

# GT 4.2.0 Migrating Guide for Java WS Core

DRAFT

## GT 4.2.0 Migrating Guide for Java WS Core

### Abstract

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

DRAFT

## Table of Contents

1. Migrating Java WS Core from GT4.0 .....	1
1. Updating higher level services .....	1
2. Code change details .....	2
3. WSDL files and namespace changes .....	4
4. Package names and class name changes .....	6
5. Constant changes .....	8
2. Migrating Java WS Core from GT3 .....	9
1. Key Migration Points .....	9
2. Related Documentation .....	10
3. Migrating Java WS Core from GT2 .....	11
Glossary .....	12

DRAFT

## List of Tables

1.1. Mappings for existing WSDL files and namespaces .....	5
1.2. New WSDL files and namespaces .....	5
1.3. Package names and class name mappings .....	7
1.4. New WSDL files and namespaces .....	8

DRAFT

# Chapter 1. Migrating Java WS Core from GT4.0

Java WS Core Framework has been updated to use the final version of the WSRF/WSN and WS Addressing specifications. [This document](#)<sup>1</sup> provides an overview of the changes and provides a link to the specifications.



## Note

Features that are new in the final specification have not been implemented, only the features from the previous specification have been ported.

## 1. Updating higher level services

The following is a suggested approach to updating the higher level services:

### 1.1. WSDL changes

The specification WSDL names and namespace have changed. The section [WSDL Files and Namespace Changes](#) provides a mapping from old WSDL file names and namespaces to the new equivalent. All operations in the old specification are in the new specification, so no other WSDL change should be needed.

If your service uses compact and schema directories, ensure that the new WSDLs are correctly written and deployed in schema.

### 1.2. Stub generation

The old targets for stub generation can be used as is. Ensure that modified WSDLs have been deployed to GLOBUS\_LOCATION before you run the targets.

### 1.3. Service, client and test code changes

Some class and package names have changed. There is minimal change in the core infrastructure API, but the addressing package has been moved from `org.apache` to `org.globus`.

The section [Package and class name changes](#) provides a table with some mappings from old to new class names. Note this does not cover all cases, but the most commonly used code.

For changes to features and sample code, refer to the section on [Code change details](#). The section discusses changes per feature and should provide details on how to edit your code.

Also, [Constant changes](#) provides information on some constants that have changed.

You should not have to change anything in your Ant build files or deployment infrastructure.

For samples, you can look at counter service.

---

<sup>1</sup> CoreSpecificationUpgradeVer11.pdf

## 2. Code change details

### 2.1. WS Addressing

The Apache WS Addressing project is being archived. So the code base has been moved to the Globus repository and has been checked into `wsrfl/java/lib-src/ws-addressing`. The library now has `org.globus` package names and guarantees support only for the final version of the WS Addressing specification.

One of the key changes in the specification is that Reference Parameters are used now rather than Reference Properties. Java WS Core adds and expects EPR key information in Reference Parameters element now.

### 2.2. Faults

Unlike the old specification, fault cause consists of a single `BaseFaultType`, rather than an array of `BaseFaultType`. The `FaultHelper` utility has been modified to handle this. Now when a fault cause is added, the helper class walks through the chain of fault causes to find one without a fault cause set and then sets the new cause. Higher level services can continue to use the same API as before and faults should get serialized and deserialized correctly.

If in your testing, you look for a specific error code in a `Fault Cause`, it is not an array anymore. You will need to walk through the chain of fault causes. An example of such a test is `testTargetedXPathQueryInvalidRP()` in `org.globus.wsrfl.impl.properties.QueryResourcePropertiesTests.java`.

### 2.3. Lifetime

The new specification allows `SetTerminationTime` to have either an `xsd:dateTime` or `xsd:duration`. Java WS Core currently does **not** support duration and **only** uses the `xsd:dateTime` value set in the request. A bug in Axis ends up with `SetTerminationTime` data binding having a constructor that takes two parameters.

### 2.4. Properties

None of the new methods in the final specification have been implemented; only a minor change in schema, which is taken care of in the providers and `Simple*` implementations.

### 2.5. Subscribing to notifications

- A new data type `Filter` has been defined, which is used to provide subscription information, including the Topics to subscribe to.
- Currently we only support `TopicExpressionType` and hence `FilterType` is expected to have just one element.
- `SubscribeSubscriptionPolicy`: the only piece we used before was `useNotify`. As per latest specification, that is default. If `useRaw` is set in subscription policy a `UnsupportedPolicy` fault will be thrown, any other policy set will cause a `UnrecognizedPolicy` fault.
- Example subscription request:

```
Subscribe request = new Subscribe();
request.setConsumerReference(consumerEPR);
TopicExpressionType topicExpression = new TopicExpressionType();
topicExpression.setDialect(WSNConstants.SIMPLE_TOPIC_DIALECT);
topicExpression.setValue(Counter.VALUE);
```

```
MessageElement element =
    (MessageElement)ObjectSerializer
        .toSOAPElement(topicExpression, WSNConstants.TOPIC_EXPRESSION);
FilterType filter = new FilterType();
filter.set_any(new MessageElement[] { element });
request.setFilter(filter);
```

## 2.6. Unsubscribing notifications

Destroy was used in the old code to remove notification subscription, Unsubscribe should be used now.

```
SubscriptionManager subscriptionPort =
    subscriptionManagerLocator.getSubscriptionManagerPort(
        subscriptionEPR);

...

subscriptionPort.unsubscribe(new Unsubscribe());
```

## 2.7. Consuming notifications

- Notify message has changed and a new utility has been added to take care of deserialization of the message:

```
import org.globus.wsrif.encoding.DeserializationException;
import org.globus.wsrif.utils.NotificationUtil;
import org.oasis.wsrif.properties.ResourcePropertyValueChangeNotificationType;

// Notification callback
public void deliver(List topicPath, EndpointReferenceType producer,
    Object message) {

    ResourcePropertyValueChangeNotificationType changeMessage
        = null;
    try {
        changeMessage = NotificationUtil.getRPValueChangeNotification(message);
    } catch (DeserializationException e) {
        // handle exception
    }
}
```

- ResourcePropertyValueChangeNotificationType API has getNewValues() and getOldValues() instead of getNewValue() and getOldValue().

## 2.8. Subscription Manager Service

This is not relevant, unless you want to write a new subscription manager and not use the one in Core.

The recent specification defines two port types for subscription manager: one with pause/resume and another without. To maintain previous functionality, the default subscription manager service provided in Core uses the pause-able subscription manager port type.

- `GetRP` - **Required**
- `Destroy` - **Required**
- `SetTerminationTime` - Not required
- **Required** resource properties:
  - consumer reference
  - filter type
  - subscription policy
  - creation time

The `BaseSubscriptionProvider` and `PausableSubscriptionProvider` cover all the required functionality.

(Refer to [Subscribing to Notification](#) section above for more details on what is supported)

## 2.9. Resources that support Subscription

The interface `org.globus.wsrp.Subscription` has been split into two. A new interface, `PausableSubscription` has been introduced in the same package and contains the method for pause/resume functionality. By default, subscription support in Core uses the `PausableSubscription` interface **only**.

If pause/renew capability is required, resources need to implement `PausableSubscription` instead of `Subscription`. If you are looking to maintain old functionality, change to `PausableSubscription`.

# 3. WSDL files and namespace changes

## 3.1. WSDL files and namespace changes

**All WSRF/WSN and WSA namespaces have changed.** The file names have also changed, but they have been updated in the same directory. This table shows the mapping from old filenames and namespaces to the new ones.

**Table 1.1. Mappings for existing WSDL files and namespaces**

Old WSDL files and namespaces	New WSDL files and namespaces
WS-ResourceProperties.xsd <a href="http://docs.oasis-open.org/wsr/2004/06/wsr-WS-ResourceProperties-1.2-draft-01.xsd">http://docs.oasis-open.org/wsr/2004/06/wsr-WS-ResourceProperties-1.2-draft-01.xsd</a>	rp-2.xsd <a href="http://docs.oasis-open.org/wsr/rp-2">http://docs.oasis-open.org/wsr/rp-2</a>
WS-ResourceProperties.wsdl <a href="http://docs.oasis-open.org/wsr/2004/06/wsr-WS-ResourceProperties-1.2-draft-01.wsdl">http://docs.oasis-open.org/wsr/2004/06/wsr-WS-ResourceProperties-1.2-draft-01.wsdl</a>	rpw-2.wsdl <a href="http://docs.oasis-open.org/wsr/rpw-2">http://docs.oasis-open.org/wsr/rpw-2</a>
WS-ResourceLifetime.xsd <a href="http://docs.oasis-open.org/wsr/2004/06/wsr-WS-ResourceLifetime-1.2-draft-01.xsd">http://docs.oasis-open.org/wsr/2004/06/wsr-WS-ResourceLifetime-1.2-draft-01.xsd</a>	rl-2.xsd <a href="http://docs.oasis-open.org/wsr/rl-2">http://docs.oasis-open.org/wsr/rl-2</a>
WS-ResourceLifetime.wsdl <a href="http://docs.oasis-open.org/wsr/2004/06/wsr-WS-ResourceLifetime-1.2-draft-01.wsdl">http://docs.oasis-open.org/wsr/2004/06/wsr-WS-ResourceLifetime-1.2-draft-01.wsdl</a>	rlw-2.wsdl <a href="http://docs.oasis-open.org/wsr/rlw-2">http://docs.oasis-open.org/wsr/rlw-2</a>
WS-BaseN.wsdl <a href="http://docs.oasis-open.org/wsn/2004/06/wsn-WS-BaseNotification-1.2-draft-01.wsdl">http://docs.oasis-open.org/wsn/2004/06/wsn-WS-BaseNotification-1.2-draft-01.wsdl</a>	bw-2.wsdl <a href="http://docs.oasis-open.org/wsn/bw-2">http://docs.oasis-open.org/wsn/bw-2</a>
WS-Topics.xsd <a href="http://docs.oasis-open.org/wsn/2004/06/wsn-WS-Topics-1.2-draft-01.xsd">http://docs.oasis-open.org/wsn/2004/06/wsn-WS-Topics-1.2-draft-01.xsd</a>	t-1.xsd <a href="http://docs.oasis-open.org/wsn/t-1">http://docs.oasis-open.org/wsn/t-1</a>
WS-BaseFault.xsd <a href="http://docs.oasis-open.org/wsr/2004/06/wsr-WS-BaseFaults-1.2-draft-01.xsd">http://docs.oasis-open.org/wsr/2004/06/wsr-WS-BaseFaults-1.2-draft-01.xsd</a>	bf-2.xsd <a href="http://docs.oasis-open.org/wsr/bf-2">http://docs.oasis-open.org/wsr/bf-2</a>
WS-BaseFault.wsdl <a href="http://docs.oasis-open.org/wsr/2004/06/wsr-WS-BaseFaults-1.2-draft-01.wsdl">http://docs.oasis-open.org/wsr/2004/06/wsr-WS-BaseFaults-1.2-draft-01.wsdl</a>	bfw-2.wsdl <a href="http://docs.oasis-open.org/wsr/bfw-2">http://docs.oasis-open.org/wsr/bfw-2</a>
WS-ServiceGroup.xsd <a href="http://docs.oasis-open.org/wsr/2004/06/wsr-WS-ServiceGroup-1.2-draft-01.xsd">http://docs.oasis-open.org/wsr/2004/06/wsr-WS-ServiceGroup-1.2-draft-01.xsd</a>	sg-2.xsd <a href="http://docs.oasis-open.org/wsr/sg-2">http://docs.oasis-open.org/wsr/sg-2</a>
WS-ServiceGroup.wsdl <a href="http://docs.oasis-open.org/wsr/2004/06/wsr-WS-ServiceGroup-1.2-draft-01.wsdl">http://docs.oasis-open.org/wsr/2004/06/wsr-WS-ServiceGroup-1.2-draft-01.wsdl</a>	sgw-2.wsdl <a href="http://docs.oasis-open.org/wsr/sgw-2">http://docs.oasis-open.org/wsr/sgw-2</a>
WS-Addressing.xsd <a href="http://schemas.xmlsoap.org/ws/2004/03/addressing">http://schemas.xmlsoap.org/ws/2004/03/addressing</a>	ws-addr.xsd <a href="http://www.w3.org/2005/08/addressing">http://www.w3.org/2005/08/addressing</a>

## 3.2. New additional schema and WSDLs

A WS Resource specification had been added. `ResourceUnknown` and `ResourceUnavailableFaultType` are defined in this namespace, instead of multiple definitions in lifetime and resource in the old specification.

**Table 1.2. New WSDL files and namespaces**

Filename	Namespace
wsrf/compact/wsr/resource/r-2.xsd	<a href="http://docs.oasis-open.org/wsr/r-2">http://docs.oasis-open.org/wsr/r-2</a>
wsrf/compact/wsr/resource/rw-2.wsdl	<a href="http://docs.oasis-open.org/wsr/rw-2">http://docs.oasis-open.org/wsr/rw-2</a>

## 4. Package names and class name changes

This section describes package name changes and equivalent classes.

DRAFT

**Table 1.3. Package names and class name mappings**

No.	Old name	New Name
1	org.oasis.wsrflifetime.ResourceUnknownFaultType org.oasis.wsn.ResourceUnknownFaultType org.oasis.wsrflifetime.ResourceUnknownFaultType	org.oasis.wsrflifetime.ResourceUnknownFaultType
	All these faults have been moved to a single common namespace and hence one package.	
2	org.oasis.wsn.TopicPathDialectUnknownFaultType	org.oasis.wsn.TopicExpressionDialectUnknownFaultType
	Equivalent as per new specification.	
3	org.apache.axis.message.addressing.Address	org.globus.axis.message.addressing.Address
	Moved over Apache addressing code to Globus repository and namespace.	
4	org.apache.axis.message.addressing.EndpointReference	org.globus.axis.message.addressing.EndpointReference
	Moved over Apache addressing code to Globus repository and namespace.	
5	org.apache.axis.message.addressing.EndpointReferenceType	org.globus.axis.message.addressing.EndpointReferenceType
	Moved over Apache addressing code to Globus repository and namespace.	
6	org.apache.axis.message.addressing.ReferenceParametersType	org.globus.axis.message.addressing.ReferenceParametersType
	Moved over Apache addressing code to Globus repository and namespace. Reference parameters are now used to store resource key information.	
7	org.globus.axis.message.addressing.AttributedURI	org.globus.axis.message.addressing.AttributedURIType
	<b>API has changed here.</b> There is no direct to set port, schema, etc. Use <code>getValue()</code> and change the URI that is returned.	
8	org.globus.wsrflifetime.notification.PauseSubscriptionProvider	org.globus.wsrflifetime.notification.PausableSubscriptionProvider
	Refer to notes on Subscription Manager [link me].	

## 5. Constant changes

**Table 1.4. New WSDL files and namespaces**

<b>Old constant</b>	<b>New constant</b>
WSNConstants.TOPIC	WSNConstants.TOPIC_SET
WSNConstants.TOPIC_EXPRESSION_DIALECTS	WSNConstants.TOPIC_EXPRESSION_DIALECT

DRAFT

# Chapter 2. Migrating Java WS Core from GT3

## 1. Key Migration Points

### 1.1. Schemas

#### 1.1.1. GWSDL

GT3 used GWSDL to describe service messages and operations. GT4 uses standard *WSDL* with one extensibility attribute to describe resource properties. Please see the [WS-ResourceProperties](#)<sup>1</sup> specification for details on expressing the resource properties in WSDL.

#### 1.1.2. Port Types

In GT3 every service inherited some basic operations and functionality (from `GridService` port type) as required by the OGSi specification. However, in GT4 there is no such requirement (because the *WSRF/WSN* specifications do not require it). Also, the `Factory` port type defined in the OGSi specification does not exist in WSRF/WSN specifications. Therefore, there is no standard create operation or functionality provided by GT4.

#### 1.1.3. WSDL formatting

GT3 relied on *wrapped* formatting of the WSDL. GT4 uses standard *document* formatting. This change introduces differences in how the Java interface for the service looks. Please see the [design document](#) and the [document/literal](#) section for more details.

## 1.2. Developing services

In GT3, the business logic of the service and the state were coupled together in one class. In GT4, the business logic and the state are decoupled and placed in two separate classes. The business logic is put in a *service* class while the state is put in a *resource* class. The *service* is stateless while the *resource* is stateful. Also, GT4 introduces *ResourceHome* classes that are responsible for managing and discovering resources. Please see the [programming model overview](#) for details.

Even though a GT4 developer needs to modify two or three separate files the coding effort is about the same as in GT3.

## 1.3. Configuring services

In GT3, most of the configuration information was stored in the `server-config.wsdd` file. In GT4, since the business logic and state were decoupled, the service information is still kept in `server-config.wsdd` but resource information is now placed in the new `jndi-config.xml` file. Please see the [JNDI](#) section for more details.

---

<sup>1</sup> <http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceProperties-1.2-draft-04.pdf>

## 2. Related Documentation

- [GT3.2 to GT4 Migration: A First HOWTO by Terry Harmer and Julie McCabe](http://www.qub.ac.uk/escience/howtos/GT3%20to%20GT4%20Version%200.3.htm)<sup>2</sup>

DRAFT

---

<sup>2</sup> <http://www.qub.ac.uk/escience/howtos/GT3%20to%20GT4%20Version%200.3.htm>

# Chapter 3. Migrating Java WS Core from GT2

Not applicable. This component did not exist in GT2.

DRAFT

# Glossary

## J

jndi-config.xml

It is an XML-based configuration file used to populate the container registry accessible via the JNDI API. See in the Java WS Core Developer's Guide] for details.

## R

ResourceHome

In Java WS Core, resources are managed and discovered via *ResourceHome* implementations. The *ResourceHome* implementations can also be responsible for creating new resources, performing operations on a set of resources at a time, etc. *ResourceHomes* are configured in JNDI and are associated with a particular web service.

## S

server-config.wsdd

Axis server-side WSDD configuration file. It contains information about the services, the type mappings and various handlers.

## W

Web Services Description Language (WSDL)

WSDL is an XML document for describing Web services. Standardized binding conventions define how to use WSDL in conjunction with SOAP and other messaging substrates. WSDL interfaces can be compiled to generate proxy code that constructs messages and manages communications on behalf of the client application. The proxy automatically maps the XML message structures into native language objects that can be directly manipulated by the application. The proxy frees the developer from having to understand and manipulate XML. See the [WSDL 1.1 specification](#)<sup>15</sup> for details.

Web Services Notification (WSN)

The WS-Notification family of specifications define a pattern-based approach to allowing Web services to disseminate information to one another. This framework comprises mechanisms for basic notification (WS-Notification), topic-based notification (WS-Topics), and brokered notification (WS-BrokeredNotification). See the [OASIS Web Services Notification \(WSN\) TC](#)<sup>18</sup> for details.

Web Services Resource Framework (WSRF)

Web Services Resource Framework (WSRF) is a specification that extends web services for grid applications by giving them the ability to retain state information while at the same time retaining statelessness (using resources). The combination of a web service and a resource is referred to as a WS-Resource. WSRF is a collection of different specifications that manage WS-Resources.

This framework comprises mechanisms to describe views on the state (WS-ResourceProperties), to support management of the state through properties associated with the Web service (WS-ResourceLifetime), to describe how these mechanisms

<sup>15</sup> <http://www.w3.org/TR/wsdl>

<sup>18</sup> [http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=wsn](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsn)

are extensible to groups of Web services (WS-ServiceGroup), and to deal with faults (WS-BaseFaults).

For more information, go to: <http://www.globus.org/wsrf/> and [OASIS Web Services Notification \(WSRF\) TC](#)<sup>19</sup> .

DRAFT

---

<sup>19</sup> [http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=wsrf](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf)