

# **GT 4.2.0 WS MDS UsefulRP: System Administrator's Guide**

DRAFT

---

## GT 4.2.0 WS MDS UsefulRP: System Administrator's Guide

### Introduction

This guide contains advanced configuration information for system administrators working with WS MDS UsefulRP. It provides references to information on procedures typically performed by system administrators, including installation, configuring, deploying, and testing the installation.

#### **Important**

This information is in addition to the basic Globus Toolkit prerequisite, overview, installation, security configuration instructions in the [Installing GT 4.2.0](#) and [WS MDS System Administrator's Guide](#). Read through these guides before continuing!

DRAFT

## Table of Contents

1. Building and Installing .....	1
2. Configuring UsefulRP .....	2
1. Configuration overview .....	2
3. Testing .....	4
4. Security Considerations .....	5
1. UsefulRP Security Considerations .....	5
5. Troubleshooting .....	6
1. Logging in Java WS Core .....	6

DRAFT

# Chapter 1. Building and Installing

Include detailed instructions for building and installing (if this component is built and installed by default when installing gt distribution, state so in the introduction and comment out this chapter).

DRAFT

# Chapter 2. Configuring UsefulRP

## 1. Configuration overview

The system administrator first enables a given service or service resource to use the `org.globus.mds.usefulrp.rpprovider.ResourcePropertyProviderCollection` operation provider by adding the fully qualified Java class name to the provider's parameter value in the service descriptor of a service or resource's `server-config.wsdd` file.

Lastly, the administrator must add a new parameter named `rpProviderConfigFile` and for its corresponding value, specify a full (absolute) OS-native file path to a valid `ResourcePropertyProviderConfig` configuration file. The `ResourcePropertyProviderConfig` file contains all required information for generating one or more Resource Properties for the hosting service or resource.

At service startup, the `ResourcePropertyProviderCollection` operation provider code is initialized and attempts to process the configuration entries found in the file specified by the `rpProviderConfigFile` parameter into a set of one or more background execution tasks (threads) that periodically update the contents of configured Resource Properties with the results of the executing information providers. By default, if there are errors that occur during the first execution of a provider, the timer that controls that provider will be canceled and a warning message output to the container log.

Seen below is a sample service descriptor for the WS MDS `DefaultIndexService` which shows how to configure the service to use the `ResourcePropertyProviderCollection` operation provider and specifies the `rpProviderConfigFile` location used for configuring the sample `GLUEResourceProperty` that the `ResourcePropertyProviderCollection` will process.

```
<service name="DefaultIndexService" provider="Handler" use="literal" style="document">
  <parameter name="providers"
    value="org.globus.mds.usefulrp.rpprovider.ResourcePropertyProviderCollection
    org.globus.wsrfl.impl.servicegroup.ServiceGroupRegistrationProvider
    GetRPPProvider
    GetMRPPProvider
    QueryRPPProvider
    DestroyProvider
    SetTerminationTimeProvider
    SubscribeProvider
    GetCurrentMessageProvider"/>
  <parameter name="rpProviderConfigFile" value="/YOUR-GLOBUS-LOCATION-HERE/etc/globus_ws
  <parameter name="scope" value="Application"/>
  <parameter name="allowedMethods" value="*/>
  <parameter name="handlerClass" value="org.globus.axis.providers.RPCProvider"/>
  <parameter name="className" value="org.globus.mds.index.impl.DefaultIndexService"/>
  <wsdlFile>share/schema/mds/index/index_service.wsdl</wsdlFile>
</service>
```

## 1.1. Configuration File Format

The configuration file format for the `ResourcePropertyProviderCollection` operation provider is simply the XML-serialized form of the `ResourcePropertyProviderConfig` stub object, as defined in the schema file [rpprovider.xsd](#)<sup>1</sup>.

Below is a sample configuration file which configures the GLUE Resource Property provider with element producers using Ganglia to provide cluster information and PBS for scheduler information. This sample configures the provider to generate cluster information using Ganglia on the localhost with the default Ganglia port, and configures PBS as the scheduler information provider. The period of execution is set to 300 seconds for each element producer, but may be configured separately if desired. This configuration mirrors a common information provider setup in the GT4 GRAM `ManagedJobExecutable` service. Using the `RPProvider` Framework, it is possible to generate this information in other services as well.

```
<ns1:ResourcePropertyProviderConfigArray
  xsi:type="ns1:ResourcePropertyProviderConfigArray"
  xmlns:ns1="http://mds.globus.org/rpprovider/2005/08"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <ns1:resourcePropertyProviderConfiguration xsi:type="ns1:resourcePropertyProviderConf
    <ns1:resourcePropertyName xsi:type="xsd:QName" xmlns:mds="http://mds.globus.org/gl
    <ns1:resourcePropertyImpl xsi:type="xsd:string">org.globus.mds.usefulrp.rpprovider.G
    <ns1:resourcePropertyElementProducers xsi:type="ns1:resourcePropertyElementProduce
      <ns1:className xsi:type="xsd:string">org.globus.mds.usefulrp.glue.GangliaElementPr
      <ns1:arguments xsi:type="xsd:string">localhost</ns1:arguments>
      <ns1:arguments xsi:type="xsd:string">8649</ns1:arguments>
      <ns1:period xsi:type="xsd:int">300</ns1:period>
      <ns1:transformClass xsi:type="xsd:string">org.globus.mds.usefulrp.rpprovider.trans
    </ns1:resourcePropertyElementProducers>
    <ns1:resourcePropertyElementProducers xsi:type="ns1:resourcePropertyElementProduce
      <ns1:className xsi:type="xsd:string">org.globus.mds.usefulrp.rpprovider.producers.
      <ns1:arguments xsi:type="xsd:string">libexec/globus-scheduler-provider-pbs</ns1:ar
      <ns1:transformClass xsi:type="xsd:string">org.globus.mds.usefulrp.rpprovider.trans
      <ns1:period xsi:type="xsd:int">300</ns1:period>
    </ns1:resourcePropertyElementProducers>
  </ns1:resourcePropertyProviderConfiguration>
</ns1:ResourcePropertyProviderConfigArray>
```

It is possible to configure the `GLUEResourceProperty` provider to use alternate mechanisms for providing scheduler information by changing the `arguments` field that follows the `SchedulerInfoElementProducer` parameter to a string with a `GLOBUS_LOCATION` relative-path that indicates the GRAM scheduler adapter to use, for example, `libexec/globus-scheduler-provider-fork`.

TBD: It is also possible to pass parameters to the `GLUESchedulerElementTransform` that control even more advanced post-processing and sorting of results when generating GLUE 1.1 XML, e.g. Teragrid resorting code.

<sup>1</sup> [http://viewcvs.globus.org/viewcvs.cgi/ws-mds/usefulrp/schema/schema/mds/usefulrp/rpprovider.xsd?rev=1.2.6.1&only\\_with\\_tag=wsmds\\_usefulrp\\_update\\_4\\_0\\_branch&content-type=text/vnd.viewcvs-markup](http://viewcvs.globus.org/viewcvs.cgi/ws-mds/usefulrp/schema/schema/mds/usefulrp/rpprovider.xsd?rev=1.2.6.1&only_with_tag=wsmds_usefulrp_update_4_0_branch&content-type=text/vnd.viewcvs-markup)

# Chapter 3. Testing

[FIXME describe]

DRAFT

# Chapter 4. Security Considerations

## 1. UsefulRP Security Considerations

Security recommendations for this component include:

- FIXME list

DRAFT

# Chapter 5. Troubleshooting

## 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](#)<sup>1</sup> 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](#)<sup>2</sup> 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](#)<sup>3</sup> as a logging implementation. Log4j uses a separate configuration file to configure itself. Please see Log4j documentation for details on the [configuration file format](#)<sup>4</sup>.

#### 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](#)<sup>5</sup>. 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.

---

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

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

<sup>3</sup> <http://logging.apache.org/log4j/>

<sup>4</sup> [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))

<sup>5</sup> <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](#)<sup>6</sup> contains many log entries for various scenarios in the Java WS container.

---

<sup>6</sup> <http://www.globus.org/toolkit/docs/4.2/4.2.0/common/javawscore/sample-container-log.txt>