

JMS Notification Framework

Author: JaiPaul Antony – jantony@us.ibm.com
May 30, 2003.

Introduction and Overview

Notification in OGSI is used to deliver messages from one service (source) to another (sink). A client subscribes to notification on the Source, perhaps on behalf of some other client, and the Source sends notification messages to the sink when appropriate.

The purpose of notification is to deliver interesting messages from a notification source to a notification sink. as described in the following.

- A *notification source* is a Grid service instance that implements the NotificationSource portType, and is the sender of notification messages. A source MAY be able to send notification messages to any number of sinks.
- A *notification sink* is a Grid service instance that receives notification messages from any number of sources. A sink MUST implement the NotificationSink portType, which allows it to receive notification messages
- A *notification message* is an XML element sent from a notification source to a notification sink. The XML type of that element is determined by the subscription expression.
- A *subscription expression* is an XML element that describes what messages should be sent from the notification source to the notification sink. The subscription expression also describes when messages should be sent, based on changes to values within a service instance's serviceDataValues.
- In order to establish what and where notification messages are to be delivered, a *subscription* request is issued to a source, containing a subscription expression, the locator of the notification sink to which notification messages are to be sent, and an initial lifetime for the subscription.
- A subscription request causes the creation of a Grid service instance, called a *subscription*, that implements the NotificationSubscription portType. This portType MAY be used by clients to manage the (soft-state) lifetime of the subscription, and to discover properties of the subscription.

This document describes the design and implementation of the framework that enables notification in an OGSA environment. The communication between various components within the framework is achieved using JMS. The client communication is via soap/http.

Requirements

Notification Framework must provide the following features to the services running in an OGSA environment.

1. Provide a Notification framework that is integrated with the OGSI components.
2. The framework would comply with the GridService Specification, and provide support for:
NotificationSource portType,

- NotificationSink portType, and
NotificationSubscription portType.
3. Allow clients (normally NotificationSinks) to subscribe to notifications to changes in ServiceData.
 4. Support for SubscribeByServiceDataName subscription expression type as defined in the GS Spec.
 5. Support for user defined subscription expression types.

NotificationSource PortType

The NotificationSource portType allows clients to subscribe to notification messages from the Grid service instance that implements this portType. The NotificationSource portType extends the GridService portType. The operations and service data definitions on the NotificationSource portType are defined in the OGSi spec.

The client can subscribe to a notification source using expression types defined in the subscriptionExtensibility service data of the notification source. This framework provides support for any queryExpression type to be used as a subscription expression. Additional expression types may be added to the framework.

The two expression types that are currently supported in this implementation are; subscribeByServiceDataNames, and subscribeByServiceDataXPath. These expression types are defined below.

subscribeByServiceDataNames

A subscribeByServiceDataNames results in notification messages being sent whenever any of the named service data elements change.

The queryByServiceDataNames element is defined as:

```
targetNamespace = "http://www.gridforum.org/namespaces/2003/03/OGSI"
<xsd:element name="subscribeByServiceDataNames"
            type="ogsi:SubscribeByNameType"/>
<xsd:complexType name="SubscribeByNameType">
  <xsd:complexContent>
    <xsd:extension base="ogsi:QNamesType">
      <xsd:attribute name="minInterval"
                    type="duration"
                    use="optional"/>
      <xsd:attribute name="maxInterval"
                    type="ogsi:MaxIntervalType"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:simpleType name="MaxIntervalType">
<xsd:union memberTypes="ogsi:InfinityType xsd:duration"/>
</xsd:simpleType>
```

ogsi:QNamesType is defined in OGSi Spec. §9.2.1.1. ogsi:InfinityType is defined in OGSi Spec. §7.6.

The minInterval property specifies the minimum interval between notification messages, expressed in xsd:duration. If this property is not specified, then the notification source MAY choose this value. A notification source MAY also reject a subscription request if it cannot satisfy the minimum interval requested.

The maxInterval property specifies the maximum interval between notification messages, expressed in xsd:duration. If this interval elapses without a change to the named service data elements' values, then the source MUST resend the same values. When the value is "infinity" the source need never resend a service data values if they do not change. If this property is not specified, then the notification source MAY choose this value.

For a subscribeByServiceDataNames subscription, the type of the notification message sent from the notification source to the notification sink MUST be a serviceDataValues element containing the SDE values for all SDE values corresponding to each requested serviceName, even if only some of the elements' values changed since the last message.

subscribeByXPath

This expression type lets a client subscribe to notification messages when the service data changes, and the xpath specified is applied to the service data value, and results if any, are returned to the sink.

```
<element name="SubscribeByServiceDataXPath">
  <complexType>
    <sequence>
      <element name="subscribeByServiceDataXPath"
type="tns:ServiceDataXPathSubscriptionExpressionType"/>
    </sequence>
  </complexType>
</element>

<complexType name="ServiceDataXPathSubscriptionExpressionType">
  <sequence>
    <element name="name" type="QName"/>
    <element name="xpath" type="string"/>
    <element name="namespaces" type="string"
      minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
</complexType>
```

NotificationSubscription PortType

NotificationSubscription grid service is created in response to a subscribe operation on a grid service that extends the NotificationSource PortType. The NotificationSubscription portType is defined in the OGSi Spec.

The NotificationSubscription has information on the subscription parameters – the notification sink, and the subscription expression. NotificationSubscription portType extends the GridService portType, and hence has the find, and lifecycle management operations.

NotificationSink PortType

Any service that needs to receive notification delivered to it from the NotificationSource, must extend and implment the NotificationSink PortType. The only operation that is defined on this portType is “deliverNotification”.

Notification Framework

NotificationSource in this framework is implemented as an OperationProvider – `org.globus.ogsa.impl.ogsi.JMSNotificationSourceProvider`. The class, and interaction diagrams for the framework is shown in the figures below.

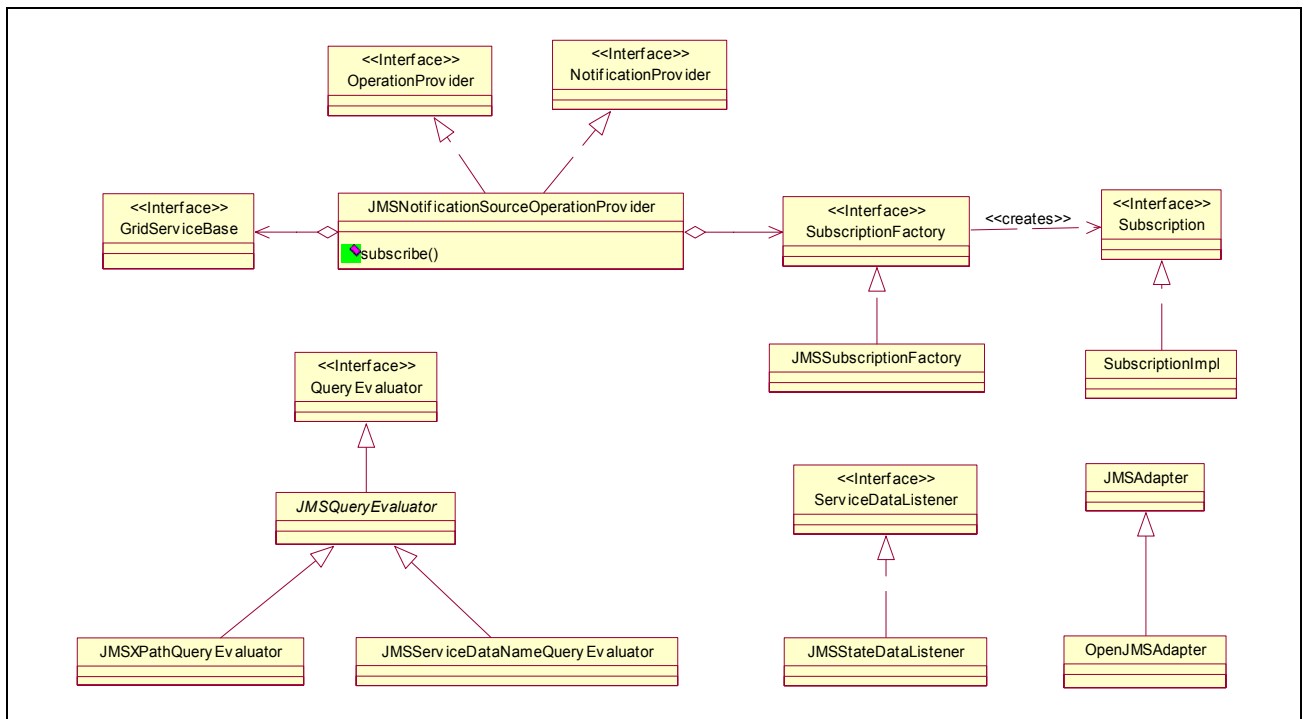


Figure 1. Notification Framework Class Diagram

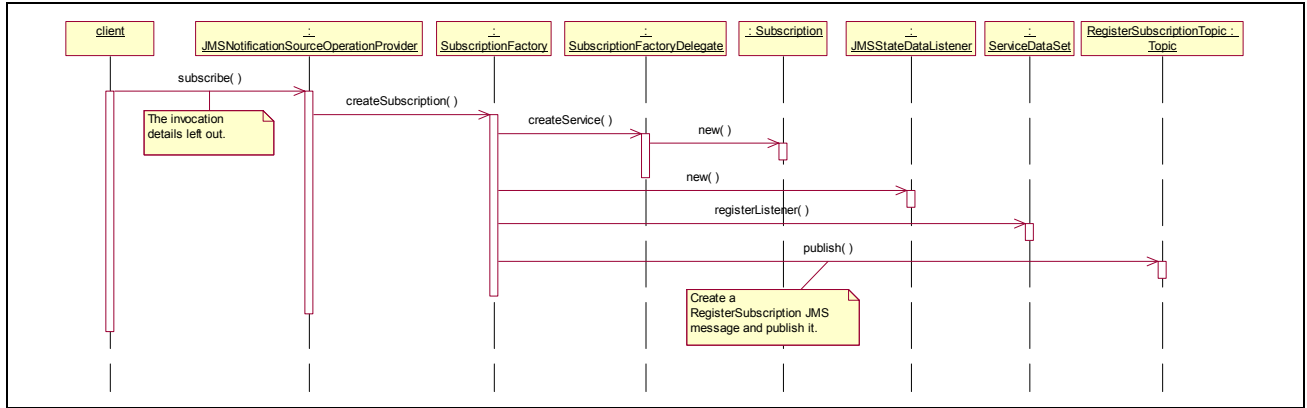


Figure 2. Subscribe operation

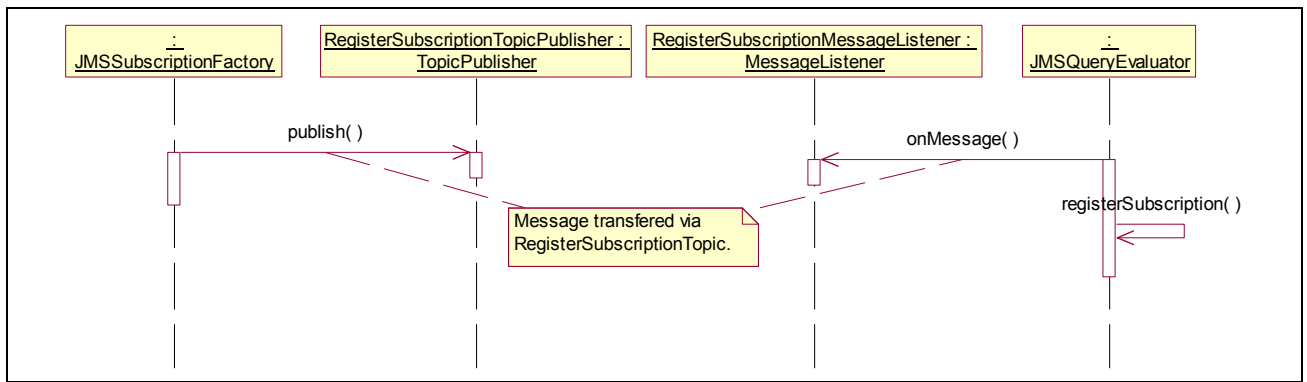


Figure 3. Register Subscription

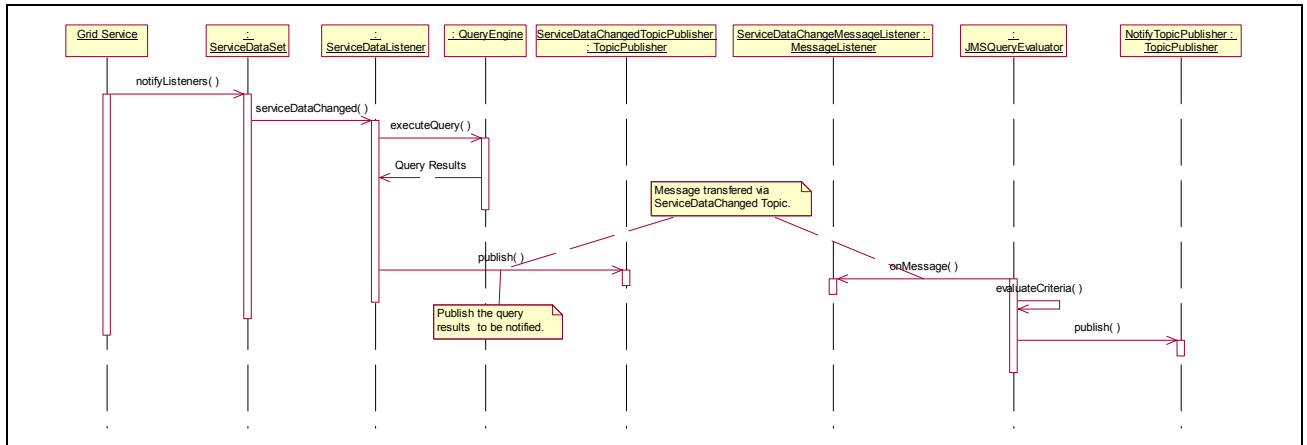


Figure 4. Service Data Change

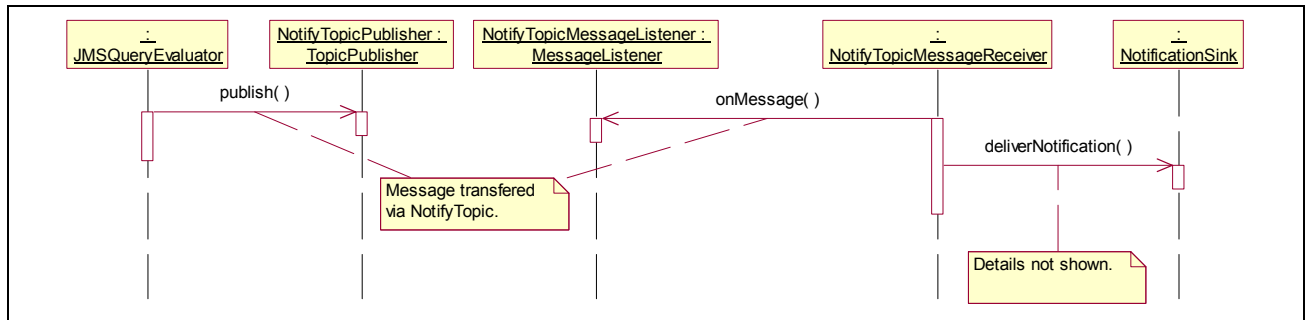


Figure 5. Notify

OpenJMS

The JMS Notification framework uses OpenJMS as the JMS Provider. OpenJMS is an open source project. More information on OpenJMS may be obtained from <http://openjms.sourceforge.net>. The version 0.7.4 of OpenJMS was used in the development and testing of the JMS Notification framework.

Using JMS Notification Source Operation Provider

To use JMS Notification Source Operation Provider, it is specified as the Operation Provider for Notification Source in the service's deployment descriptor. A sample deployment descriptor (extracted from the samples) is shown below. The relevant parts of this descriptor are highlighted.

```

<service name="samples/counter/notification/JMSCounterFactoryService"
  provider="Handler" style="wrapped">
  <parameter name="name"
    value="Notification Counter Factory"/>
  <parameter name="instance-name" value="Notification Counter"/>
  <parameter name="instance-schemaPath"
    value="schema/samples/counter/notification_counter_service.wsdl"/>
  <parameter name="instance-baseClassName"
    value="org.globus.ogsa.impl.samples.counter.notification.CounterNotificationServiceDataImpl"/>
  <parameter name="instance-operationProviders"
    value="org.globus.ogsa.impl.ogsi.JMSNotificationSourceProvider"/>
  <parameter name="instance-className"
    value="org.globus.ogsa.samples.counter.notification.NotificationCounterPortType"/>
  <parameter name="persistent"
    value="true"/>
  <parameter name="schemaPath"
    value="schema/ogsi/ogsi_notification_factory_service.wsdl"/>
  <parameter name="baseClassName"
    value="org.globus.ogsa.impl.ogsi.PersistentGridServiceImpl"/>
  <parameter name="handlerClass"
  
```

```

        value="org.globus.ogsa.handlers.RPCURIProvider"/>
<parameter name="className"
        value="org.gridforum.ogsi.NotificationFactory"/>
<parameter name="allowedMethods"
        value="*"/>
<parameter name="factoryCallback"
        value="org.globus.ogsa.impl.ogsi.DynamicFactoryCallbackImpl"/>
<parameter name="operationProviders"
        value="org.globus.ogsa.impl.ogsi.FactoryProvider
org.globus.ogsa.impl.ogsi.JMSNotificationSourceProvider"/>
</service>

```

Once the service is deployed, the user needs to start the OpenJMS server. This is accomplished by invoking the “startup” command in the OpenJMS bin directory. By default, it is set to use port 1099.

Subscribe

The code snippet below shows how to subscribe to notification in changes service data using the `SubscribeByNameType ExpressionType`.

```

//
// Use the SubscriptionHelper class to create the subscription expression.
// the service data name is "CounterUpdate"
//
QName sdQName = new QName("", "CounterUpdate");
SubscriptionHelper subscriptionHelper = new SubscriptionHelper();
ExtensibilityType expression = new ExtensibilityType();

expression = subscriptionHelper.getNameExpression(new QName[] { sdQName });

//
// get the notification source.
//
OGSIServiceGridLocator serviceLocator = new OGSIServiceGridLocator();
NotificationSource source =
    serviceLocator.getNotificationSourcePort(sourceHandle);

//
// now subscribe.
//
source.subscribe(subscriptionExpression, // subscription expression
                sinkLocator,           // ServiceLocator of the sink
                expirationTime,        // Requested termination time
                subscriptionLocator,    // Locator for the returned subscription
                termination);           // Termination time that was set

```

In order to subscribe using the `ServiceDataXPathSubscriptionExpressionType`, a valid xpath expression has to be provided. The code snippet below is an example how a xpath subscription expression is created using the subscription helper.

```

String[] namespaces = new String[] { "xmlns:ogsi=" + GridConstants.GLOBUS_NS };
String xpath = "/*";

```

```
expression = aSubscriptionHelper.getXPathExpression(sdQName xpath, namespaces);
```

Additionally, this framework also supports the min/max interval timers on the `SubscribeByNameType` expression. Refer to OGSi spec on the behavior of min/max intervals in `SubscribeByNameType`. The following code show an example of how this may set in the expression.

```
//  
// Set the Min and Max Intervals  
//  
Duration minDuration = new Duration();  
minDuration.setSeconds(10); // min interval is set to 10 secs.  
  
Duration maxDuration = new Duration();  
maxDuration.setSeconds(60); // max interval is set to 60 secs.  
  
MaxIntervalType maxInterval = new MaxIntervalType();  
maxInterval.setValue(maxDuration);  
  
SubscribeByNameType sbnt= (SubscribeByNameType)  
    AnyHelper.getAsSingleObject(  
        subscriptionExpression,  
        SubscribeByNameType.class);  
  
sbnt.setMaxInterval(maxInterval);  
sbnt.setMinInterval(minDuration);
```