key in the service EPR to be named KEYNAME with VALUE as its value. This can be combined with -s to construct an EPR without having an xml file on hand. The KEYNAME is a QName string in the format {namespaceURI}localPart . while the VALUE is a literal string to place in the element. For example, the option -k '{http://www.globus.org}MyKey' 128 would be rendered as <MyKey xmlns="http://www.globus.org">128</MyKey>
-m, --securityMech TYPE	Set authentication mechanism. TYPE is one of msg for WS-SecureMessage or conv for WS-SecureConversation.
-p, --protection LEVEL	Set message protection level. LEVEL is one of sig for digital signature or enc for encryption. The default is 'sig'.
-s --service ENDPOINT	Set ENDPOINT the service URL to use. Will be composed with the -k parameter if present to add ReferenceProperties to the ENDPOINT
-t --timeout SECONDS	Set client timeout to SECONDS.
-u --usage	Print short usage message.
-V --version	Show version information and exit.
-v --certKeyFiles CERTIFICATE-FILENAME KEY-FILENAME	Use credentials located in CERTIFICATE-FILENAME and KEY-FILENAME. The key file must be unencrypted.
-x --proxyFilename FILENAME	Use proxy credentials located in FILENAME.
-z --authorization TYPE	Set authorization mode. TYPE can be self , host , none , or a string specifying the identity of the remote party. The default is self .
--versions	Show version information for all loaded modules and exit.

SERVICE-SPECIFIER: [-s URI [-k KEY VALUE] | -e FILENAME]

PROPERTY-NAME: [{Namespace-URI}]Property-Name

Example:

```
% globus-wsrf-get-properties \
  -s http://grid.example.org:8080/wsrf/services/WidgetService \
  -k "{http://www.globus.org/namespaces/2004/06/core}WidgetKey" 123 \
  "{http://widgets.com}foo" \
  "{http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-draft-01.xsd}foo"
<ns02:foo
  xmlns:ns00="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ns01="http://www.w3.org/2001/XMLSchema"
  xmlns:ns02="http://widgets.com"
  ns00:type="ns01:string">
Foo Value String
</ns02:foo><ns03:CurrentTime
  xmlns:ns00="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ns01="http://www.w3.org/2001/XMLSchema"
  xmlns:ns03="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-draft-01.xsd"
  ns00:type="ns01:dateTime">2006-05-30T16:04:15Z</ns03:CurrentTime>
```

Output and Exit Code

If the properties exist, their values (if any) are displayed to *stdout* and the program terminates with exit code 0. In the case of an error, the type of error will be displayed to *stderr* and the program will terminate with a non-0 exit code.

Chapter 2. Graphical User Interface

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

Chapter 3. Troubleshooting

General troubleshooting information is based on Java WS Core and is included below.

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

1. Error Messages

Table 3.1. 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.		

2. General troubleshooting information

- In general, if you want to investigate a problem on your own please see [Chapter 10, Debugging](#) for details on how to turn on debugging.
- Most of the command line clients have a `-debug` option that will display more detailed error messages, including the error stack traces.
- [Search the mailing lists](#)¹ such as gt-user@globus.org² or jwscore-user@globus.org³ (before posting a message).
- If you think you have found a bug please report it in our [Bugzilla](#)⁴ system. Please include as much as detail about the problem as possible.

¹ <http://www.globus.org/email-archive-search.php>

² <mailto:gt-user@globus.org>

³ <mailto:jwscore-user@globus.org>

⁴ <http://bugzilla.globus.org/bugzilla/>

Glossary

C

certificate A public key plus information about the certificate owner bound together by the digital signature of a CA. In the case of a CA certificate, the certificate is self signed, i.e. it was signed using its own private key.

W

Web Services Addressing (WSA) The WS-Addressing specification defines transport-neutral mechanisms to address web services and messages. Specifically, it defines XML elements to identify web service endpoints and to secure end-to-end endpoint identification in messages. See the [W3C WS Addressing Working Group](#)¹⁴ for details.

X

XML Extensible Markup Language (XML) is standard, flexible, and extensible data format used for web services. See the [W3C XML site](#)²⁰ for details.

¹⁴ <http://www.w3.org/2002/ws/addr/>

²⁰ <http://www.w3.org/XML/>

Index

E

errors, 17

G

getting information from the Index Service (end users), 1

R

resource

- globus-wsrf-query, 9

- querying resource properties, 9

resource properties

- getting a single resource property from a resource, 5

- getting multiple resource properties from a resource, 7

- getting multiple values, 14

- getting value, 12

- globus-wsrf-get-properties, 14

- globus-wsrf-get-property, 12

- querying, 9

- querying the resource property document of a resource, 3

- wsrf-get-properties, 7

- wsrf-get-property, 5

- wsrf-query, 3

T

troubleshooting, 17

U

usage

- simple, 1

using, 1

- basic, 1

W

wsrf-query, 1

GT 4.2.1 Index Service: Developer's Guide

GT 4.2.1 Index Service: Developer's Guide

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 4.2.1 best practices](#).

¹ http://www.globus.org/toolkit/docs/development/coding_guidelines.html

Table of Contents

Index Service How-tos	7
1. Before you begin	1
1. Feature summary	1
2. Tested platforms	1
3. Technology dependencies	1
4. WS MDS Aggregator Services (Index Service and Trigger Service) Security Considerations	1
2. Usage scenarios	2
1. Retrieving information from an index service	2
2. Adding information to an index	2
3. Writing a Execution Aggregator Information Provider	3
1. Introduction	3
2. Choosing (or conforming to) a Schema	3
3. The Code	4
4. Enabling The Provider	4
5. An Example Query	7
6. Contact the author	8
4. Tutorials	9
5. Architecture and design overview for the WS MDS Index Service	10
6. Architecture and design overview for the WS MDS Aggregator Framework	11
1. Standard aggregator sinks	12
2. Standard aggregator sources	12
7. APIs	14
1. Programming Model Overview	14
8. WS and WSDL	15
1. Protocol overview	15
2. Operations	15
3. WS MDS Aggregator Framework Resource Properties	16
4. Faults	16
5. WSDL and Schema Definition	16
I. WS MDS Index User Commands	?
wsrf-query	18
wsrf-get-property	20
wsrf-get-properties	22
globus-wsrf-query	24
globus-wsrf-get-property	27
globus-wsrf-get-properties	29
II. WS MDS Index Admin Commands	?
mds-servicegroup-add	32
globus-index-add	35
9. Graphical User Interface	36
10. Configuring an executable to retrieve information	37
1. Interface introduction	37
2. Syntax of the interface	37
11. Configuring the WS MDS Index Service	38
1. Configuration overview	38
2. Defining the Aggregator Sources	38
12. Debugging	40
1. Development Logging in Java WS Core	40
2. Enable Debug Logging for the Index Service	40
13. Troubleshooting	42
1. Java WS Core Errors	43

2. General troubleshooting information	45
14. Related Documentation	46
Glossary	47
Index	49

List of Figures

6.1. Graphic of Information Services Flow 11

List of Tables

6.1. Standard aggregator sinks	12
6.2. Standard aggregator sources	13
3. Common options	19
4. Common options	21
5. Common options	23
6. Application-specific options	24
7. Common options	25
8. Common options	27
9. Common options	29
13.1. Java WS Core Errors	44

Index Service How-tos

A

- aggregator sources,
 - execution,
 - query,
 - GetMultipleResourcePropertiesPollType,
 - GetResourcePropertyPollType,
 - QueryResourcePropertiesPollType,
 - subscription,
- AggregatorServiceGroup resource properties,
- apis,
- architecture,

C

- configuration interface,
 - overview,
- configuring,
 - defining aggregator sources,
 - execution aggregator source,
 - overview,

D

- debugging,
 - logging,
- debugging (developer),
- dependencies,

E

- errors,
- execution aggregator information provider,
- execution aggregator source,

F

- features,

G

- globus-index-add,

I

- information providers
 - writing execution aggregator information provider,

L

- logging
 - debugging,

M

- mds-servicegroup-add,

O

- org.globus.mds.aggregator.impl,

P

- platforms,

R

- refresh existing entries in an index service,
- register entries to an index service,
- registering resources to MDS aggregator services,
- related documents,
- resource
 - globus-wsrf-query,
 - querying resource properties,
- resource properties
 - getting a single resource property from a resource,
 - getting multiple resource properties from a resource,
 - getting multiple values,
 - getting value,
 - globus-wsrf-get-properties,
 - globus-wsrf-get-property,
 - querying,
 - querying the resource property document of a resource,
 - wsrf-get-properties,
 - wsrf-get-property,
 - wsrf-query,
- resource properties, AggregatorServiceGroup,

S

- security considerations,
- services,

T

- troubleshooting
 - for developers,
- tutorial
 - build a grid service,

U

- usage scenarios,
- usage scenarios (programming)
 - adding information to an Index Service,
 - retrieving information from an Index Service,

W

- WSDL,

Chapter 1. Before you begin

1. Feature summary

Features new in release 4.2.1

- The Index Service now supports, in addition to queries made using the default XPath dialect, the new TargetedXPath dialect, which enables users to specify their own namespace mappings in queries.

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

<xi:include></xi:include>

3. Technology dependencies

The Index Service depends on the following GT components:

- Java WS Core
- Aggregator Framework

4. 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 2. Usage scenarios

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 [Chapter 6, APIs](#) for API details.

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](#) for more details). To create your own custom information source, see the [Developer's Guide](#).

Chapter 3. GT 4.2.1 Index Service: How to Write a Simple Execution Aggregator Information Provider for WS MDS

1. Introduction

This document is intended to be a starting guide to writing non web-service based *information providers* for the WS MDS. It covers the concepts and walks through a simple example of how to get arbitrary information into the WS MDS using the [Chapter 6, Configuring Execution Aggregator Source](#). 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 WS MDS 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 [Chapter 4, Configuration file: parameters for the query aggregator source](#). 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 WS MDS'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 WS MDS 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 WS MDS. 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>
  <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 WS MDS 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](http://www.cygwin.com/)¹ 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_wsrfd_mds_index/jndi-config.xml` file.

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

```
<parameter>
  <name>executableMappings</name>
  <value>
    agrgr-test=aggregator-exec-test.sh,
```

¹ <http://www.cygwin.com/>

```
    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](#). 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](#)², 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:

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.

² fortune-provider-registration.xml

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 manner:

```
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.gov³.

³ <mailto:neillm@mcs.anl.gov>

Chapter 4. Tutorials

Use of the index service is covered in the [Build a Grid Service Tutorial \(GlobusWORLD 2005\)](#)¹.

¹ <http://www.globus.org/toolkit/tutorials/BAS/>

Chapter 5. Architecture and design overview for the WS MDS Index Service

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.

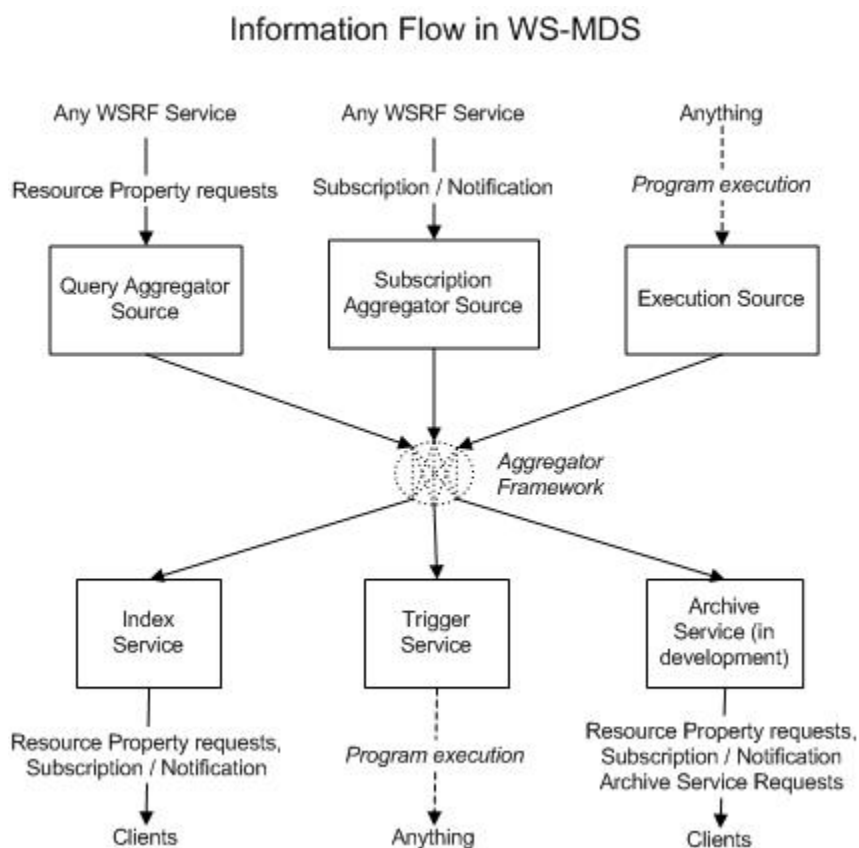
Chapter 6. Architecture and design overview for the WS MDS Aggregator Framework

The WS MDS Aggregator Framework is the software framework on which WS MDS aggregator 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 re-publishing, logging, or archiving the information.

Figure 6.1. Graphic of Information Services Flow



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.

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 6.1. Standard aggregator sinks

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

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`

Table 6.2. Standard aggregator sources

Aggregator Source	Description
QueryAggregatorSource	<p>The query source collects information from a registered resource by using WS-Resource Properties polling mechanisms:</p> <ul style="list-style-type: none">• <code>GetResourcePropertyPollType</code>; requests a single Resource Property from the remote resource.• <code>GetMultipleResourcePropertiesPollType</code>; requests multiple Resource Properties from the remote resource.• <code>QueryResourcePropertiesPollType</code>; requests a query be executed against the Resource Property Set of the remote resource. <p>Polls are made periodically, with both the period and target Resource Properties specified in the registration message.</p>
SubscriptionAggregator-Source	<p>The subscription source collects information from a registered resource using WS-Notification mechanisms. Data is delivered when property values change, rather than periodically.</p>
ExecutionAggregator-Source	<p>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 Chapter 6, Configuring Execution Aggregator Source.</p>

Chapter 7. APIs

1. Programming Model Overview

Index Service queries are performed using resource property requests; consult [Java WS Core](#) 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](#); information about how to create new aggregator sources can be found in [Developer's Guide](#).

Chapter 8. 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 Index User Commands

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 [Java WS Core](#) .

- [wsrf-query](#)
- [wsrf-get-property](#)
- [wsrf-get-properties](#)

A set of functionally equivalent tools exist written using WS C core. They tend to be faster alternatives to the above java programs. These tools are part of [C WS Core](#) .

- [globus-wsrf-query\(1\)](#)
- [globus-wsrf-get-property\(1\)](#)
- [globus-wsrf-get-properties\(1\)](#)

The following commands are originally documented under their respective component guides, but are reproduced here for convenience.

Name

wsrf-query -- Performs query on a resource property document

wsrf-query

Tool description

Queries the resource property document of a resource. By default, a simple XPath query is assumed that returns the entire resource property document.

Command syntax

```
wsrf-query [options] [query expression] [dialect]
```

Table 3. Common options

-h, --help	Displays help information about the command.
-d, --debug	Enables debug mode. For example, full stack traces of errors will be displayed.
-e, --eprFile <file>	Specifies an <i>XML</i> file that contains the <i>WS-Addressing</i> endpoint reference.
-s, --service <url>	Specifies the service URL.
-k, --key <name value>	Specifies the resource key. The name is the QName of the resource key in the string form: {namespaceURI}localPart , while the value is the simple value of the key. For complex keys, use the --eprFile option. Example: -k "{http://www.globus.org}MyKey" 123
-f, --descriptor <file>	Specifies a client security descriptor. Overrides all other security settings.
-a, --anonymous	Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.
-g, --delegation <mode>	Enables delegation. mode can be either 'limited' or 'full' . Only supported with the GSI Secure Conversation authentication mechanism.
-l, --contextLifetime <value>	Sets the lifetime of the client security context. value is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.
-m, --securityMech <type>	Specifies the authentication mechanism. type can be 'msg' for GSI Secure Message, or 'conv' for GSI Secure Conversation.
-c, --serverCertificate <file>	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
-p, --protection <type>	Specifies the protection level. type can be 'sig' for signature or 'enc' for encryption.
-x, --proxyFilename <value>	Sets the proxy file to use as client credential.
-z, --authorization <type>	Specifies authorization type. type can be 'self' , 'host' , 'none' , or a string specifying the expected identity of the remote party.
-t, --timeout <timeout>	Specifies client timeout (in seconds). The client will wait maximum of the timeout value for a response from the server before returning an error. By default the timeout value is 10 minutes.

Examples:

```
$ wsrif-query -s https://127.0.0.1:8443/wsrif/services/DefaultIndexService \
  "count(//*[local-name()='Entry'])"
```

```
$ wsrif-query -s https://127.0.0.1:8443/wsrif/services/DefaultIndexService \
  "number(//*[local-name()='GLUECE']/glue:ComputingElement/glue:State/@glue:FreeCPUs)=0"
```

```
$ wsrif-query -s http://localhost:8080/wsrif/services/ContainerRegistryService \
  "/*/*/*/*[local-name()='Address']"
```

Name

`wsrp-get-property --` Gets values of a single resource property

`wsrp-get-property`

Tool description

Gets a single resource property from a resource.

Command syntax

`wsrp-get-property [options] <property>`

The `<property>` is a QName of the resource property in the string form: `{namespaceURI}localPart`.

Table 4. Common options

-h, --help	Displays help information about the command.
-d, --debug	Enables debug mode. For example, full stack traces of errors will be displayed.
-e, --eprFile <file>	Specifies an <i>XML</i> file that contains the <i>WS-Addressing</i> endpoint reference.
-s, --service <url>	Specifies the service URL.
-k, --key <name value>	Specifies the resource key. The name is the QName of the resource key in the string form: {namespaceURI}localPart , while the value is the simple value of the key. For complex keys, use the --eprFile option. Example: -k "{http://www.globus.org}MyKey" 123
-f, --descriptor <file>	Specifies a client security descriptor. Overrides all other security settings.
-a, --anonymous	Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.
-g, --delegation <mode>	Enables delegation. mode can be either 'limited' or 'full' . Only supported with the GSI Secure Conversation authentication mechanism.
-l, --contextLifetime <value>	Sets the lifetime of the client security context. value is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.
-m, --securityMech <type>	Specifies the authentication mechanism. type can be 'msg' for GSI Secure Message, or 'conv' for GSI Secure Conversation.
-c, --serverCertificate <file>	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
-p, --protection <type>	Specifies the protection level. type can be 'sig' for signature or 'enc' for encryption.
-x, --proxyFilename <value>	Sets the proxy file to use as client credential.
-z, --authorization <type>	Specifies authorization type. type can be 'self' , 'host' , 'none' , or a string specifying the expected identity of the remote party.
-t, --timeout <timeout>	Specifies client timeout (in seconds). The client will wait maximum of the timeout value for a response from the server before returning an error. By default the timeout value is 10 minutes.

Example:

```
$ wsrfg-get-property -s http://localhost:8080/wsrfg/services/CounterService \ -k
  "{http://counter.com}CounterKey" 123 \
  "{http://docs.oasis-open.org/wsrfg/2004/06/wsrfg-WS-ResourceLifetime-1.2-draft-01.xsd}Cu
```

Name

`wsrf-get-properties --` Gets values of multiple resource properties

`wsrf-get-properties`

Tool description

Gets multiple resource properties from a resource.

Command syntax

```
wsrf-get-properties [options] <property1> [<property2>...  
  <propertyN>]
```

Each **<propertyN>** is a QName of the resource property in the string form: **{namespaceURI}localPart**.

Table 5. Common options

-h, --help	Displays help information about the command.
-d, --debug	Enables debug mode. For example, full stack traces of errors will be displayed.
-e, --eprFile <file>	Specifies an <i>XML</i> file that contains the <i>WS-Addressing</i> endpoint reference.
-s, --service <url>	Specifies the service URL.
-k, --key <name value>	Specifies the resource key. The name is the QName of the resource key in the string form: {namespaceURI}localPart , while the value is the simple value of the key. For complex keys, use the --eprFile option. Example: -k "{http://www.globus.org}MyKey" 123
-f, --descriptor <file>	Specifies a client security descriptor. Overrides all other security settings.
-a, --anonymous	Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.
-g, --delegation <mode>	Enables delegation. mode can be either 'limited' or 'full' . Only supported with the GSI Secure Conversation authentication mechanism.
-l, --contextLifetime <value>	Sets the lifetime of the client security context. value is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.
-m, --securityMech <type>	Specifies the authentication mechanism. type can be 'msg' for GSI Secure Message, or 'conv' for GSI Secure Conversation.
-c, --serverCertificate <file>	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
-p, --protection <type>	Specifies the protection level. type can be 'sig' for signature or 'enc' for encryption.
-x, --proxyFilename <value>	Sets the proxy file to use as client credential.
-z, --authorization <type>	Specifies authorization type. type can be 'self' , 'host' , 'none' , or a string specifying the expected identity of the remote party.
-t, --timeout <timeout>	Specifies client timeout (in seconds). The client will wait maximum of the timeout value for a response from the server before returning an error. By default the timeout value is 10 minutes.

Example:

```
$ wsrfg-get-properties -s http://localhost:8080/wsrfg/services/CounterService \ -k
  "{http://counter.com}CounterKey" 123 \
  "{http://docs.oasis-open.org/wsrfg/2004/06/wsrfg-WS-ResourceLifetime-1.2-draft-01.xsd}Cu
  \
  "{http://docs.oasis-open.org/wsrfg/2004/06/wsrfg-WS-ResourceLifetime-1.2-draft-01.xsd}Te
```

Name

globus-wsrf-query -- Query a WSRF resource's Resource Property document

globus-wsrf-query [OPTIONS]... SERVICE-SPECIFIER QUERY-EXPRESSION

Tool description

Perform an XPATH query on a resource property document.

Command syntax

globus-wsrf-query [OPTIONS]... SERVICE-SPECIFIER QUERY-EXPRESSION

Table 6. Application-specific options

-n ---nsMapFile FILENAME.	Use the namespace map entries in <i>FILENAME</i> in the XPATH context.
-N --namespace PREFIX=NAMESPACE-URI	Create a namespace mapping in the XPATH context for the <i>PREFIX</i> string to resolve to the <i>NAMESPACE-URI</i> namespace.
-D --dialect DIALECT-URI	Set query dialect to <i>DIALECT-URI</i> . The value targeted will be interpreted as http://wsrf.globus.org/core/query/targetedXPath (default: http://www.w3.org/TR/1999/REC-xpath-19991116).

Table 7. Common options

-a --anonymous	Use anonymous authentication. Requires either -m 'conv' or transport (https) security.
-d, --debug	Enables debug mode. In debug mode, all SOAP messages will be displayed to stderr and full WSRF Fault messages will be displayed.
-e --eprFile FILENAME	Load service EPR from FILENAME. This EPR is used to contact the WSRF service.
-h --help	Displays help information about the command.
-k --key KEYNAME VALUE	Set resource key in the service EPR to be named KEYNAME with VALUE as its value. This can be combined with -s to construct an EPR without having an xml file on hand. The KEYNAME is a QName string in the format {namespaceURI}localPart . while the VALUE is a literal string to place in the element. For example, the option -k '{http://www.globus.org}MyKey' 128 would be rendered as <MyKey xmlns="http://www.globus.org">128</MyKey>
-m, --securityMech TYPE	Set authentication mechanism. TYPE is one of msg for WS-SecureMessage or conv for WS-SecureConversation.
-p, --protection LEVEL	Set message protection level. LEVEL is one of sig for digital signature or enc for encryption. The default is 'sig'.
-s --service ENDPOINT	Set ENDPOINT the service URL to use. Will be composed with the -k parameter if present to add ReferenceProperties to the ENDPOINT
-t --timeout SECONDS	Set client timeout to SECONDS.
-u --usage	Print short usage message.
-V --version	Show version information and exit.
-v --certKeyFiles CERTIFICATE-FILENAME KEY-FILENAME	Use credentials located in CERTIFICATE-FILENAME and KEY-FILENAME . The key file must be unencrypted.
-x --proxyFilename FILENAME	Use proxy credentials located in FILENAME .
-z --authorization TYPE	Set authorization mode. TYPE can be self , host , none , or a string specifying the identity of the remote party. The default is self .
--versions	Show version information for all loaded modules and exit.

SERVICE-SPECIFIER: [-s URI [-k KEY VALUE] | -e FILENAME]

QUERY-EXPRESSION: XPath-Expression-String

Examples:

```
% globus-wsrf-query -e widget.epr "//*[local-name() = 'CurrentTime']"
<ns0:CurrentTime
  xmlns:ns0="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ns1="http://www.w3.org/2001/XMLSchema"
  xmlns:ns2="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-draft
  ns0:type="ns1:dateTime">2006-05-30T13:53:15Z</ns0:CurrentTime>
```

```
% globus-wsrf-query -e widget.epr "//*[local-name() = 'CurrentTime']/text()"
2006-05-30T13:53:35Z
```

```
% globus-wsrf-query -e widget.epr \
    -N wsrl=http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-draft-
    "/wsrl:CurrentTime/text()"
2006-05-30T13:54:36Z
```

Contents of *widget.epr*:

```
<ns01:EndpointReference xmlns:ns01="http://schemas.xmlsoap.org/ws/2004/03/addressing">
  <ns01:Address>http://globus.my.org:8080/wsrf/services/WidgetService</ns01:Address>
  <ns01:ReferenceProperties>
    <ResourceID xmlns:ns02="http://www.w3.org/2001/XMLSchema-instance" xmlns:ns03="http://
  </ns01:ReferenceProperties>
</ns01:EndpointReference>
```

Limitations

- The namespace mapping option and use of namespace prefixes in the *XPath-Expression-String* does not work when communicating with the Java container unless the *http://wsrf.globus.org/core/query/targetedXPath* dialect is used.

Output and Exit Code

If the query is successful, the program displays the output of the query to *stdout* and terminates with exit code 0. In the case of an error, the type of error will be displayed to *stderr* and the program will terminate with a non-0 exit code.

Name

globus-wsrf-get-property -- Get a resource property's value

globus-wsrf-get-property [OPTIONS]... SERVICE-SPECIFIER PROPERTY-NAME

Tool description

Get the value of a resource property from a WSRF resource.

Command syntax

globus-wsrf-get-property [OPTIONS]... SERVICE-SPECIFIER PROPERTY-NAME

Table 8. Common options

-a --anonymous	Use anonymous authentication. Requires either -m 'conv' or transport (https) security.
-d, --debug	Enables debug mode. In debug mode, all SOAP messages will be displayed to stderr and full WSRF Fault messages will be displayed.
-e --eprFile FILENAME	Load service EPR from FILENAME. This EPR is used to contact the WSRF service.
-h --help	Displays help information about the command.
-k --key KEYNAME VALUE	Set resource key in the service EPR to be named KEYNAME with VALUE as its value. This can be combined with -s to construct an EPR without having an xml file on hand. The KEYNAME is a QName string in the format {namespaceURI}localPart , while the VALUE is a literal string to place in the element. For example, the option -k '{http://www.globus.org}MyKey' 128 would be rendered as <MyKey xmlns="http://www.globus.org">128</MyKey>
-m, --securityMech TYPE	Set authentication mechanism. TYPE is one of msg for WS-SecureMessage or conv for WS-SecureConversation.
-p, --protection LEVEL	Set message protection level. LEVEL is one of sig for digital signature or enc for encryption. The default is 'sig'.
-s --service ENDPOINT	Set ENDPOINT the service URL to use. Will be composed with the -k parameter if present to add ReferenceProperties to the ENDPOINT
-t --timeout SECONDS	Set client timeout to SECONDS.
-u --usage	Print short usage message.
-V --version	Show version information and exit.
-v --certKeyFiles CERTIFICATE-FILENAME KEY-FILENAME	Use credentials located in CERTIFICATE-FILENAME and KEY-FILENAME. The key file must be unencrypted.
-x --proxyFilename FILENAME	Use proxy credentials located in FILENAME.
-z --authorization TYPE	Set authorization mode. TYPE can be self , host , none , or a string specifying the identity of the remote party. The default is self .
--versions	Show version information for all loaded modules and exit.

SERVICE-SPECIFIER: [-s URI [-k KEY VALUE] | -e FILENAME]

PROPERTY-NAME: [{Namespace-URI}]Property-Name

Example:

```
% globus-wsrf-get-property -e widget.epr \  
    '{http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-draft-01.xsd  
  
<ns02:CurrentTime  
    xmlns:ns00="http://www.w3.org/2001/XMLSchema-instance"  
    xmlns:ns01="http://www.w3.org/2001/XMLSchema"  
    xmlns:ns02="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-draft  
    ns00:type="ns01:dateTime">2006-05-30T14:26:35Z</ns02:CurrentTime>
```

Output and Exit Code

If the property exists, its values (if any) are displayed to *stdout* and the program terminates with exit code 0. In the case of an error, the type of error will be displayed to *stderr* and the program will terminate with a non-0 exit code.

Name

globus-wsrf-get-properties -- Get multiple resource property value

globus-wsrf-get-properties [OPTIONS]... SERVICE-SPECIFIER PROPERTY-NAME...

Tool description

Get the value of multiple resource properties from a WSRF resource.

Command syntax

globus-wsrf-get-properties [OPTIONS]... SERVICE-SPECIFIER PROPERTY-NAME...

Table 9. Common options

-a --anonymous	Use anonymous authentication. Requires either -m 'conv' or transport (https) security.
-d, --debug	Enables debug mode. In debug mode, all SOAP messages will be displayed to stderr and full WSRF Fault messages will be displayed.
-e --eprFile FILENAME	Load service EPR from FILENAME. This EPR is used to contact the WSRF service.
-h --help	Displays help information about the command.
-k --key KEYNAME VALUE	Set resource key in the service EPR to be named KEYNAME with VALUE as its value. This can be combined with -s to construct an EPR without having an xml file on hand. The KEYNAME is a QName string in the format {namespaceURI}localPart , while the VALUE is a literal string to place in the element. For example, the option -k '{http://www.globus.org}MyKey' 128 would be rendered as <MyKey xmlns="http://www.globus.org">128</MyKey>
-m, --securityMech TYPE	Set authentication mechanism. TYPE is one of msg for WS-SecureMessage or conv for WS-SecureConversation.
-p, --protection LEVEL	Set message protection level. LEVEL is one of sig for digital signature or enc for encryption. The default is 'sig'.
-s --service ENDPOINT	Set ENDPOINT the service URL to use. Will be composed with the -k parameter if present to add ReferenceProperties to the ENDPOINT
-t --timeout SECONDS	Set client timeout to SECONDS.
-u --usage	Print short usage message.
-V --version	Show version information and exit.
-v --certKeyFiles CERTIFICATE-FILENAME KEY-FILENAME	Use credentials located in CERTIFICATE-FILENAME and KEY-FILENAME. The key file must be unencrypted.
-x --proxyFilename FILENAME	Use proxy credentials located in FILENAME.
-z --authorization TYPE	Set authorization mode. TYPE can be self , host , none , or a string specifying the identity of the remote party. The default is self .
--versions	Show version information for all loaded modules and exit.

SERVICE-SPECIFIER: [-s URI [-k KEY VALUE] | -e FILENAME]

PROPERTY-NAME: [{Namespace-URI}]Property-Name

Example:

```
% globus-wsrf-get-properties \
  -s http://grid.example.org:8080/wsrf/services/WidgetService \
  -k "{http://www.globus.org/namespaces/2004/06/core}WidgetKey" 123 \
  "{http://widgets.com}foo" \
  "{http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-draft-01.xsd}foo"
<ns02:foo
  xmlns:ns0="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ns1="http://www.w3.org/2001/XMLSchema"
  xmlns:ns2="http://widgets.com"
  ns0:type="ns01:string">
Foo Value String
</ns02:foo><ns03:CurrentTime
  xmlns:ns0="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ns1="http://www.w3.org/2001/XMLSchema"
  xmlns:ns3="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-draft-01.xsd"
  ns0:type="ns01:dateTime">2006-05-30T16:04:15Z</ns03:CurrentTime>
```

Output and Exit Code

If the properties exist, their values (if any) are displayed to *stdout* and the program terminates with exit code 0. In the case of an error, the type of error will be displayed to *stderr* and the program will terminate with a non-0 exit code.

WS MDS Index Admin Commands

The mds-servicegroup-add(1) command in the Aggregator Framework is used to configure the Index Service to query information from various sources.

The globus-index-add(1) command line tool is written using WS C and offers similar functionality to mds-servicegroup-add(1) with a few new options.

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: [Chapter 3, 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:

<code>-s ht-tp(s)://host:port/service-group-address</code>	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.
<code>-o outputFile</code>	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.
<code>-q seconds</code>	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.
<code>-a</code>	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).
<code>-z auth_type</code>	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 Chapter 3, Registering Aggregator Sources).</p> <p>The Globus Toolkit distribution includes an example configuration file: <code>\$GLOBUS_LOCATION/etc/globus_wsrp_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/wsrp/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_wsrp_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/wsrp/services/DefaultIndexService \
$GLOBUS_LOCATION/etc/globus_wsrp_mds_aggregator/example-aggregator-registratio
```

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.

Name

globus-index-add -- Registering grid resources to MDS index services

globus-index-add

Tool description

globus-index-add Allows a user to register entries to an Index service and to refresh existing entries. The tool can be run in daemon mode where it is much like [mds-servicegroup-add\(1\)](#). In daemon mode it runs until canceled, periodically refreshing the entry. Unlike [mds-servicegroup-add\(1\)](#) the add and refresh step can be separated. When adding the user can save the EPR of the entry they just added. Then at a later time they can use that EPR to update the entry. This feature makes it possible to script around updates and adds. Further it allows for entry information to be persisted in the event that the add client dies.

Registrations are defined in an XML configuration file, which is documented here: [Section 1, “Registering resources \(general\)”](#).

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

Command syntax

The basic syntax for **globus-index-add** is:

```
globus-index-add [options]
```

where:

-h	Print a usage message.
-q	Write no output messages
-d	Run in daemon mode, refreshing the entry every updates cycle.
-u	Update the entry given in the epr.
-vb	Verbose output
-e <endpoint string>	The endpoint string. Ex: http(s)://host:port/service-group-address. This is used with -a
-a <file>	Add to the index service the entry describe in this ServiceGroupRegistrations file.
-E <file>	<file> contains the EPR of the entry to update with -u
-t <minutes>	Set the number of minutes for the entry to live. Used for both -a and -u.
-me <integer>	Set the maximum amount of retries to update before failing. The program will continue to retry to attempt the refresh <integer> number of times
-z [self] [host] [id <subject>]	Set the authz method.
-ms [sig] [conv] [trans]	Security mechanism. 'msg' for secure message, 'conv' for secure conversation, 'trans' for transport. The default is trans.

Chapter 9. Graphical User Interface

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

Chapter 10. Configuring an executable to retrieve information

1. Interface introduction

The ExecutionAggregatorSource, which may be used by the Index Service, has a domain-specific interface (specifically, the inputs provided to and outputs expected from the executable program).

2. Syntax of the interface

The syntax of the execution source's domain-specific interface is described in [Chapter 6, Configuring Execution Aggregator Source](#).

Chapter 11. 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 12. Debugging

Log output from WS MDS is a useful tool for debugging issues. Because WS MDS is built on top of Java WS Core, developer debugging is the same as described in [Chapter 10, Debugging](#). For information on sys admin logs, see [Chapter 6, Debugging](#).

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](#)¹ 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](#)² as a logging implementation. Log4j uses a separate configuration file to configure itself. Please see Log4j documentation for details on the [configuration file format](#)³.

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](#)⁴, . 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
```

1.2. 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.

2. Enable Debug Logging for the Index Service

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:

¹ <http://jakarta.apache.org/commons/logging/>

² <http://logging.apache.org/log4j/>

³ [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))

⁴ <http://tomcat.apache.org/tomcat-5.5-doc/logging.html>

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

Chapter 13. Troubleshooting

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

1. Java WS Core Errors

Table 13.1. Java WS Core Errors

Error Code	Definition
Failed to acquire notification consumer home instance from registry	Caused by <code>javax.naming.NameNotFoundException</code> : Name <code>services</code> is not bound in
The WS-Addressing 'To' request header is missing	This warning is logged by the container if the request did not contain the necessary <i>WS-Addressing</i> headers, those headers at all or is somehow misconfigured.
java.io.IOException: Token length X > 33554432	If you see this error in the container log, it usually means you are trying to connect to HTTPS server using <code>https</code> specifies 8443 as a port number and <code>http</code> as the protocol name.
java.lang.NoSuchFieldError: DOCUMENT	This error usually indicates a mismatch between the version of Apache Axis that the code was compiled with and the version currently running with.
org.globus.wsrfl.InvalidResourceKeyException: Argument key is null / Resource key is missing	These errors usually indicate that a resource key was not passed with the request or that an invalid resource key was used (the element QName of the resource key did not match what the service expected).
Unable to connect to localhost:xxx	Cannot resolve localhost. The machine's <code>/etc/hosts</code> isn't set up correctly and/or you do not have DNS for
org.globus.common.ChainedIOException: Failed to initialize security context	This may indicate that the user's proxy is invalid.
Error: org.xml.sax.SAXException: Unregistered type: class xxx	This may indicate that an Axis generated XML type, defined by the WS RLS XSD, was not properly registered upon deployment without intervention by the user, sometimes they do not.
No socket factory for 'https' protocol	<p>When a client fails with the following exception:</p> <pre>java.io.IOException: No socket factory for 'https' protocol at org.apache.axis.transport.http.HTTPSender.getSocket(HTTPSender.java:100) org.apache.axis.transport.http.HTTPSender.writeToSocket(HTTPSender.java:110) org.apache.axis.transport.http.HTTPSender.invoke(HTTPSender.java:120)</pre> <p>FIXME - it may have happened because...</p>

Chapter 14. Related Documentation

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

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 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.

C

certificate A public key plus information about the certificate owner bound together by the digital signature of a CA. In the case of a CA certificate, the certificate is self signed, i.e. it was signed using its own private key.

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.

information provider A "helper" software component that collects or formats resource information, for use in WS MDS by an aggregator source or by a WSRF service when creating 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).

W

Web Services Addressing (WSA) The WS-Addressing specification defines transport-neutral mechanisms to address web services and messages. Specifically, it defines XML elements to identify web service endpoints and to secure end-to-end endpoint identification in messages. See the [W3C WS Addressing Working Group](http://www.w3.org/2002/ws/addr/)¹⁴ for details.

¹⁴ <http://www.w3.org/2002/ws/addr/>

X

XML

Extensible Markup Language (XML) is standard, flexible, and extensible data format used for web services. See the [W3C XML site](http://www.w3.org/XML/)²⁰ for details.

²⁰ <http://www.w3.org/XML/>

Index

A

- aggregator sources, 13
 - execution, 13
 - query, 13
 - GetMultipleResourcePropertiesPollType, 13
 - GetResourcePropertyPollType, 13
 - QueryResourcePropertiesPollType, 13
 - subscription, 13
- AggregatorServiceGroup resource properties, 16
- apis, 14
- architecture, 10–11

C

- configuration interface, 38
 - overview, 38
- configuring, 38
 - defining aggregator sources, 38
 - execution aggregator source, 37
 - overview, 38

D

- debugging, 40
 - logging, 40
- debugging (developer), 40
- dependencies, 1

E

- errors, 43
- execution aggregator information provider, 3
- execution aggregator source, 37

F

- features, 1

G

- globus-index-add, 35

I

- information providers
 - writing execution aggregator information provider, 3

L

- logging
 - debugging, 40

M

- mds-servicegroup-add, 32

O

- org.globus.mds.aggregator.impl, 12

P

- platforms, 1

R

- refresh existing entries in an index service, 35
- register entries to an index service, 35
- registering resources to MDS aggregator services, 32
- related documents, 46
- resource
 - globus-wsrf-query, 24
 - querying resource properties, 24
- resource properties
 - getting a single resource property from a resource, 20
 - getting multiple resource properties from a resource, 22
 - getting multiple values, 29
 - getting value, 27
 - globus-wsrf-get-properties, 29
 - globus-wsrf-get-property, 27
 - querying, 24
 - querying the resource property document of a resource, 18
 - wsrf-get-properties, 22
 - wsrf-get-property, 20
 - wsrf-query, 18
- resource properties, AggregatorServiceGroup, 16

S

- security considerations, 1
- services, 15

T

- troubleshooting
 - for developers, 42
- tutorial
 - build a grid service, 9

U

- usage scenarios, 2
- usage scenarios (programming)
 - adding information to an Index Service, 2
 - retrieving information from an Index Service, 2

W

- WSDL, 15

GT 4.2.1 WS MDS Migration Guide

GT 4.2.1 WS MDS Migration Guide

Introduction

The following provides available information about migrating from previous versions of the Globus Toolkit.

Table of Contents

1. Migrating MDS from GT4	1
2. Migrating MDS from GT3	2
3. Migrating MDS from GT2	3
Glossary	4

List of Tables

1.1. Comparison of MDS in GT3 and GT4	1
2.1. Comparison of MDS in GT3 and GT4	2
3.1. Comparison of MDS in GT2 and GT4	3

Chapter 1. Migrating MDS from GT4

Although the basic functionality remains the same for MDS in GT4, the architecture has changed from OGSi in GT3 to WSRF in GT4. In OGSi, services advertise *service data*; in WSRF, services advertise *resource properties*. Resource Properties and service data are very similar -- both provide a mechanism for expressing arbitrary data about grid resources in XML format, as well as query and notification/subscription interfaces to that data.

The GT4 *Index Service* provides the same functionality as the GT3 Index Service; however, the GT4 Index Service supports WSRF service group registration and resource property query and subscription/notification mechanisms, while the GT3 Index Service supported OGSi service group registration and service data query and subscription/notification mechanisms.

The following table shows a mapping of some GT3 concepts/tools to GT4.

Table 1.1. Comparison of MDS in GT3 and GT4

Description	GT2 Version	GT4 Version
Query Operations	FindServiceData (to retrieve a single service data element by name or to perform an XPath query against a service's service data elements)	GetResourceProperty (to retrieve a single resource property by name), GetMultipleResourceProperties (to retrieve multiple resource properties by name), and QueryResourceProperties (to perform an XPath query against a service's resource properties).
APIs used for queries	OGSi (GT3) Core APIs	WS Core APIs
Command-line clients used for queries	<code>ogsi-find-service-data</code>	<code>wsrf-get-property</code> , <code>wsrf-get-properties</code> , <code>wsrf-query</code>
Available GUIs	globus-sdb (standalone client) and WebSDB (web interface)	WebMDS (web interface)
Operations for subscription/notification	OGSi NotificationSource / NotificationSink	WS-Notification
APIs used for subscription/notification	OGSi (GT3) Core APIs	WS Core APIs
Index registration mechanism	GT3 services can be configured to publish their service data to index services.	Index Servers maintain aggregating service groups that include registration information (timeout values, the mechanism to use to acquire information, and additional mechanism-specific parameters) The registration is accomplished by adding an entry to an aggregating service group via the <code>mds-servicegroup-add</code> command. In addition, services may be configured to register themselves to the default index server running in the same container.

A more detailed mapping of OGSi concepts to WSRF concepts can be found [here](http://www-106.ibm.com/developerworks/library/ws-resource/ogsi_to_wsrf_1.0.pdf)¹.

¹ http://www-106.ibm.com/developerworks/library/ws-resource/ogsi_to_wsrf_1.0.pdf

Chapter 2. Migrating MDS from GT3

Although the basic functionality remains the same for MDS in GT4, the architecture has changed from OGSi in GT3 to WSRF in GT4. In OGSi, services advertise *service data*; in WSRF, services advertise *resource properties*. Resource Properties and service data are very similar -- both provide a mechanism for expressing arbitrary data about grid resources in XML format, as well as query and notification/subscription interfaces to that data.

The GT4 *Index Service* provides the same functionality as the GT3 Index Service; however, the GT4 Index Service supports WSRF service group registration and resource property query and subscription/notification mechanisms, while the GT3 Index Service supported OGSi service group registration and service data query and subscription/notification mechanisms.

The following table shows a mapping of some GT3 concepts/tools to GT4.

Table 2.1. Comparison of MDS in GT3 and GT4

Description	GT2 Version	GT4 Version
Query Operations	FindServiceData (to retrieve a single service data element by name or to perform an XPath query against a service's service data elements)	GetResourceProperty (to retrieve a single resource property by name), GetMultipleResourceProperties (to retrieve multiple resource properties by name), and QueryResourceProperties (to perform an XPath query against a service's resource properties).
APIs used for queries	OGSi (GT3) Core APIs	WS Core APIs
Command-line clients used for queries	<code>ogsi-find-service-data</code>	<code>wsrf-get-property</code> , <code>wsrf-get-properties</code> , <code>wsrf-query</code>
Available GUIs	globus-sdb (standalone client) and WebSDB (web interface)	WebMDS (web interface)
Operations for subscription/notification	OGSi NotificationSource / NotificationSink	WS-Notification
APIs used for subscription/notification	OGSi (GT3) Core APIs	WS Core APIs
Index registration mechanism	GT3 services can be configured to publish their service data to index services.	Index Servers maintain aggregating service groups that include registration information (timeout values, the mechanism to use to acquire information, and additional mechanism-specific parameters) The registration is accomplished by adding an entry to an aggregating service group via the <code>mds-servicegroup-add</code> command. In addition, services may be configured to register themselves to the default index server running in the same container.

A more detailed mapping of OGSi concepts to WSRF concepts can be found [here](http://www-106.ibm.com/developerworks/library/ws-resource/ogsi_to_wsrf_1.0.pdf)¹.

¹ http://www-106.ibm.com/developerworks/library/ws-resource/ogsi_to_wsrf_1.0.pdf

Chapter 3. Migrating MDS from GT2

Although the basic functionality remains the same for MDS in GT4, the architecture, standards used, and implementation have changed significantly in GT2. The following table shows a mapping of some GT2 concepts to GT4 concepts.

Table 3.1. Comparison of MDS in GT2 and GT4

Description	GT2 Version	GT4 Version
Format of data describing a resource	LDAP data hierarchy	XML data document
Query language	LDAP queries	XPath queries
Wire protocol for queries	LDAP	WS-ResourceProperties
APIs used for queries	LDAP APIs	WS Core APIs
Command-line clients used for queries	<code>grid-info-search</code>	<code>wsrf-get-property</code> , <code>wsrf-get-properties</code> , <code>wsrf-query</code>
Available GUIs	Various LDAP browsers	WebMDS
Wire protocol for subscription/notification	Not supported	WS-Notification
APIs used for subscription/notification	Not supported	WS Core APIs
Security support	SAML-based security using X.509 user, proxy and host certificates	HTTPS-based security using X.509 user, proxy and host certificates
Queryable index of aggregated information	GIIS, which publishes data using the LDAP-related standards listed above	WS MDS Index Server, which publishes data using the WSRF-related standards listed above
Queryable source of non-aggregated information	GRIS, which uses <i>information providers</i> to gather data from services and then publishes that data the LDAP-related standards listed above	Individual web services, which publish data about their own resources using WSRF-related standards listed above.
Index registration mechanism	MDS servers (GRIS's and, in some cases, GIIS's) register themselves with a GIIS. An MDS server is configured to register itself to a remote index by editing the local MDS server's <code>grid-info-resource-register.conf</code> file, providing information about the location of the remote index to register to and timeout values for the registration	WS MDS Index servers maintain aggregating service groups that include registration information (timeout values, the mechanism to use to acquire information, and additional mechanism-specific parameters) The registration is accomplished by adding an entry to an aggregating service group via the <code>mds-servicegroup-add</code> command. In addition, services may be configured to register themselves to the default index server running in the same container.
Mechanism used by an index to collect information	GIIS's send LDAP queries to remote serves.	WS MDS Index servers use a plugin-based architecture to support several mechanisms to collect information. The Globus Toolkit supplies plugins that support collecting information via polling (resource property queries), subscription/notification, and by program execution.

Glossary

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.

information provider

A "helper" software component that collects or formats resource information, for use in WS MDS by an aggregator source or by a WSRF service when creating resource properties.

GT 4.2.1 Index Service: Quality Profile

Table of Contents

1. Test coverage reports	1
2. Code analysis reports	1
3. Outstanding bugs	1
4. Bug Fixes	1
5. Performance reports	1

<titleabbrev>Quality Profile</titleabbrev>

1. Test coverage reports

- None available at this time.

2. Code analysis reports

- None available at this time.

3. Outstanding bugs

- [All open Index Service bugs and enhancement requests](#)¹

4. Bug Fixes

- [All fixed Index Service bugs and enhancement requests](#)²

5. Performance reports

- None available at this time.

¹ http://bugzilla.globus.org/globus/buglist.cgi?short_desc_type=allwordssubstr&short_desc=&product=MDS&component=wsrcf_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=&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=

² http://bugzilla.globus.org/globus/buglist.cgi?short_desc_type=allwordssubstr&short_desc=&product=MDS&component=wsrcf_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=

GT 4.2.1 Release Notes: Index Service

Table of Contents

1. Component Overview	1
2. Feature summary	1
3. Summary of Changes in WS MDS Index Service	1
4. Bug Fixes	1
5. Known Problems	1
6. Technology dependencies	2
7. Tested platforms	2
8. Associated Standards	2
9. For More Information	3
Glossary	3

<titleabbrev>Release Notes</titleabbrev>

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 4.2.1

- The Index Service now supports, in addition to queries made using the default XPath dialect, the new TargetedXPath dialect, which enables users to specify their own namespace mappings in queries.

3. Summary of Changes in WS MDS Index Service

No new changes in this release

4. Bug Fixes

- All fixed Index Service bugs and enhancement requests.¹

5. Known Problems

The following problems and limitations are known to exist for WS MDS Index Service at the time of the 4.2.1 release:

¹ http://bugzilla.globus.org/globus/buglist.cgi?short_desc_type=allwordssubstr&short_desc=&product=MDS&component=wsrcf_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=&changed-in=&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=

5.1. Limitations

- The WS-MDS Index Service currently keeps the entire set of index data in memory; the size of the index is thus limited by the amount of memory available to the Globus container.

5.2. Outstanding bugs

- [All open Index Service bugs and enhancement requests²](#)

6. Technology dependencies

The Index Service depends on the following GT components:

- [Java WS Core](#)
- [Aggregator Framework](#)

7. 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

<xi:include></xi:include>

8. 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

² 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=&changedin=&chfieldfrom=&chfieldto=Now&chfieldvalue=&cmdtype=doit&newqueryname=&order=Reuse+same+sort+as+last+time&field0-0=noop&type0-0=noop&value0-0=

9. For More Information

See [Index Service](#) for more information about this component.

Glossary