

Functional Specification

Community Grids Lab, Indiana University

THIS DOCUMENT IS BASED ON REVISION 0.3 OF THE FUNCTION SPECIFICATION TEMPLATE!

1. Introduction

This software provides an implementation of the WS-Eventing specification that was released in August 2004. This specification -- developed jointly by Microsoft, BEA, IBM and Sun -- can be found at <http://www-128.ibm.com/developerworks/webservices/library/specification/ws-eventing/>. This particular specification provides abstractions for publish/subscribe interactions within the Web Services domain and the ability to route notifications to the registered entities. Please note that what we briefly describe in this section is a background and overview to the WS-Eventing specification. If one is looking for comprehensive details the best place is the specification. Any attempt to describe the spec in detail, will end up making this document similar to the specification: which would be pointless.

1.1 A background on notifications

There are two main entities involved in a notification: the source which is the generator of notifications and the sink which is interested in these notifications. A sink first needs to register its interest in a situation, this operation is generally referred to as a subscribe operation. The source first wraps occurrences into notification messages. Next, the source checks to see if the message satisfies the constraints specified in the previously registered subscriptions. If so, the source routes the message to the sink. This routing of the message from the source to the sink is referred to as a notification.

It should be noted that there could be multiple sources and sinks within the system. Furthermore, each sink could register its interests with multiple sources, while a given source can manage multiple sinks. The complexity of the subscriptions registered by a sink could vary from simple strings such as "Weather/Warnings" to complex XPath or SQL queries.

Typically a source comprises two distinct roles: producer and publisher. A producer is responsible for packing occurrences into notification messages, while the publisher is responsible for publishing these notifications. Similarly, a sink comprises two distinct roles: subscriber and consumer. The subscriber is responsible for registering the consumer's interests with a source, while the consumer is responsible for consuming notifications received from a source.

Depending on the nature of the underlying frameworks the coupling between the sources and sinks can vary. In loosely-coupled systems a source need not be aware of the sinks: the source generates events and an intermediary, typically a messaging middleware, is responsible for routing the message to appropriate sinks. In tightly-coupled systems there is no intermediary between the source and the sink.

1.2 Brief Overview of WS-Eventing

Figure 1 depicts the chief components in WS-Eventing. When the sink subscribes with the source, the source includes information regarding the subscription manager in its response. Subsequent operations -- such as getting the status of, renewing and unsubscribing -- pertaining to previously registered subscriptions are all directed to the subscription manager. The source sends both notifications and a message signifying the end of registered subscriptions to the sink.

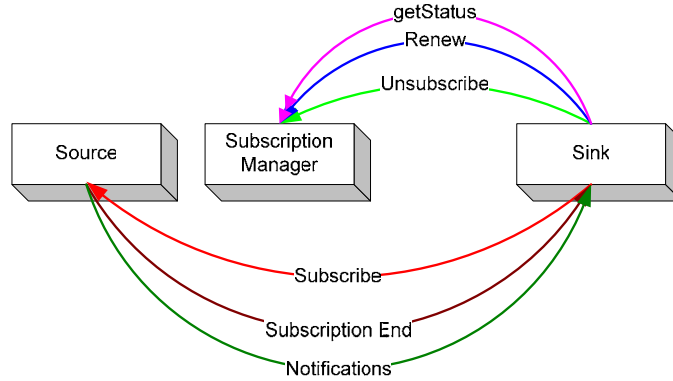


Figure 1: WS-Eventing - Chief components

This software provides an implementation of the WS-Eventing specification and provides support for all the abstractions that are part of this specification. The schemas that are relevant to this specification are enumerated below.

- WS-Eventing: <http://schemas.xmlsoap.org/ws/2004/08/eventing>
- SOAP: <http://schemas.xmlsoap.org/soap/envelope/>
- WS-Addressing: <http://schemas.xmlsoap.org/ws/2004/08/addressing>

2. Product Description

This software provides an implementation of the WS-Eventing specification. This specification abstracts exchanges between entities (consumers and producers) in the area of publish/subscribe systems. There are similar specifications in the J2EE environment viz. Java Message Service (JMS) and in the Web Services environment -- WS-Notification. The table below provides a comparison of WS-Eventing and WS-Notification.

Table 1: Comparison of WS-Notification and WS-Eventing

	WS-Notification	WS-Eventing
Related Specifications	SOAP, WS-Addressing, WS-BaseNotification, WS-Brokered Notification, WS-Topics, WS-Resource Properties and WS-ResourceLifetime	SOAP, WS-Addressing
Support for loosely coupled notifications. (Producers need not know consumers)	Yes. The intermediary called Notification Broker and the exchanges that need to be supported are defined in the WS-Brokered Notification specification.	No.
Delivery modes supported	Push	Push Batched, Pull, & Trap (udp) defined in WS-Management
Delegated Management of subscriptions	Yes. Through the subscription manager interface.	Yes. Through the subscription manager interface.
Support for replay like features	One can get last message to a topic. A sink can also retrieve message issued between the pausing and resumption of a subscription.	No. However WS-Management introduces notion of resume/pause subscriptions.

Subscription operations	Subscribe, Pause and Resume. (There is NO exchange to <i>unsubscribe</i>).	Subscribe, Renew, Unsubscribe and Subscription End.
Subscription termination notification	NO	Yes. There is a SubscriptionEnd notification that is sent out by the source to the sink anytime the subscription ends (either an unsubscribe or termination)
Support for filters on to narrow notifications	YES	YES
Subscription lifetimes	Defined using the WS-Resource Lifetime specification.	Contained within the Subscribe and Renew exchanges.
Notification filters and topic expressions supported	Topic Expressions supported: QName, "/" separated Strings, and XPath path expressions.	Filter supported is XPath.
Hierarchical topics and Wildcards support	Yes. Supports * and // wildcards for selection of topic descendants in a topic tree.	No.
Topic space management	Defined using WS-Topics. The topic space will also support exchanges as defined by the WS-ResourceProperties specification.	No formal recommendation regarding topic management.
Advertisement of supported topics	Yes. The NotificationProducer interface allows inspection of available topics.	No.
On demand publishing	YES. This is supported through the WS-Brokered Notification specification. This allows a publisher to publish ONLY if there is a consumer interested in receipt of notifications.	NA. A source always keeps information regarding the sinks, so on-demand publishing is the default mode.
Notification messages	Provides support for both a Notify message as well as "raw" application specific message,	Does not define any special Notification message type.
Retrieve information about Topics from producer	Yes. Also indicates if the set of topics is going to be dynamic.	NO.
Retrieve info about topic expression dialects	Yes.	NO
Suggested Security	WS-Security and assorted specifications.	WS-Security & assorted specifications.

WS-Eventing provides a framework for notification between 2 Web Service endpoints. All exchanges between communicating entities are encapsulated within SOAP messages. The WS-Eventing specification provides a platform-independent and language-independent framework for notifications. We are also in the process of implementing WS-Notification, which is a competing specification. In summary, this software provides a complete implementation of the WS-Eventing specification. Finally, to effectively use this software you should be familiar with Java and have some working knowledge of Web Services.

3. Use Cases

Publish/subscribe systems have been deployed in a wide-variety of application domains. This software can be deployed in settings where distributed entities need to be able to communicate with each other through the exchange of messages. Furthermore, entities will be able to register an interest in specific events/messages (through previously registered subscriptions) and have the system route these events as and when they occur.

The WS-Eventing specification delegates security related issues to the WS-Security specification, which facilitates message-level security. Nothing in the WS-Eventing specification or this implementation of the specification precludes the use of WS-Security. We fully expect this implementation to work with existing WS-Security implementations. The WS-Security specification provides the ability to securely (leveraging encryption, message digest and signing) route SOAP messages between endpoints irrespective of the underlying transport mechanism.

We will be using this implementation within the SERVO (Solid Earth Research Virtual Observatory) Grid <http://www.servogrid.org/> project. The software is quite robust since it deals with ALL the exception/fault conditions outlined within the WS-Eventing specification.

This software will report faults (as outlined in the WS-Eventing specification) if there are malformed requests. We enumerate some of these below

1. If the subscription identifier specified for subscription renewals is an incorrect one.
2. If the time specified for subscription renewal is in the past. For example if the correct time is New Year's Eve 2005, and the renewal request is for Christmas 2005, the request will fail and a fault reporting this problem will be thrown by the *subscription manager*.
3. If the subscription identifier specified in the unsubscribe request (or the renewal request) is an unknown one, a fault will be reported by the *subscription manager*.
4. If the Filter dialect specified in the subscription request is an invalid one the *source* will respond with a fault message outlining this problem.
5. If the expiration time specified in the subscription request is an invalid one the *source* will respond with a fault message outlining this problem.

The WS-Eventing specification, and this implementation, concerns itself only with the delivery of messages (notifications) from the source to the sinks that had previously registered their interests. As far as we know nothing in this software precludes its use by researchers, system administrators or project managers. What has been provided here is the underlying middleware, applications can be easily developed which hide the innards of this specification/implementation from the end-users.

4. Evaluation Metrics

We now include performance measurements from our experiments. These experiments were performed on a 3.5 GHz Pentium IV machine with Sun's 1.4.2 Java Virtual Machine. For each measurement we performed the experiment 100 times. An outlier removal program was used to remove outliers, if any, in the result set. For each run we also tracked the memory utilization. This was done by simply recording the memory utilization prior to the invocation of a specific operation and after the invocation. In some cases this calculation resulted in a negative utilization because of garbage collection (via the Java garbage collector thread) in the intervening period.

In our benchmarking we sought to measure the overheads involved in the creation of various exchanges that are part of the WS-Eventing specification. We have also reported measurements related to the creation of XMLBeans based SOAP Envelopes (Figure 2) and the creation of WS-Addressing Endpoint References both with and without `wsa:ReferenceProperties`. These costs have been depicted in Figure 3.

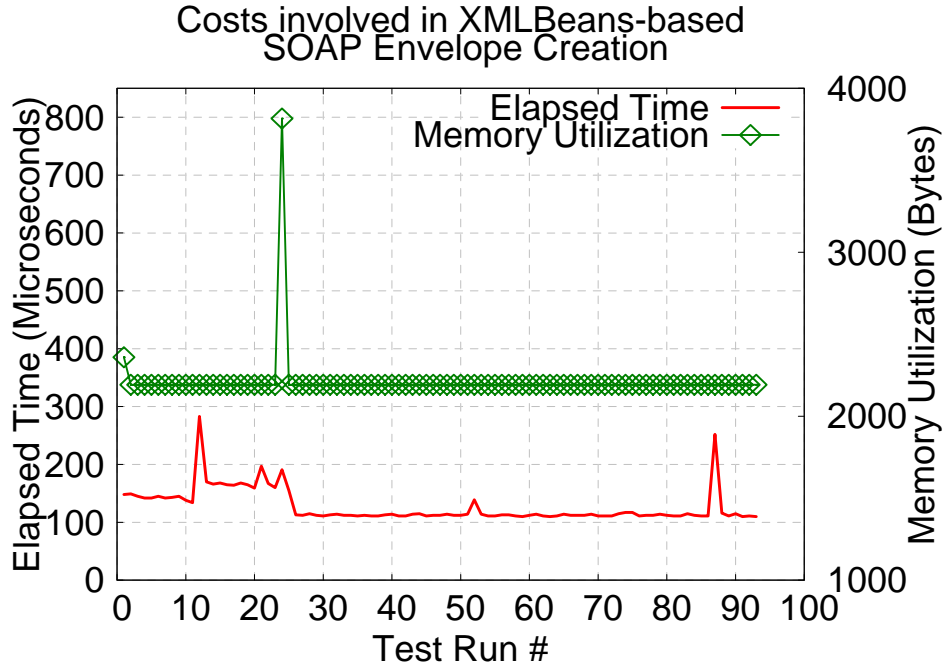


Figure 2: Costs involved in creation of XMLBeans based SOAP Envelope

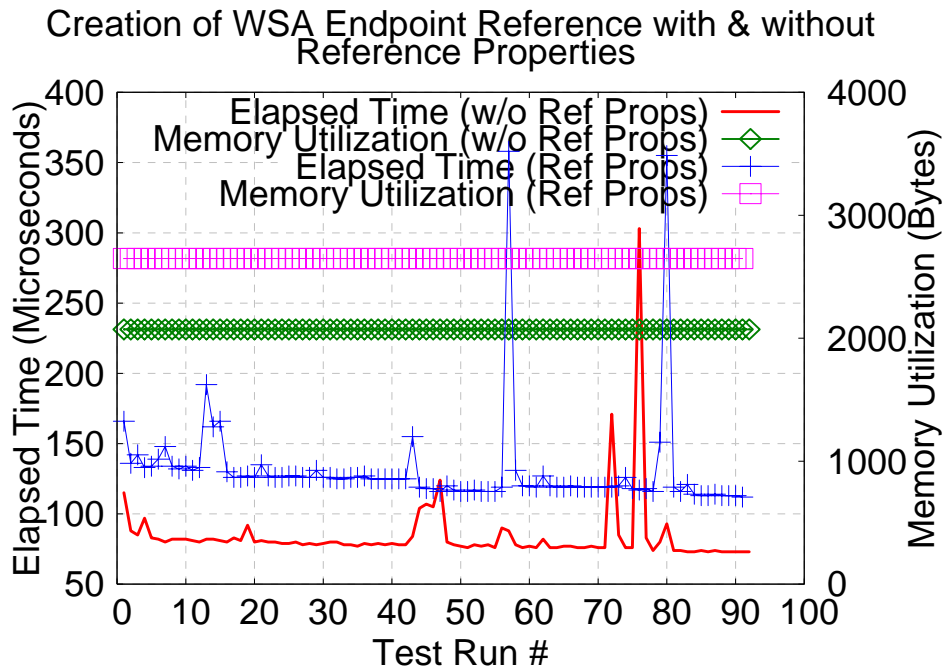


Figure 3: Creation of WS-Addressing Endpoint References

Figure 4 and Figure 5 depict the costs involved in the creation of WSE Subscription Requests and WSE Subscription responses respectively.

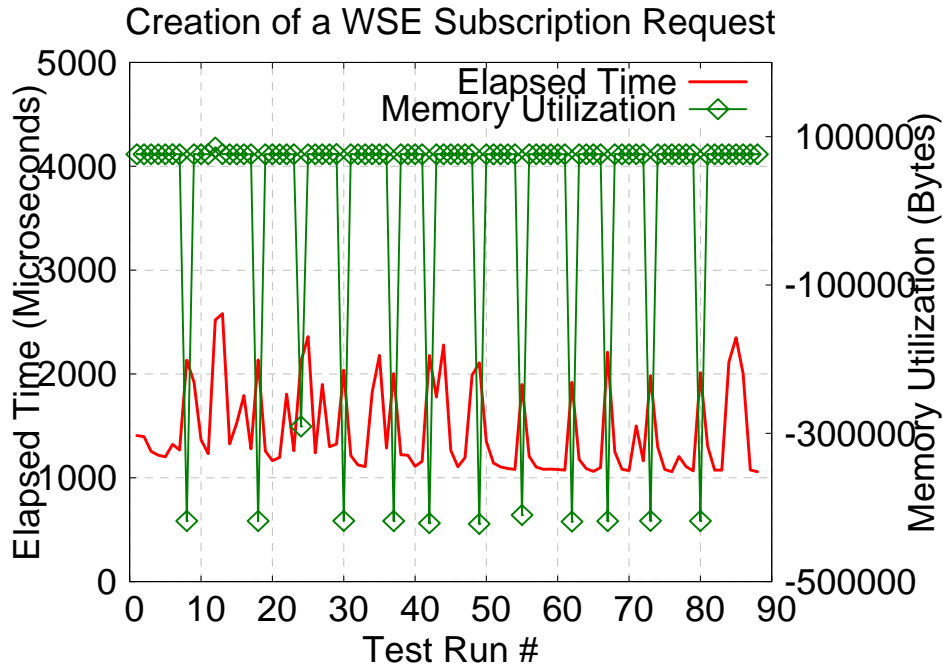


Figure 4: Creation of a WSE Subscription Request

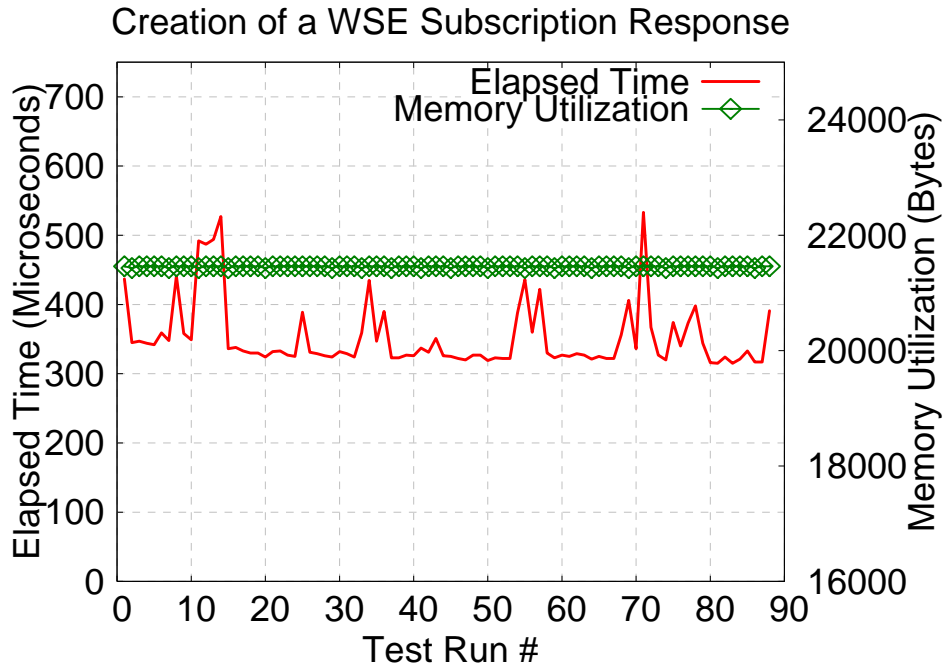


Figure 5: Creation of WSE Subscription Response

Figure 6 and Figure 7 depict the costs involved in the creation of an Unsubscribe Request and the corresponding response associated with it.

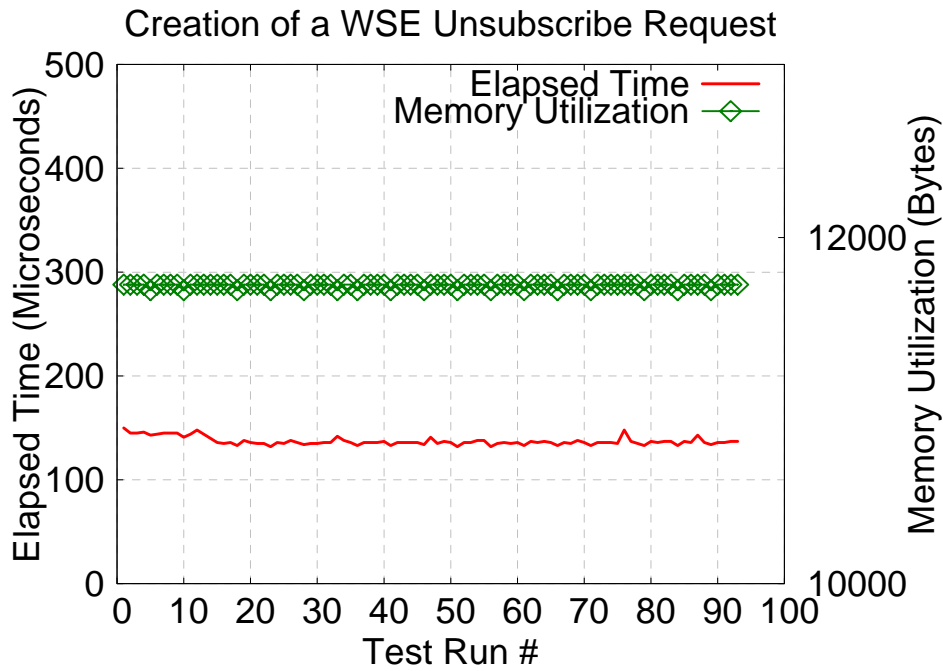


Figure 6: Creation of a WSE Unsubscribe request

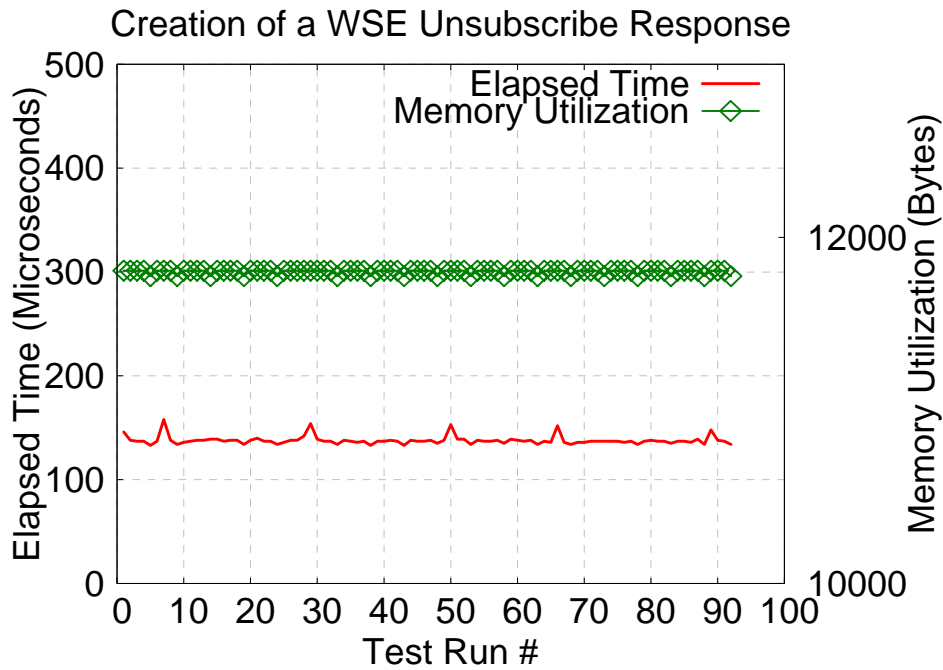


Figure 7: Creation of a WSE Unsubscribe Response

Figure 8 and Figure 9 depict the cost associated with the creation of a WSE Subscription Renewal request and the corresponding response associated with it.

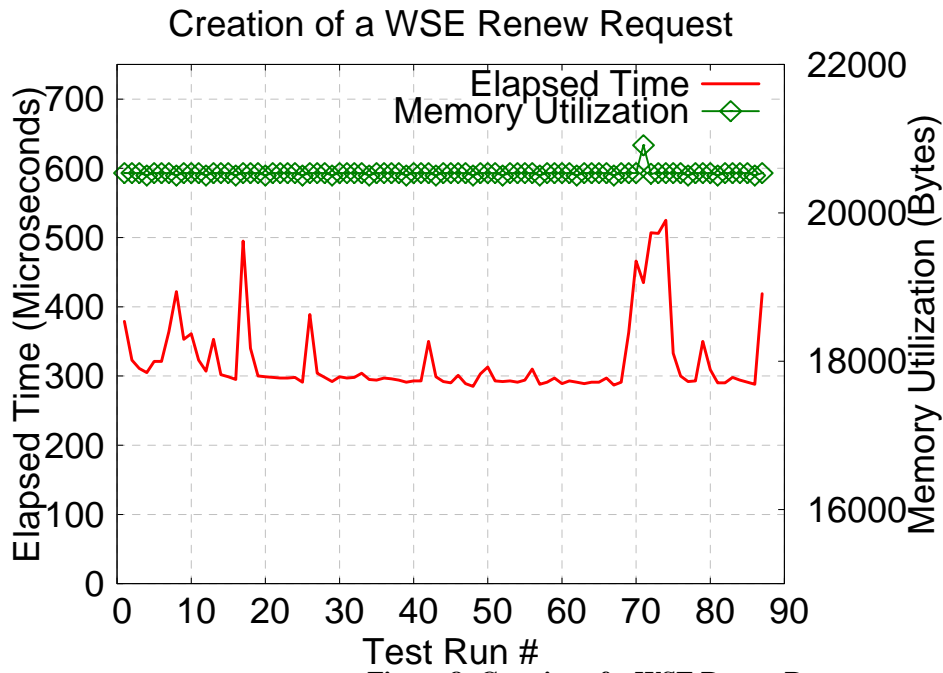


Figure 8: Creation of a WSE Renew Request

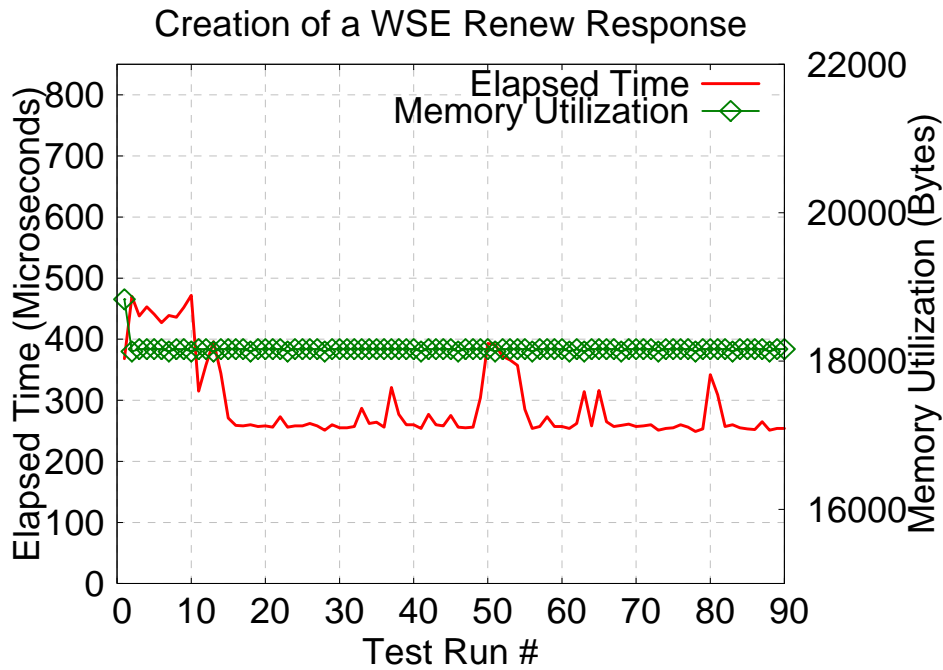


Figure 9: Creation of WSE Renew Response

Figure 10 and Figure 11 depict the costs associated with the creation of a WSE GetStatus Request and Response respectively.

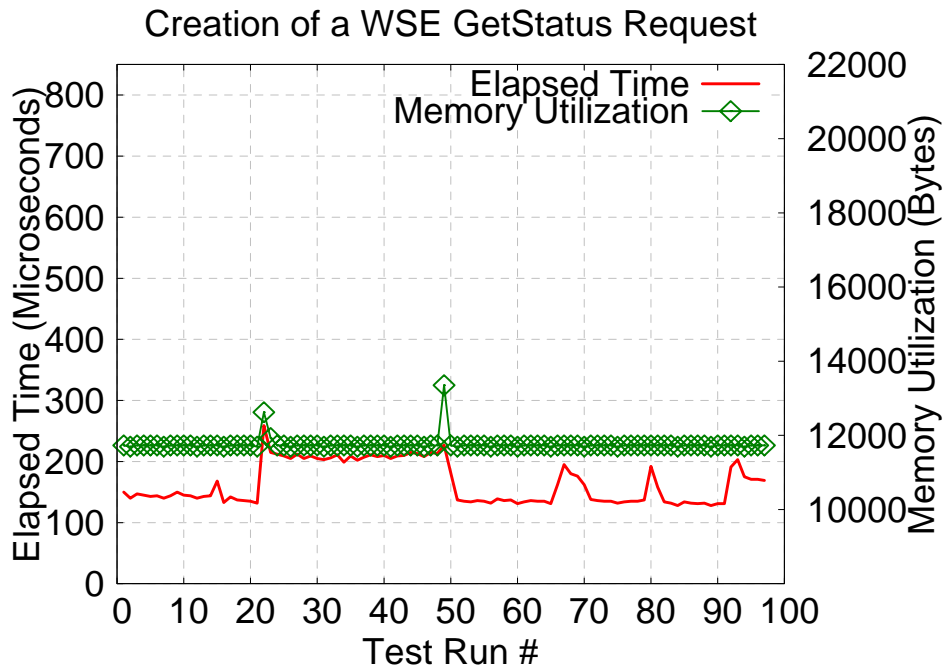


Figure 10: Creation of a WSE GetStatus Request

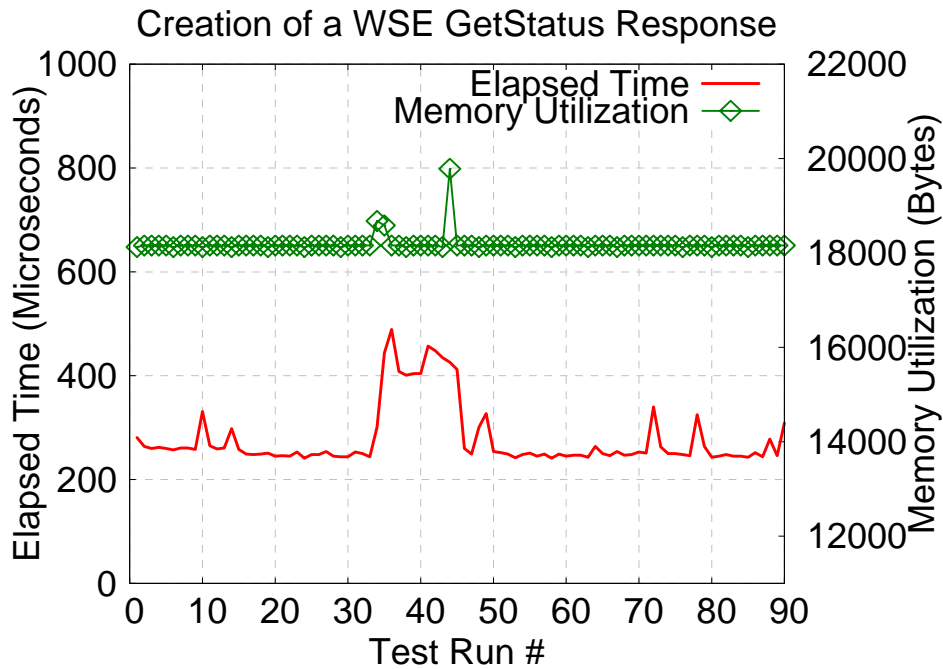


Figure 11: Creation of a GetStatus Response

Figure 12 depicts the cost associated with a SubscriptionEnd message.

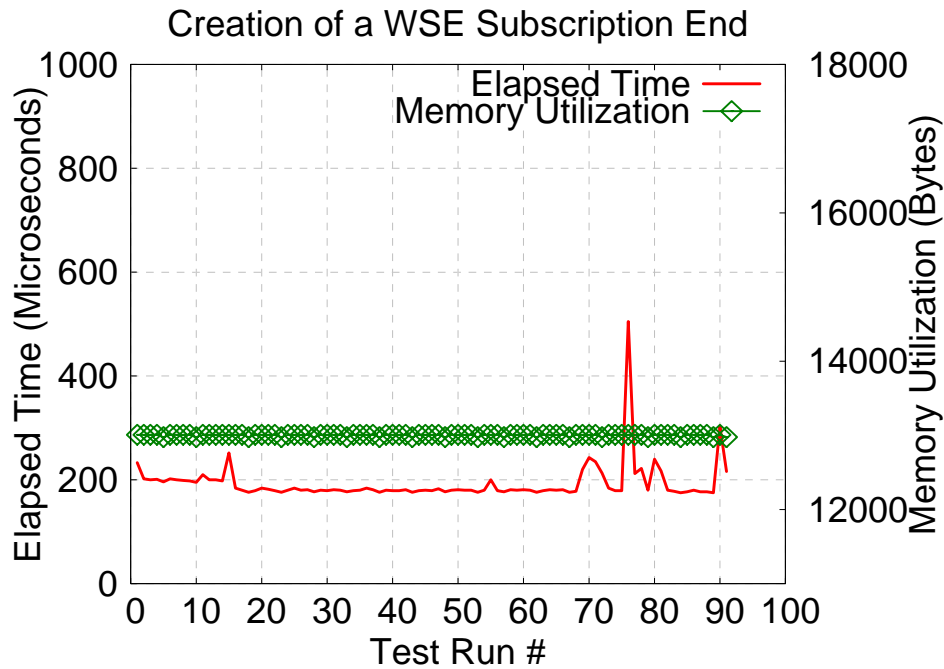


Figure 12: Creation of a WSE Subscription End message

Operation	Mean	Standard Deviation	Standard Error	Outliers	Min	Max	Memory Utilization (Bytes)
Create EnvelopeDocument	127.505	29.833	3.093	7	110	283	2192
Create SOAPMessage	85.150	64.851	6.724	7	34	488	1824
Create Epr	84.260	26.367	2.749	8	73	303	2072
Create EPR With ReferenceProperties	131.065	36.396	3.815	9	112	358	2648
Create Epr Envelope	364.955	158.310	16.687	10	261	834	7184
Create Epr Envelope With Most Wsa Fields	491.263	204.220	21.408	9	347	1099	13880
Create Subscribe Request	1466.068	436.588	46.540	12	1059	2581	76360
Create Subscribe Response	352.545	48.714	5.193	12	315	533	21464
Create Unsubscribe Request	137.290	3.908	0.405	7	132	150	11728
Create Unsubscribe Response	137.804	4.185	0.436	8	133	158	11808
Create Renew Request	321.0	54.462	5.839	13	285	525	20536
Create Renew Response	294.066	63.293	6.671	10	249	472	18160
Create GetStatus Request	164.463	34.121	3.464	3	128	259	11728
Create GetStatus Response	279.088	60.820	6.411	10	241	489	18160
Create SubscriptionEnd	193.307	39.088	4.097	9	175	505	13008

5. Functional Design Considerations

Assumptions that were made during functional design: The WS-Eventing specification leverages WS-Addressing and SOAP. These related technologies have a few schemas each corresponding to a different version of the aforementioned specifications. To be consistent with the javax.xml.soap.SOAPMessage format we used the SOAP schema specified at <http://schemas.xmlsoap.org/soap/envelope/> . At the time the software was written (Dec 2004) we used the latest WS-Addressing schema that was available at that time -- <http://schemas.xmlsoap.org/ws/2004/08/addressing>. Here we note that a new version of the WS-Addressing schema released in March 2005 is now available.

Prerequisites for the correct working of the product: This software has been written in Java. For the correct working of this software one needs to use JDK 1.4 or higher.

Main decisions/reasons regarding functionality: This software is an implementation of the WS-Eventing specification. The decisions/reasons were driven by the considerations involved in the implementation of this specification. One of the major decisions was the choice of schema - we discussed this in the assumptions section. We also decided to support topic ("/" separated Strings) and regular-expressions based subscriptions in addition to the XPath format mandated by the WS-Eventing specification. The rationale for this was that there might be users used to JMS who would prefer the overall simplicity of topic based subscriptions.

Scale of deployment: The three major components have been deployed within the OMII Containers (both 1.2 and the recently released 2.0 specification). The source, sink, and subscription manager have all been tested in various distributed settings (all in the same machine, 2 in the same machine and 1 on another, and finally all components in different machines).

Resource requirements in terms of hardware, data volume, other software or equipment: All you need is a machine that has a JVM for it. For the correct working of this software one needs to use JDK 1.4 or higher.

Portability: This software is written in Java. So, this will work on any machine/operating-system that has a Java Virtual Machine for it. As mentioned previously one needs to use JDK 1.4 or higher.

Reliability/Maintainability/Availability: The WS-Eventing specification does not deal with reliable messaging. We have also implemented two specifications related to reliable messaging viz. WS-Reliability and WS-ReliableMessaging. This software will be released separately as part of the **FIRMS** release in the coming weeks.

Installation: We wanted the software to be easy to use and install. We have included ANT scripts which facilitate the deployment of the software in a variety of settings.

Security: As mentioned previously WS-Eventing delegates security issues to WS-Security. The WS-Security specification provides the ability to securely route SOAP messages between endpoints irrespective of the underlying transport mechanism. Nothing in the WS-Eventing (or this implementation of the spec) precludes the use of WS-Security.

Configuration and customization: This software is an implementation of a specification. We have provided scripts for deployment of the software in various settings.

Error handling: This software will report faults if there are problems with any of the exchanges outlined in the WS-Eventing specification. We enumerated some of these faults in an earlier section (section 3.0) of this document. The API for this software also facilitates the creation of appropriate requests and responses. These methods also reports if there are problems in any of the parameters involved in the method invocation.

6. List Of Functions

This section provides a description of the core classes, and their methods, within the FINS software.

Package: `cgl.narada.wsinfra.wse`

public class WseActions

This class provides a one-stop for all the URIs related to WS-Addressing Action URIs used within WS-Eventing. This eliminates the need to hard-code this in several places. Furthermore, if the spec changes the impact of this change will be felt at far fewer places.

public static WseActions getInstance()

Description	This will return instance of this class
Input arguments Process	
Output	Object of WseActions
Exceptions	
Comments	This will return single instance of the class.

public String getSubscribe()

Description	Retrieve the String action associated with the exchange
Input arguments Process	
Output	String value of wseSubscribe
Exceptions	
Comments	

public static String getSubscribeResponse()

Description	Retrieve the String action associated with the exchange
Input arguments Process	
Output	String value of wseSubscribeResponse
Exceptions	
Comments	

public static String getRenewResponse()

Description	Retrieve the String action associated with the exchange
Input arguments Process	
Output	String value of wseRenewResponse
Exceptions	
Comments	

public String getGetStatus()

Description	Retrieve the String action associated with the exchange
Input arguments Process	
Output	String value of wseGetStatus
Exceptions	
Comments	

public String getGetStatusResponse()

Description	Retrieve the String action associated with the exchange
Input arguments Process	
Output	String value of wseGetStatusResponse
Exceptions	
Comments	

public String getUnsubscribe()

Description	Retrieve the String action associated with the exchange
Input arguments Process	
Output	String value of wseUnsubscribe
Exceptions	
Comments	

public String getUnsubscribeResponse()

Description	Retrieve the String action associated with the exchange
Input arguments Process	
Output	String value of wseUnsubscribeResponse
Exceptions	
Comments	

public String getSubscriptionEnd()

Description	Retrieve the String action associated with the exchange
Input arguments Process	
Output	String value of wseSubscriptionEnd
Exceptions	
Comments	

public boolean isValidEventingAction(String actionToCheck)

Description	Check to see if this is one of the registered WS-Eventing actions
Input arguments Process	String actionToCheck
Output	Boolean value
Exceptions	
Comments	

public ActionDocument getSubscribeAction()

Description	Retrieve the ActionDocument associated with the exchange
Input arguments Process	
Output	ActionDocument value of wse getSubscribeAction
Exceptions	
Comments	

public ActionDocument getSubscribeResponseAction()

Description	Retrieve the ActionDocument associated with the exchange
Input arguments Process	
Output	ActionDocument value of wseSubscribeResponseAction

Exceptions	
Comments	

public ActionDocument getRenewAction()

Description	Retrieve the ActionDocument associated with the exchange
Input arguments Process	
Output	ActionDocument value of getRenewAction
Exceptions	
Comments	

public ActionDocument getRenewResponseAction()

Description	Retrieve the ActionDocument associated with the exchange
Input arguments Process	
Output	ActionDocument value of wseRenewResponseAction
Exceptions	
Comments	

public ActionDocument getGetStatusAction()

Description	Retrieve the ActionDocument associated with the exchange
Input arguments Process	
Output	ActionDocument value of wseGetStatusAction
Exceptions	
Comments	

public ActionDocument getGetStatusResponseAction()

Description	Retrieve the ActionDocument associated with the exchange
Input arguments Process	
Output	ActionDocument value of wseGetStatusResponseAction
Exceptions	
Comments	

public ActionDocument getUnsubscribeAction()

Description	Retrieve the ActionDocument associated with the exchange
Input arguments Process	
Output	ActionDocument value of wseUnsubscribeAction
Exceptions	
Comments	

public ActionDocument getUnsubscribeResponseAction()

Description	Retrieve the ActionDocument associated with the exchange
Input arguments Process	
Output	ActionDocument value of wseUnsubscribeResponseAction
Exceptions	
Comments	

public ActionDocument getSubscriptionEndAction()

Description	Retrieve the ActionDocument associated with the exchange
Input arguments	
Process	
Output	ActionDocument value of wseSubscriptionEndAction
Exceptions	
Comments	

private ActionDocument prepareActionDocument(String wsaAction)

Description	Retrieve the ActionDocument associated with the exchange
Input arguments	String value of wsaAction
Process	
Output	ActionDocument value of wseUnsubscribeResponseAction
Exceptions	
Comments	

public interface WseElementCreation

This is a utility class which is responsible for the creation of elements that are used within WS-Eventing.

public SubscribeDocument newSubscribeDocument(String deliveryMode, EndpointReferenceType endTo, String filterDialect, String filterConstraint, Calendar expiresAt, EndpointReferenceType notifyTo) throws ProcessingException

Description	This method creates a subscribe request document based on the specified arguments
Input arguments	String deliveryMode, EndpointReferenceType endTo, String filterDialect, String filterConstraint, Calendar expiresAt, EndpointReferenceType notifyTo
Process	It checks NotifyTo element otherwise throws exception and adds all other parameters to SubscribeDocument.
Output	SubscribeDocument
Exceptions	ProcessingException
Comments	

public SubscribeDocument newSubscribeDocument(String filterDialect, String filterConstraint, Calendar expiresAt, EndpointReferenceType notifyTo) throws ProcessingException

Description	This method creates a subscribe request document based on the specified arguments.
Input arguments	String filterDialect, String filterConstraint, Calendar expiresAt, EndpointReferenceType notifyTo
Process	It adds deliveryMode and endTo elements and calls previous method.
Output	SubscribeDocument
Exceptions	ProcessingException
Comments	

public RenewDocument newRenewDocument(Calendar expiresAt) throws ProcessingException

Description	This method creates a renew request document based on the
--------------------	---

	newly created subscription entry
Input arguments	Calendar expiresAt
Process	It checks for expiresAt element for null value and throws exception.
Output	RenewDocument
Exceptions	ProcessingException
Comments	

public GetStatusDocument newGetStatusDocument()

Description	This method creates a getStatus request document based on the newly created subscription entry
Input arguments	
Process	
Output	GetStatusDocument
Exceptions	
Comments	

public SubscribeResponseDocument newSubscribeResponseDocument(SubscriptionEntry subscriptionEntry) throws ProcessingException

Description	This method creates a subscribe response document based on the newly created subscription entry
Input arguments	SubscriptionEntry subscriptionEntry
Process	Checks for SubscriptionEntry for problems and returns new Subscription Response Document
Output	SubscribeResponseDocument
Exceptions	ProcessingException
Comments	

public RenewResponseDocument newRenewResponseDocument(SubscriptionEntry subscriptionEntry) throws ProcessingException

Description	This method creates a renew response document based on the newly created subscription entry
Input arguments	SubscriptionEntry subscriptionEntry
Process	Check for Subscription entry for problems and creates new Renew Response Document.
Output	RenewResponseDocument
Exceptions	ProcessingException
Comments	

public GetStatusResponseDocument newGetStatusResponseDocument(SubscriptionEntry subscriptionEntry) throws ProcessingException

Description	This method creates a getStatus response document based on the newly created subscription entry
Input arguments	SubscriptionEntry subscriptionEntry
Process	Check for Subscription Entry for problems and creates new GetStatus Response Document
Output	GetStatusResponseDocument
Exceptions	ProcessingException

Comments	
-----------------	--

public SubscriptionEndDocument newSubscriptionEndDocument(SubscriptionEntry subscriptionEntry, String status, String reason) throws ProcessingException

Description	This method create a SubscriptionEnd document based on the specified subscription entry
Input arguments	SubscriptionEntry subscriptionEntry, String status, String reason
Process	Checks the Subscription Entry for problems, reason for not null. It will return new Subscription end document
Output	SubscriptionEndDocument
Exceptions	ProcessingException
Comments	

public interface WseRequestCreator

This is a utility class which is used to create requests based on the specified parameters. The requests created by this utility class include

- Subscribe
- GetStatus
- Renew
- Unsubscribe

public EnvelopeDocument createSubscribe(EndpointReferenceType sourceEpr, EndpointReferenceType sinkEpr, SubscribeDocument subscribeDocument) throws ProcessingException

Description	Create the SOAP envelope with the specified subscribe request. This envelope's destination is the Sink which generated the original request message.
Input arguments	EndpointReferenceType sourceEpr, EndpointReferenceType sinkEpr, SubscribeDocument subscribeDocument
Process	Check for sourceEpr Null values and creates subscribe EnvelopeDocument
Output	EnvelopeDocument
Exceptions	ProcessingException
Comments	This method is called from Sink Client

public EnvelopeDocument createRenew(EndpointReferenceType subscriptionManagerEpr, EndpointReferenceType sinkEpr, RenewDocument renewDocument) throws ProcessingException

Description	Create the SOAP envelope with the specified renew request. This envelope's destination is the sink which originated the message.
Input arguments	EndpointReferenceType subscriptionManagerEpr, EndpointReferenceType sinkEpr, RenewDocument renewDocument
Process	Check for subscriptionManagerEpr for null values and creates renew Envelope document
Output	EnvelopeDocument
Exceptions	ProcessingException

Comments	This method is called from Sink Client
-----------------	--

```
public EnvelopeDocument createGetStatus(EndpointReferenceType
subscriptionManagerEpr, EndpointReferenceType sinkEpr, GetStatusDocument
getStatusDocument) throws ProcessingException
```

Description	This method create a SubscriptionEnd document based on the specified subscription entry
Input arguments	EndpointReferenceType subscriptionManagerEpr, EndpointReferenceType sinkEpr, GetStatusDocument getStatusDocument
Process	Check for subscriptionManagerEpr for null values and creates get Status Envelope Document
Output	EnvelopeDocument
Exceptions	ProcessingException
Comments	This method is called from Sink Client

```
public EnvelopeDocument createUnsubscribe(EndpointReferenceType
subscriptionManagerEpr, EndpointReferenceType sinkEpr) throws
ProcessingException
```

Description	Create the SOAP envelope with the specified getStatus request. This envelope's destination is the sink which originated the message.
Input arguments	EndpointReferenceType subscriptionManagerEpr, EndpointReferenceType sinkEpr
Process	Check for subscriptionMnagerEpr for null values and creates Unsubscribe Envelope Document.
Output	EnvelopeDocument
Exceptions	ProcessingException
Comments	This method is called from Sink Client

public interface WseResponseCreator

This is a utility class which creates various responses based on the specified parameters. This is typically used by the source and the subscriptionManager.

```
public EnvelopeDocument createSubscribeResponse(EndpointReferenceType sinkEpr,
EndpointReferenceType sourceEpr, RelatesToDocument relatesTo,
SubscribeResponseDocument subscribeResponseDocument) throws
ProcessingException
```

Description	Create the SOAP envelope with the specified subscribe response. This envelope's destination is the Sink which generated the original request message.
Input arguments	EndpointReferenceType sinkEpr, EndpointReferenceType sourceEpr, RelatesToDocument relatesTo, SubscribeResponseDocument subscribeResponseDocument
Process	Check for sinkEpr null values and creates subscribe response to Sink Envelope Document
Output	EnvelopeDocument
Exceptions	ProcessingException
Comments	This method is called from Source

```
public EnvelopeDocument createSubscribeResponseToSM(EndpointReferenceType
subscriptionManagerEpr, EndpointReferenceType sourceEpr, SubscribeDocument
subscribeDocument, SubscribeResponseDocument subResponseDocument) throws
ProcessingException
```

Description	Create the SOAP envelope with the specified subscribe response AND subscribe original request. This envelope's destination is the SubscriptionManager for the source in question.
Input arguments	EndpointReferenceType subscriptionManagerEpr, EndpointReferenceType sourceEpr, SubscribeDocument subscribeDocument, SubscribeResponseDocument subResponseDocument
Process	Check for SubscriptionManagerEpr for null values and creates subscribe response to Subscription Manager Envelope Document
Output	EnvelopeDocument
Exceptions	ProcessingException
Comments	This method is called from Source

```
public EnvelopeDocument createRenewResponse(EndpointReferenceType sinkEpr,
EndpointReferenceType subscriptionManagerEpr, RelatesToDocument relatesTo,
RenewResponseDocument renewResponseDocument) throws ProcessingException
```

Description	Create the SOAP envelope with the specified renew response. This envelope's destination is the sink which originated the message.
Input arguments	EndpointReferenceType sinkEpr, EndpointReferenceType subscriptionManagerEpr, RelatesToDocument relatesTo, RenewResponseDocument renewResponseDocument
Process	Check for sinkEpr for null values and creates renew response to Sink Envelope Document.
Output	EnvelopeDocument
Exceptions	ProcessingException
Comments	This method is called from Subscription Manager

```
public EnvelopeDocument createRenewResponseToSource(EndpointReferenceType
sourceEpr, EndpointReferenceType subscriptionManagerEpr, String
subscriptionIdentifier, RenewResponseDocument renewResponseDocument) throws
ProcessingException
```

Description	Create the SOAP envelope with the specified renew response. This envelope's destination is the source which notifies the relevant sink.
Input arguments	EndpointReferenceType sourceEpr, EndpointReferenceType subscriptionManagerEpr, String subscriptionIdentifier, RenewResponseDocument renewResponseDocument
Process	Check for SourceEpr for null values and creates renew response document to Source Envelope Document
Output	EnvelopeDocument
Exceptions	ProcessingException
Comments	This method is called from Subscription Manager.

```
public EnvelopeDocument createGetStatusResponse(EndpointReferenceType sinkEpr,
EndpointReferenceType subscriptionManagerEpr, RelatesToDocument relatesTo,
GetStatusResponseDocument getStatusResponseDocument) throws
ProcessingException
```

Description	Create the SOAP envelope with the specified getStatus response. This envelope's destination is the sink which originated the message.
Input arguments	EndpointReferenceType sinkEpr, EndpointReferenceType subscriptionManagerEpr, RelatesToDocument relatesTo, GetStatusResponseDocument getStatusResponseDocument
Process	Check for sinkEpr for null values and creates getStatus response to Sink Envelope Document.
Output	EnvelopeDocument
Exceptions	ProcessingException
Comments	This method is called from Subscription Manager.

```
public EnvelopeDocument createUnsubscribeResponse(EndpointReferenceType
sinkEpr, EndpointReferenceType subscriptionManagerEpr, RelatesToDocument
relatesTo) throws ProcessingException
```

Description	Create the SOAP envelope with the specified getUnsubscribe response. This envelope's destination is the sink which originated the message.
Input arguments	EndpointReferenceType sinkEpr, EndpointReferenceType subscriptionManagerEpr, RelatesToDocument relatesTo
Process	Check for sinkEpr for null values and creates unsubscribe response Envelope Document to Sink
Output	EnvelopeDocument
Exceptions	ProcessingException
Comments	This method is called from Subscription Manager

```
public EnvelopeDocument
createUnsubscribeResponseToSource(EndpointReferenceType sourceEpr,
EndpointReferenceType subscriptionMgrEpr, String subscriptionIdentifier)
throws ProcessingException
```

Description	Create the SOAP envelope with the specified getStatus response. This envelope's destination is the source for the sink in question.
Input arguments	EndpointReferenceType sourceEpr, EndpointReferenceType subscriptionMgrEpr, String subscriptionIdentifier
Process	Check for sourceEpr for null values and creates unsubscribe response Document to Source
Output	EnvelopeDocument
Exceptions	ProcessingException
Comments	This method is called from Subscription Manager

```
public EnvelopeDocument createSubscriptionEnd(EndpointReferenceType
destinationEpr, EndpointReferenceType sourceEpr, SubscriptionEndDocument
subscriptionEndDocument) throws ProcessingException
```

Description	Creates a SOAP envelope with the subscription end document
Input arguments	EndpointReferenceType destinationEpr, EndpointReferenceType sourceEpr, SubscriptionEndDocument subscriptionEndDocument

Process	Checks for destinationEpr for null values and creates subscription end Envelope Document
Output	EnvelopeDocument
Exceptions	ProcessingException
Comments	This method is called from Subscription Manager.

public EndpointReferenceType getSendResponseTo (AddressingHeaders requestHeaders, EndpointReferenceType alternateEPR, String requestType) throws ProcessingException

Description	Try and the initialize the EPR to send a response message to. This is typically used to initialize the [wsa:To] element. The addressing headers that are passed here are the headers that were constructed based on the original request that came in previously. So this method will look for the [wsa:ReplyTo] or [wsa:From] elements to construct the responses.
Input arguments	AddressingHeaders requestHeaders, EndpointReferenceType alternateEPR, String requestType
Process	Check '[was:To]' and 'alternateEpr' for null values and sends either 'to' if not null or alternateEpr based on request AddressingHeaders.
Output	EndpointReferenceType
Exceptions	ProcessingException
Comments	This method is called form Source and Subscription Manager

public RelatesToDocument getResponseRelatesTo (AddressingHeaders requestHeaders)

Description	Initializes the relates to element appropriately, based on whether there is a messageID in the requestHeaders. This will return a NULL if there is no MessageID within the AddressingHeaders
Input arguments	AddressingHeaders requestHeaders
Process	It checks for messageID in AddressingHeaders and returns 'relatesTo' is messageID is present.
Output	RelatesToDocument
Exceptions	
Comments	This method is called from Source and Subscription Manger Processors.

Package: cgl.narada.wsinfra.wse.impl

public class SourceSubscribeRequestProcessing

Class which implements functionality related to processing subscription requests received at a source.

Following methods are in the SourceSubscribeRequestProcessing class

public static SourceSubscribeRequestProcessing getInstance ()

Description	It will returns the instance of this class
Input arguments	

Process	This method will return the single instance of the class.
Output	SourceSubscribeRequestProcessing
Exceptions	
Comments	Constructor of this class is declares as private.

```
public SubscriptionEntry processSubscribeRequest(SubscribeDocument
subscribeDocument, SubscriptionManagement subscriptionManagement,
EndpointReferenceType sourceEpr, EndpointReferenceType subscriptionManagerEpr)
throws WsFaultException
```

Description	The method to process a SubscriptionRequest issued by a Sink. If this request is successfully processed a SubscriptionEntry object is returned. If the request is NOT successful a WsFaultException is thrown.
Input arguments	SubscribeDocument, SubscriptionManagement, EndpointReferenceType sourceEpr, EndpointReferenceType subscriptionManagerEpr
Process	
Output	SourceSubscribeRequestProcessing
Exceptions	WsFaultException
Comments	

```
private SubscribeResponseDocument
createSubscribeResponseDocument(SubscriptionEntry subscribeEntry)
```

Description	This method create a subscribe response document based on the newly create subscription entry
Input arguments	SubscriptionEntry
Process	
Output	SubscribeResponseDocument
Exceptions	
Comments	

```
public EnvelopeDocument createEnvelopedResponse(AddressingHeaders
requestHeaders, SubscribeResponseDocument subscribeResponseDocument)
```

Description	This method creates the appropriate SOAP envelope for the subscription response. It takes care to ensure that the appropriate Action headers, RelatesTo, MessageId, To and From Headers are appropriately filled
Input arguments	AddressingHeaders, SubscribeResponseDocument
Process	
Output	EnvelopeDocument
Exceptions	
Comments	

```
public EnvelopeDocument createEnvelopeForSubManager(EndpointReferenceType
sourceEpr, EndpointReferenceType subscriptionManagerEpr, SubscribeDocument
subscribeDocument, SubscribeResponseDocument subResponseDocument)
```

Description	This method creates the appropriate SOAP envelope for the
--------------------	---

	subscription response. It takes care to ensure that the appropriate Action headers, RelatesTo, MessageId, To and From Headers are appropriately filled.
Input arguments	EndpointReferenceType sourceEpr, EndpointReferenceType subscriptionManagerEpr, SubscribeDocument subscribeDocument, SubscribeResponseDocument subResponseDocument
Process	
Output	EnvelopeDocument
Exceptions	
Comments	

private EndpointReferenceType initializeTo(AddressingHeaders requestHeaders)

Description	Computes the destination to send a Message To based on the specified Addressing Headers
Input arguments	AddressingHeaders
Process	
Output	EndpointReferenceType
Exceptions	
Comments	

public class WseNodeUtilsImpl extends WseNodeUtils

A utility class which retrieves the subscription identifier associated with an exchange. The method has to throw exceptions, if either the identifier is null or is not present. Furthermore, the identifier should also be a valid subscription maintained within subscription tables.

public static WseNodeUtils getInstance()

Description	This will return instance of WseNodeUtils class
Input arguments	
Process	
Output	WseNodeUtils
Exceptions	
Comments	

public SubscribeDocument getSubscribeDocument(EnvelopeDocument envelopeDocument) throws WsFaultException

Description	This method retrieves the SubscribeDocument element from the BODY of the SOAP envelope. The method has to throw exceptions, if either the element is null or is not present
Input arguments	EnvelopeDocument
Process	It checks for subscribeDocument for null values
Output	SubscribeDocument
Exceptions	WsFaultException
Comments	

public SubscribeResponseDocument getSubscribeResponseDocument(EnvelopeDocument envelopeDocument) throws WsFaultException

Description	This method retrieves the SubscribeResponseDocument element from the BODY of the SOAP envelope. The method has to throw exceptions, if either the element is null or is not present
Input arguments	EnvelopeDocument

Process	It checks for null values.
Output	SubscribeResponseDocument
Exceptions	WsFaultException
Comments	

public RenewDocument getRenewDocument(EnvelopeDocument envelopeDocument) throws WsFaultException

Description	This method retrieves the RenewDocument element from the BODY of the SOAP envelope. The method has to throw exceptions, if either the element is null or is not present.
Input arguments Process	EnvelopeDocument
Output	RenewDocument
Exceptions	WsFaultException
Comments	

public RenewResponseDocument getRenewResponseDocument(EnvelopeDocument envelopeDocument) throws WsFaultException

Description	This method retrieves the RenewResponseDocument element from the BODY of the SOAP envelope. The method has to throw exceptions, if either the element is null or is not present
Input arguments Process	EnvelopeDocument
Output	RenewResponseDocument
Exceptions	WsFaultException
Comments	

public GetStatusDocument getGetStatusDocument(EnvelopeDocument envelopeDocument) throws WsFaultException

Description	This method retrieves the GetStatusDocument element from the BODY of the SOAP envelope. The method has to throw exceptions, if either the element is null or is not present
Input arguments Process	Envelope Document
Output	GetStatusDocument
Exceptions	WsFaultException
Comments	

public GetStatusResponseDocument getGetStatusResponseDocument(EnvelopeDocument envelopeDocument) throws WsFaultException

Description	This method retrieves the GetStatusResponseDocument element from the BODY of the SOAP envelope. The method has to throw exceptions, if either the element is null or is not present
Input arguments Process	Envelope Document
Output	GetStatusResponseDocument
Exceptions	WsFaultException
Comments	

public UnsubscribeDocument getUnsubscribeDocument(EnvelopeDocument envelopeDocument) throws WsFaultException

Description	This method retrieves the UnsubscribeDocument element from the BODY of the SOAP envelope. The method has to throw exceptions, if
--------------------	--

	either the element is null or is not present.
Input arguments Process	Envelope Document
Output	UnsubscribeDocument
Exceptions	WsFaultException
Comments	

public SubscriptionEndDocument getSubscriptionEndDocument(EnvelopeDocument envelopeDocument) throws WsFaultException

Description	This method retrieves the SubscriptionEndDocument element from the BODY of the SOAP envelope. The method has to throw exceptions, if either the element is null or is not present
Input arguments Process	Envelope Document
Output	SubscriptionEndDocument
Exceptions	WsFaultException
Comments	

public IdentifierDocument getIdentifierDocument(EnvelopeDocument envelopeDocument) throws WsFaultException

Description	This method retrieves the identifier element from the HEADER of the SOAP envelope. The method has to throw exceptions, if either the element is null or is not present
Input arguments Process	Envelope Document
Output	IdentifierDocument
Exceptions	WsFaultException
Comments	

public IdentifierDocument getIdentifierDocumentFromBody(EnvelopeDocument envelopeDocument) throws WsFaultException

Description	This method retrieves the identifier element from the BODY of the SOAP envelope. The method has to throw exceptions, if either the element is null or is not present.
Input arguments Process	Envelope Document
Output	IdentifierDocument
Exceptions	WsFaultException
Comments	

public EndpointReferenceType getNotifyTo(SubscribeDocument subscribeDocument) throws WsFaultException

Description	This method tries to retrieve the notifyTo element. The method also issues exception in case it is unable to do so or if the retrived element is actually null
Input arguments Process	SubscribeDocument
Output	EndpointReferenceType
Exceptions	WsFaultException
Comments	

public class WseSinkProcessor extends WsProcessor implements SubscriptionExpiryListener

This class which implements the interface that encapsulates the behavior of a WS-Eventing sink node.

public boolean processExchange(EnvelopeDocument envelopeDocument, int direction)

Description	Process the exchange. The argument also indicates the direction in which the exchange has actually traversed
Input arguments	Envelope Document
Process	It gets the Addressing headers from Envelope Document. It gets the Action Document from addressing headers. It stores the request info based on action type and process response.
Output	Boolean true or false
Exceptions	UnknownExchangeException, IncorrectExchangeException, MessageFlowException, ProcessingException
Comments	

public EnvelopeDocument createSubscribeRequest(EndpointReferenceType sourceEpr, String deliveryMode, EndpointReferenceType endTo, String filterDialect, String filterConstraint, Calendar expiresAt)

Description	Creates a subscribe request based on the specified parameters
Input arguments	EndpointReferenceType sourceEpr, String deliveryMode, EndpointReferenceType endTo, String filterDialect, String filterConstraint, Calendar expiresAt
Process	It creates Subscribe Document first and then it creates Envelope based on subscribe document.
Output	Envelope document
Exceptions	ProcessingException
Comments	

public EnvelopeDocument createGetStatus(String subscriptionIdentifier) throws ProcessingException

Description	Creates getStatus Envelope Document based on subscription Identifier.
Input arguments	String subscriptionIdentifier
Process	First it creates GetStatusDocument and then creates Envelope Document.
Output	EnvelopeDocument
Exceptions	ProcessingException
Comments	

public EnvelopeDocument createRenew(String subscriptionIdentifier, Calendar renew) throws ProcessingException

Description	Creates Renew EnvelopeDocument based on subscription Identifier and renew date as a Calender format
Input arguments	String subscriptionIdentifier, Calendar renew
Process	It creates RenewDocument and then creates Envelope Document

	based on it.
Output	EnvelopeDocument
Exceptions	ProcessingException
Comments	

```
public EnvelopeDocument createUnsubscribe(String subscriptionIdentifier)
throws ProcessingException
```

Description	Creates Unsubscribe EnvelopeDocument based on subscription Identifier.
Input arguments	String subscriptionIdentifier
Process	It creates SubscriptionEntry based on Identifier and EnvelopeDocument based on subscriptionEpr. It stores the request info.
Output	EnvelopeDocument
Exceptions	ProcessingException
Comments	

```
private SubscriptionEntry retrieveSubscriptionEntry(String
subscriptionIdentifier) throws ProcessingException
```

Description	This method retrieves the subscription entry associated with the subscription identifier
Input arguments	String subscriptionIdentifier
Process	It checks for null values of identifier and existence in Subscription Management. If it exists in the Subscription Management, it will return subscription entry.
Output	SubscriptionEntry
Exceptions	ProcessingException
Comments	

```
private void storeRequestInfo(EnvelopeDocument envelopeDocument, Hashtable
requestsTable) throws ProcessingException
```

Description	Add information regarding a given request into the appropriate table.
Input arguments	EnvelopeDocument envelopeDocument, Hashtable requestsTable
Process	It checks for null values and gets the required info message Id form addressing headers. It stores messageId and EnvelopeDocument in Hashtable.
Output	void
Exceptions	ProcessingException
Comments	

```
private EnvelopeDocument retrieveRequestEnvelope(Hashtable requestsTable,
String messageIdOfRequest, String requestType) throws WsFaultException
```

Description	Retrieve the envelope associated with the requestId from the requestsTable that is associated with requests.
Input arguments	Hashtable requestsTable, String messageIdOfRequest, String requestType
Process	

	It checks for null values for messageId and hashtable and returns Envelope Document for messageId.
Output	EnvelopeDocument
Exceptions	WsFaultException
Comments	

public void processNotification(EnvelopeDocument envelopeDocument, AddressingHeaders addressingHeaders) throws ProcessingException

Description	This method is used to ensure process notifications
Input arguments	EnvelopeDocument, AddressingHeaders
Process	
Output	void
Exceptions	ProcessingException
Comments	

public void processSubscribeResponse(EnvelopeDocument envelopeDocument, AddressingHeaders addressingHeaders) throws WsFaultException, ProcessingException

Description	This method is used to process SubscribeResponses
Input arguments	EnvelopeDocument, AddressingHeaders
Process	This method gets the subscribe document and subscribe response document from Envelope Documents. It checks for expiry of the subscription. It creates entry into subscription management.
Output	Void
Exceptions	WsFaultException, ProcessingException
Comments	

public void processRenewResponse(EnvelopeDocument envelopeDocument, AddressingHeaders addressingHeaders) throws WsFaultException, ProcessingException

Description	This method is used to process RenewResponses
Input arguments	EnvelopeDocument, AddressingHeaders
Process	This gets the renew response document form Envelope. Checks subscription Identifier for problems. It updates subscription entry with renew details.
Output	void
Exceptions	WsFaultException, ProcessingException
Comments	

public void processGetStatusResponse(EnvelopeDocument envelopeDocument, AddressingHeaders addressingHeaders) throws WsFaultException, ProcessingException

Description	This method is used to process GetStatusRponses
Input arguments	EnvelopeDocument, AddressingHeaders
Process	This method gets the message id and subscription id from the response envelope document. It gets the expiry time and prints to the console.
Output	void

Exceptions	WsFaultException, ProcessingException
Comments	

```
public void processUnsubscribeResponse(EnvelopeDocument envelopeDocument,
AddressingHeaders addressingHeaders) throws WsFaultException
```

Description	This method is used to process UnsubscribeResponses
Input arguments	EnvelopeDocument, AddressingHeaders
Process	This method gets the messageId and subscriptionId from the Envelope Document. It deletes entry from subscription management.
Output	Void
Exceptions	WsFaultException, ProcessingException
Comments	

```
public void processSubscriptionEnd(EnvelopeDocument envelopeDocument,
AddressingHeaders addressingHeaders ) throws WsFaultException,
ProcessingException
```

Description	This method is used to process subscription end
Input arguments	EnvelopeDocument, AddressingHeaders
Process	This method gets the identifier form subscription end document and deletes entry from subscription management.
Output	void
Exceptions	WsFaultException, ProcessingException
Comments	

public class WseSourceProcessor extends WsProcessor

This class which implements the interface that encapsulates the behavior of a WS-Eventing source node.

```
public boolean processExchange(EnvelopeDocument envelopeDocument, int
direction) throws UnknownExchangeException, IncorrectExchangeException,
MessageFlowException, ProcessingException
```

Description	Process the exchange. The argument also indicates the direction in which the exchange has actually traversed
Input arguments	EnvelopeDocument envelopeDocument, int direction
Process	This method gets the Action Document from addressing headers. It checks the valid exchanges subscribe, unsubscribe and renew and process them.
Output	Boolean true or false
Exceptions	UnknownExchangeException, IncorrectExchangeException, MessageFlowException, ProcessingException
Comments	

```
private void processSubscriptionRequest(EnvelopeDocument envelopeDocument,
AddressingHeaders addressingHeaders) throws ProcessingException,
WsFaultException, MessageFlowException
```

Description	Processes subscription request from sink
Input arguments	EnvelopeDocument, AddressingHeaders
Process	This method gets the subscribe document and relatesToDocument

	from Envelope Document. It creates two responses one to Sink and one to Subscription Manager.
Output	void
Exceptions	ProcessingException, WsFaultException, MessageFlowException
Comments	

public SubscribeResponseDocument processSubscribeRequest(SubscribeDocument subscribeDocument) throws WsFaultException, ProcessingException

Description	The method to process a SubscriptionRequest issued by a Sink. If this request is successfully processed a SubscribeResponseDocument is issued. If the request is NOT successful a WsFaultException is thrown
Input arguments	SubscribeDocument subscribeDocument
Process	This method gets the subscription entry from SubscribeDocument. Generates response based on subscription entry.
Output	SubscribeResponseDocument
Exceptions	WsFaultException, ProcessingException
Comments	

public void processSubscriptionRenewal(EnvelopeDocument envelopeDocument, AddressingHeaders addressingHeaders) throws WsFaultException

Description	This method is used to update the subscription tables maintained at the source. ONLY requests successfully renewed by the SubscriptionManager are forwarded onto the source for updates at the source. This method will throw an error if the request was issued by anyone BUT the registered SubscriptionManager. wse:Identifier of the subscription. wse:From should match Subscription Manager for the subscription. wse:RenewResponse which would indicate the new expiry time for the subscription.
Input arguments	EnvelopeDocument, AddressingHeaders
Process	This method gets the Identifier Document from Envelope. Checks subscriptionId for errors and updates it in subscription management.
Output	void
Exceptions	WsFaultException
Comments	

public void processUnsubscribe(EnvelopeDocument envelopeDocument, AddressingHeaders addressingHeaders) throws WsFaultException

Description	This method is used to update the subscription tables maintained at the source. Unsubscribe requests processed by the SubscriptionManager are forwarded onto the source for updates at the source. This method will throw an error if the request was issued by anyone BUT the registered SubscriptionManager for the subscription in question.
--------------------	---

	wse:Identifier of the subscription wse:From should match SubscriptionManager for the subscription.
Input arguments	EnvelopeDocument, AddressingHeaders
Process	This method gets the message identifier and deletes entry from subscription management.
Output	void
Exceptions	WsFaultException
Comments	

public void processNotification(EnvelopeDocument envelopeDocument, AddressingHeaders addressingHeaders) throws ProcessingException, MessageFlowException

Description	This method is used to ensure the dissemination of a notification to registered subscriptions
Input arguments	EnvelopeDocument, AddressingHeaders
Process	Check for 'TO' element and it should not present. Checks for no of destinations for zero values. Copy the same envelope to multiple destinations and calls enroutNetwork() to send SOAPMessage notification.
Output	void
Exceptions	ProcessingException, MessageFlowException
Comments	

public void onSubscriptionExpiry(SubscriptionEntry expiredEntry)

Description	Method that is called when one of the managed subscriptions expires
Input arguments	SubscriptionEntry expiredEntry
Process	Checks for null values and terminate subscription.
Output	void
Exceptions	
Comments	

public String cancelSubscription(String subscriptionId, String additionalReason)

Description	A source can also facilitate cancellation of subscriptions that aren't yet slated to expire
Input arguments	String subscriptionId, String additionalReason
Process	Checks for entry in subscription management and terminate subscription entry form subscription management.
Output	String value of problems if any.
Exceptions	
Comments	

public void terminateServices()

Description	This is called by the source when it is doing a planned shutdown of the event source. All the subscriptions will be terminated and an appropriate SubscriptionEnd message will be sent to all the registered entities.
--------------------	--

Input arguments	
Process	This method will terminate all subscriptions in the subscription management.
Output	void
Exceptions	
Comments	

```
private String terminateSubscription(String subscriptionId, String status, String reason)
```

Description	Here we terminate the subscription. We first check to see if the subscriptionId is a valid one. Next, we create the appropriate subscriptionEnd. If there is an endTo that had been previously specified during the subscribe request, we send this endTo notification to that entity. If not, this message is sent to the sink
Input arguments	String subscriptionId, String status, String reason
Process	This method checks for subscription entry in subscription management. This will terminate subscription and creates subscription end Envelope. It will send one envelope to endTo or Sink and one to Subscription Manager.
Output	String value of error report
Exceptions	
Comments	

```
private void checkSubscriptionIdentifierForProblems(String subscriptionIdentifier) throws WsFaultException
```

Description	Check the specified subscription identifier for problems. Specifically throw an wse:InvalidFault if there are problems.
Input arguments	String subscriptionIdentifier
Process	Check subscription identifier for null values and entry in subscription management.
Output	
Exceptions	
Comments	

```
public class WseSubscriptionManagerProcessor extends WsProcessor implements WseSubscriptionManager, SubscriptionExpiryListene
```

This class encapsulates the behavior of a WS-Eventing subscription manager node.

```
public boolean processExchange(EnvelopeDocument envelopeDocument, int direction) throws UnknownExchangeException, IncorrectExchangeException, ProcessingException, MessageFlowException
```

Description	Process the exchange. The argument also indicates the direction in which the exchange has actually traversed
Input arguments	EnvelopeDocument, int direction
Process	This method gets action document from addressing headers. Check

	<p>action element for null values and calls appropriate process based on action element.</p> <p>Valid exchanges are:</p> <p>Subscription addition Subscription end Subscription renewal Subscription status Unsubscribe</p>
Output	Boolean true or false
Exceptions	UnknownExchangeException, IncorrectExchangeException, ProcessingException, MessageFlowException
Comments	

```
public void processSubscriptionRenewal(EnvelopeDocument envelopeDocument,
AddressingHeaders addressingHeaders) throws ProcessingException,
WsFaultException, MessageFlowException
```

Description	Process a subscription renewal. This method will throw exceptions (or generate faults) if the identifier contained in the subscription renewal is an unknown one.
Input arguments	EnvelopeDocument , AddressingHeaders
Process	This method get the identifier, subscriptionEntry, expiry date and renewal date and creates renew response document. It will enroute response document to Sink and subscription manager.
Output	void
Exceptions	ProcessingException, WsFaultException, MessageFlowException
Comments	

```
public void processSubscriptionStatus(EnvelopeDocument envelopeDocument,
AddressingHeaders addressingHeaders) throws ProcessingException,
WsFaultException, MessageFlowException
```

Description	Process a request to get the status of a subscription. This method will throw exceptions (or generate faults) if the identifier contained in the subscription GetStatus is an unknown one
Input arguments	EnvelopeDocument, AddressingHeaders
Process	This method gets the status document, subscription Id from EnvelopeDocument. It checks for presence of [wse:GetStatus] element and creates response Envelope. It will enroute creates EnvelopeDocument to the Network (Sink)
Output	void
Exceptions	ProcessingException, WsFaultException, MessageFlowException
Comments	

```
public void processSubscriptionUnsubscribe(EnvelopeDocument envelopeDocument,
AddressingHeaders addressingHeaders) throws ProcessingException,
WsFaultException, MessageFlowException
```

Description	Process a request to unsubscribe subscription. This method will throw exceptions (or faults) if the identifier contained in the UnSubscribe is an unknown one.
Input arguments	EnvelopeDocument, AddressingHeaders

Process	This method will get the UnsubscribeDocument, subscription identifier from Envelope Document. It will get the [wse:Unsubscribe] request type and creates unsubscribe response document. This method will send response document to Sink and Source.
Output	
Exceptions	ProcessingException, WsFaultException, MessageFlowException
Comments	

public void processSubscriptionAddition(EnvelopeDocument envelopeDocument, AddressingHeaders addressingHeaders) throws WsFaultException, MessageFlowException

Description	This method is used to notify a subscription manager of the a successful subscription request-response combination. The request (issued by the sink) allows a subscription manager to know about the subscription, while the response (issued by the source) allows it know the wse:Identifier for this subscription and also the expiry associated with the subscription. Also note that the wse:Identifier is what is used by the sinks in all their interactions with the subscription manager
Input arguments	EnvelopeDocument, AddressingHeaders
Process	This method will get subscribeResponseDocument, subscription Id from Envelope document. Check for subscriptionId for null values. It checks for expiry of subscription and then adds entry into subscription management.
Output	void
Exceptions	WsFaultException, MessageFlowException
Comments	

private void processSubscriptionEnd(EnvelopeDocument envelopeDocument, AddressingHeaders addressingHeaders) throws WsFaultException, MessageFlowException

Description	This method is used to notify a subscription manager of a subscription end. The wse:Identifier for this subscription can be retrieve from this message
Input arguments	EnvelopeDocument, AddressingHeaders
Process	This method will identifier from subscription document and deletes entry from subscription management.
Output	Void
Exceptions	WsFaultException, MessageFlowException
Comments	

private void checkSubscriptionIdentifierForProblems(String subscriptionIdentifier) throws WsFaultException

Description	Check the specified subscription identifier for problems. Specifically throw an wse:InvalidFault if there are problems
Input arguments	String subscriptionIdentifier
Process	This method checks Identifier for null values and entry in the subscription management.

Output	void
Exceptions	WsFaultException
Comments	

private void checkExchangeType(String action, int direction) throws UnknownExchangeException, IncorrectExchangeException

Description	Checks for direction where message is originated, i.e., network or application.
Input arguments	String action, int direction
Process	This method checks for message direction weather it is coming from network or from application. It will call the appropriate method.
Output	void
Exceptions	UnknownExchangeException, IncorrectExchangeException
Comments	

private void checkExchangeFromNetwork(String action) throws UnknownExchangeException, IncorrectExchangeException

Description	This will method will check the exchanges from Network and valid eventing actions from Network.
Input arguments	String action
Process	This method checks for null values. This method checks for valid exchanges getSubscribe and getRenewResponse from Network throws throwIncorrectExchangeException.
Output	void
Exceptions	UnknownExchangeException, IncorrectExchangeException
Comments	

private void checkExchangeFromApplication(String action) throws UnknownExchangeException, IncorrectExchangeException

Description	This will method will check for valid exchanges coming from Application
Input arguments	String action
Process	This will check for exchanges for null values. It will check for valid exchanges from application and throws invalid incorrectExchangeException.
Output	Void
Exceptions	UnknownExchangeException, IncorrectExchangeException
Comments	Subscription manager will not receive any exchanges from Application.

private EndpointReferenceType initializeToFromRequest(AddressingHeaders addressingHeaders, EndpointReferenceType alternateEPR, String requestType) throws ProcessingException

Description	Try and the initialize the EPR to send a response message to. This is typically used to initialize the [wsa:To] element. The addressing headers that are passed here are the headers that were constructed based on the original request that came in previously. So this method will look for the [wsa:ReplyTo] or [wsa:From] elements to construct the responses
--------------------	--

Input arguments	AddressingHeaders addressingHeaders, EndpointReferenceType alternateEPR, requestType String
Process	This method will check for “to” for null values. If ‘to’ value is null it will check for alternate end point reference. If both are null, it will throw processing exception.
Output	EndpointReferenceType
Exceptions	ProcessingException
Comments	

Package: cgl.narada.wsinfra.util

public class ConfigurationParamsReader implements Configuration

public void load(String file) throws IOException

Description	Loads the properties from the given file
Input arguments	String file: The properties file to load
Process	Retrieves InputStream by passing “String file” argument to method “InputStream getPropertyStream(String)” and calls the method “void load(InputStream , String)”
Output	-
Exceptions	IOException
Comments	-

public void load(InputStream input) throws IOException

Description	Loads the properties from the given input stream.
Input arguments	InputStream input: An InputStream on the properties file
Process	Calls the method void load(InputStream , String) by passing argument “InputStream input”
Output	-
Exceptions	IOException
Comments	-

public void load(InputStream input, String enc) throws IOException

Description	Loads the properties from the given input stream and using the specified encoding.
Input arguments	InputStream input: An InputStream String enc: An encoding
Process	Loads each and every property from stream specified by InputStream input and adds each property to configuration by calling method “void addProperty(String key, Object token)” method
Output	-
Exceptions	IOException
Comments	-

public Input Stream getPropertyStream(String resourceName) throws IOException

Description	Gets the InputStream from the file specified by String resourceName
Input arguments	String resourceName - The file name from which to get InputStream

Process	Checks whether file specified by String resourceName exists. If not exists then throws FileNotFoundException. If file exists and can be read properly then creates InputStream associated with the file and returns it.
Output	Returns an InputStream associated with file specified by String resourceName
Exceptions	If file specified by String resourceName could not be read then throws IOException
Comments	-

public void addProperty(String key, Object token)

Description	Adds the property to configuration
Input arguments	String key - The Key to add the property to. Object token - The Value to add.
Process	Adds a property to the configuration. If it already exists then the value stated here will be added to the configuration entry. For example, if resource.loader = file is already present in the configuration and you addProperty("resource.loader", "classpath") Then you will end up with a Vector like the following: ["file", "classpath"]
Output	-
Exceptions	-
Comments	-

protected List processString(String token)

Description	Returns the Strings associated with String token
Input arguments	String token - A String token
Process	Returns the Strings associated with String token as a List.
Output	Returns List of Strings
Exceptions	-
Comments	-

protected void addPropertyDirect(String key, Object obj)

Description	Adds Key and Object to Hashtable
Input arguments	String key - key to use for mapping Object obj - object to store
Process	Adds Key and Object to Hashtable
Output	-
Exceptions	-
Comments	-

public String testBoolean(String value)

Description	Maps true, on, yes to true; false, off, no to false.
Input arguments	String value - The value to test for boolean state.
Process	Returns "true" if String value = "true", "on" or "yes" Returns "false" if String value = "false", "off" or "no" Else returns null
Output	Returns String value
Exceptions	-
Comments	-

public Configuration subset(String prefix)

Description	Creates an Configuration object that is a subset of “this” one
Input arguments	<code>String prefix</code> - The prefix used to select the properties.
Process	Creates a Configuration object that is a subset of this one. The new Configuration object contains every key from the current Configuration that starts with prefix. The prefix is removed from the keys in the subset.
Output	Returns Configuration Object
Exceptions	-
Comments	-

public boolean isEmpty()

Description	Checks if the configuration is empty.
Input arguments	-
Process	Returns true if the configuration contains no key/ value pair, false otherwise
Output	Returns boolean
Exceptions	-
Comments	-

public void setProperty(String key, Object value)

Description	Sets a property, this will replace any previously set values.
Input arguments	<code>String key</code> - The key of the property to change <code>Object value</code> - The new value
Process	implicitly calls <code>clearProperty(key)</code> , <code>addProperty(key,value)</code>
Output	-
Exceptions	-
Comments	-

public void clearProperty(String key)

Description	Clears a property in the configuration.
Input arguments	<code>String key</code> - the key to remove along with corresponding value.
Process	implicitly calls <code>clearProperty(key)</code> , <code>addProperty(key, value)</code>
Output	Removes key, value pair(property) from Configuration Hashtable object
Exceptions	-
Comments	-

public Iterator getKeys()

Description	Get the list of the keys contained in the configuration repository.
Input arguments	-
Process	Returns an Iterator associated with keys.
Output	An Iterator object containing list of keys
Exceptions	-
Comments	-

public Properties getProperties(Properties _defaults)

Description	Returns all the properties that have been read from the specified configuration file
Input arguments	<code>Properties _defaults</code> - Default properties

Process	Returns all the properties from the configuration. If don't not exist then returns default properties specified by Properties _defaults
Output	Properties Object contains list of properties associated with configuration
Exceptions	-
Comments	-

public Properties getProperties(String key)

Description	Gets a list of properties associated with the given configuration key
Input arguments	String key - The configuration key
Process	Calls the method "Properties getProperties(String key, Properties defaults) " method
Output	Properties Object contains list of properties associated with configuration key specified by String key.
Exceptions	-
Comments	-

public Properties getProperties(String key, Properties defaults)

Description	Gets a list of properties associated with the given configuration key
Input arguments	String key - The configuration key Properties defaults - Default properties
Process	Returns all the properties associated with String key from the configuration. If don't exist then return default properties specified by parameter Properties defaults
Output	Properties Object contains list of properties associated with configuration key specified by String key.
Exceptions	-
Comments	-

public String getProperty(String key)

Description	Gets a property from the configuration associated with key
Input arguments	String key - property to retrieve
Process	Returns value as object associated with key. Will return user value if exists, if not then default value if exists, otherwise null
Output	Returns string representation of value.
Exceptions	-
Comments	-

public boolean getBoolean(String key)

Description	Gets a boolean associated with the given configuration key.
Input arguments	String key - The configuration key.
Process	Calls method Boolean getBoolean(String key, Boolean defaultValue) by passing defaultValue as null.
Output	Returns the associated boolean value.
Exceptions	-
Comments	-

public boolean getBoolean(String key, boolean defaultValue)

Description	Gets a boolean associated with the given configuration key.
Input arguments	String key - The configuration key boolean defaultValue - default value

Process	Calls method Boolean <code>getBoolean(String key, Boolean default)</code>
Output	Returns the associated boolean value.
Exceptions	-
Comments	-

public Boolean getBoolean(String key, Boolean defaultValue)

Description	Gets a Boolean associated with the given configuration key.
Input arguments	<code>String key</code> - The configuration key <code>Boolean defaultValue</code> - default value
Process	Returns the associated boolean if key is found and has valid format, default value otherwise.
Output	Returns the boolean value.
Exceptions	-
Comments	-

public byte getByte(String key)

Description	Gets a byte associated with the given configuration key.
Input arguments	<code>String key</code> - The configuration key
Process	Calls method Byte <code>getByte(String key, Byte defaultValue)</code> by passing <code>defaultValue</code> as null.
Output	Returns the associated byte value.
Exceptions	-
Comments	-

public byte getByte(String key, byte defaultValue)

Description	Gets a byte associated with the given configuration key.
Input arguments	<code>String key</code> - The configuration key <code>byte defaultValue</code> - default value
Process	Calls method Byte <code>getByte(String key, Byte default)</code>
Output	Returns the associated byte value.
Exceptions	-
Comments	-

public Byte getByte(String key, Byte defaultValue)

Description	Gets a Byte associated with the given configuration key.
Input arguments	<code>String key</code> - The configuration key <code>Byte defaultValue</code> - default value
Process	Returns the associated Byte if key is found and has valid format, default value otherwise.
Output	Returns the Byte value.
Exceptions	-
Comments	-

public double getDouble(String key)

Description	Gets a double associated with the given configuration key.
Input arguments	<code>String key</code> - The configuration key
Process	Calls method Double <code>getDouble (String key, Double defaultValue)</code> by passing <code>defaultValue</code> as null.
Output	Returns the associated double value.
Exceptions	-

Comments	-
-----------------	---

public double getDouble (String key, double defaultValue)

Description	Gets a double associated with the given configuration key.
Input arguments	String key - The configuration key double defaultValue - default value
Process	Calls method Double getDouble (String key, Double default)
Output	Returns the associated double value.
Exceptions	-
Comments	-

public Double getDouble(String key, Double defaultValue)

Description	Gets a Double associated with the given configuration key.
Input arguments	String key - The configuration key Double defaultValue - default value
Process	Returns the associated Double if key is found and has valid format, default value otherwise.
Output	Returns the Double value.
Exceptions	-
Comments	-

public float getFloat(String key)

Description	Gets a float associated with the given configuration key.
Input arguments	String key - The configuration key
Process	Calls method Float getFloat (String key, Float defaultValue) by passing defaultValue as null.
Output	Returns the associated float value.
Exceptions	-
Comments	-

public float getFloat(String key, float defaultValue)

Description	Gets a float associated with the given configuration key.
Input arguments	String key - The configuration key float defaultValue - default value
Process	Calls method Float getFloat (String key, Float default)
Output	Returns the associated float value.
Exceptions	-
Comments	-

public Float getFloat(String key, Float defaultValue)

Description	Gets a Float associated with the given configuration key.
Input arguments	String key - The configuration key Float defaultValue - default value
Process	Returns the associated Float if key is found and has valid format, default value otherwise.
Output	Returns the Float value.
Exceptions	-
Comments	-

public int getInteger(String key)

Description	Gets an integer associated with the given configuration key.
Input arguments	String key - The configuration key
Process	Calls method Integer getInteger (String key, Integer defaultValue) by passing defaultValue as null.
Output	Returns the associated integer value.
Exceptions	-
Comments	-

public int getInteger(String key, int defaultValue)

Description	Gets an integer associated with the given configuration key.
Input arguments	String key - The configuration key int defaultValue - default value
Process	Calls method Integer getInteger (String key, Integer default)
Output	Returns the associated integer value.
Exceptions	-
Comments	-

public Integer getInteger(String key, Integer defaultValue)

Description	Gets an Integer associated with the given configuration key.
Input arguments	String key - The configuration key Integer defaultValue - default value
Process	Returns the associated Integer if key is found and has valid format, default value otherwise.
Output	Returns the Integer value.
Exceptions	-
Comments	-

public long getLong(String key)

Description	Gets a long associated with the given configuration key.
Input arguments	String key - The configuration key
Process	Calls method Long getLong (String key, Long defaultValue) by passing defaultValue as null.
Output	Returns the associated long value.
Exceptions	-
Comments	-

public long getLong(String key, long defaultValue)

Description	Gets a long associated with the given configuration key.
Input arguments	String key - The configuration key long defaultValue - default value
Process	Calls method Long getLong (String key, Long default)
Output	Returns the associated long value.
Exceptions	-
Comments	-

public Long getLong(String key, Long defaultValue)

Description	Gets a Long associated with the given configuration key.
Input arguments	String key - The configuration key Long defaultValue - default value
Process	Returns the associated Long if key is found and has valid format, default value otherwise.

Output	Returns the Long value.
Exceptions	-
Comments	-

public short getShort(String key)

Description	Gets a short associated with the given configuration key.
Input arguments	String key - The configuration key
Process	Calls method Short getShort (String key, Short defaultValue) by passing defaultValue as null.
Output	Returns the associated short value.
Exceptions	-
Comments	-

public short getShort(String key, short defaultValue)

Description	Gets a short associated with the given configuration key.
Input arguments	String key - The configuration key short defaultValue - default value
Process	Calls method Short getShort (String key, Short default)
Output	Returns the associated short value.
Exceptions	-
Comments	-

public Short getShort(String key, Short defaultValue)

Description	Gets a Short associated with the given configuration key.
Input arguments	String key - The configuration key Short defaultValue - default value
Process	Returns the associated Short if key is found and has valid format, default value otherwise.
Output	Returns the Short value.
Exceptions	-
Comments	-

public String getString(String key)

Description	Gets a String associated with the given configuration key.
Input arguments	String key - The configuration key
Process	Calls method String getString (String key, String defaultValue) by passing defaultValue as null.
Output	Returns the associated String value.
Exceptions	-
Comments	-

public String getString(String key, String defaultValue)

Description	Gets a String associated with the given configuration key.
Input arguments	String key - The configuration key String defaultValue - default value
Process	Returns the associated String if key is found and has valid format, default value otherwise.
Output	Returns the associated String value.
Exceptions	-
Comments	-

public String[] getStringArray(String key)

Description	Get an array of strings associated with the given configuration key.
Input arguments	String key - The configuration key
Process	Returns the associated string array if key is found.
Output	Returns the String[]
Exceptions	-
Comments	-

public Vector getVector(String key)

Description	Get a Vector of strings associated with the given configuration key
Input arguments	String key - The configuration key
Process	Returns the associated Vector if key is found.
Output	Returns the Vector
Exceptions	-
Comments	-

public Vector getVector(String key, Vector defaultValue)

Description	Get a Vector of strings associated with the given configuration key
Input arguments	String key - The configuration key Vector defaultValue - The default value.
Process	Returns the associated Vector if key is found otherwise returns default one
Output	Returns the Vector
Exceptions	-
Comments	-

public Class ExceptionToFaultConversion

public static ExceptionToFaultConversion getInstance()

Description	Returns an instance of Class ExceptionToFaultConversion
Input arguments	-
Process	Returns an instance of Class ExceptionToFaultConversion
Output	Instance of ExceptionToFaultConversion class
Exceptions	-
Comments	-

public EnvelopeDocument convertToSOAPFault(WsFaultException wsFaultException)

Description	Returns an EnvelopeDocument associated with fault
Input arguments	WsFaultException wsFaultException
Process	This method returns a NULL if either the faultTo or the faultCodeQName in the wsFaultException is null. Otherwise creates EnvelopeDocument from the argument wsFalutException and returns it.
Output	Null or EnvelopeDocument
Exceptions	-
Comments	-

public class FaultCreator extends Object

This class enables the creation of faults. Note that currently this class creates on SOAP 1.1 faults, it should be no big deal to have a similar capability for SOAP 1.2. If a need arises one will be provided.

public static FaultCreator getInstance()

Description	Returns an instance of Class FaultCreator
Input arguments	-
Process	Returns an instance of Class FaultCreator
Output	Instance of FaultCreator
Exceptions	-
Comments	-

public EnvelopeDocument createFault(QName faultCodeQName, String reason, EndpointReferenceType faultTo)

Description	Returns an instance of Class FaultCreator
Input arguments	-
Process	Returns an instance of Class FaultCreator
Output	Instance of FaultCreator
Exceptions	-
Comments	-

public class XmlDateTimeConversion extends Object

This is a utility class which converts between Calendar elements and the XmlDateTime element.

public static XmlDateTimeConversion getInstance()

Description	Returns an instance of Class XmlDateTimeConversion
Input arguments	-
Process	Returns an instance of Class XmlDateTimeConversion
Output	Instance of XmlDateTimeConversion
Exceptions	-
Comments	-

public XmlDateTime getXmlDateTime(Calendar calendar)

Description	Get the XMLDateTime associated with the specified calendar value
Input arguments	Calendar calendar
Process	Creates XmlDataTime object and sets its calendar value with parameter calender
Output	Instance of XmlDateTime
Exceptions	-
Comments	-

public ExpiresDocument getExpires(Calendar expiresAt)

Description	Gets the ExpiresDocument associated with the specified calendar value
Input arguments	Calendar expiresAt
Process	Creates ExpiresDocument object and sets its AttributedDateTime value with parameter calender
Output	Instance of ExpiresDocument
Exceptions	-
Comments	-

public Calendar getExpires(ExpiresDocument expires)throws XmlException

Description	Gets the Calendar value from specified ExpiresDocument
Input arguments	ExpiresDocument expires
Process	Retrieves Calendar value from ExpiresDocument
Output	Returns Calendar
Exceptions	XmlException will be thrown if there is an error during parsing ExpiresDocument.
Comments	-

public class XmlContentTransfer extends Object

This is a utility class which performs several functions related to the transfer of XmlContent between elements.

public static XmlContentTransfer getInstance()

Description	Returns an instance of Class XmlContentTransfer
Input arguments	-
Process	Returns an instance of Class XmlContentTransfer
Output	Instance of XmlContentTransfer
Exceptions	-
Comments	-

public void copyFromSourceToDestination(XmlObject source, XmlObject destination)

Description	Copy the Xml Contents from the source to the destination.
Input arguments	XmlObject source XmlObject destination
Process	Retrieves XmlCursor for input parameters source and destination. It also positions the destination cursor at the right location prior to the actual copy. and finally calls the method copyFromSourceToDestination(XmlCursor source, XmlCursor destination)
Output	-
Exceptions	-
Comments	-

public void copyFromSourceToDestination(XmlObject source, XmlCursor destinationCursor)

Description	Copy the Xml Contents from the source to the destination.
Input arguments	XmlObject source XmlCursor destinationCursor
Process	Retrieves XmlCursor for input parameter source. It also positions the destination cursor at the right location prior to the actual copy. And finally calls the method copyFromSourceToDestination(XmlCursor source, XmlCursor destination)
Output	-
Exceptions	-
Comments	-

public void copyFromSourceToDestination(XmlCursor sourceCursor, XmlCursor destinationCursor)

Description	Copy of the XmlContents from the source to the destination cursor.
Input arguments	XmlCursor sourceCursor XmlCursor destinationCursor
Process	Copy of the XmlContents from the source to the destination cursor.

Output	-
Exceptions	-
Comments	-

public void copyFromSourceToDestinationAsLastChild(XmlObject sourceObject, XmlObject destinationObject)

Description	Copies the contents of the destination object as the last child within the source object.
Input arguments	XmlObject sourceObject, XmlObject destinationObject
Process	Retrieves XmlCursor for input parameters sourceObject and destinationObject. Proceed to position the destination cursor at the location just after its last child element if there is one prior to the actual copy. And finally calls the method copyFromSourceToDestination(XmlCursor source, XmlCursor destination)
Output	-
Exceptions	-
Comments	-

public void copyFromSourceToDestinationAsFirstChild(XmlObject sourceObject, XmlObject destinationObject)

Description	Copies the contents of the destination object as the first child within the source object.
Input arguments	XmlObject sourceObject, XmlObject destinationObject
Process	Retrieves XmlCursor for input parameters sourceObject and destinationObject. Proceeds to position the destination cursor at the location just before its first child element if there is one prior to the actual copy. And finally calls the method copyFromSourceToDestination(XmlCursor source, XmlCursor destination)
Output	-
Exceptions	-
Comments	-

public final class UUIDRetrieval extends Object

This class provides a small wrapper around the JUG UUIDGenerator written by Tatu Saloranta, tatu.saloranta@iki.fi. This class just adds a couple of things such as abstract the namespace UUID generation from the applications that need to generate uuids.

public static UUIDRetrieval getInstance()

Description	Returns an instance of Class UUIDRetrieval
Input arguments	-
Process	Returns an instance of Class UUIDRetrieval
Output	Instance of UUIDRetrieval
Exceptions	-
Comments	-

public org.doomdark.uuid.UUID generateTimeBasedUUID()

Description	Generates time based UUID
Input arguments	-
Process	Calls the method generateTimeBasedUUID() of class UUIDGenerator
Output	Returns an instance of org.doomdark.uuid.UUID
Exceptions	-
Comments	-

public org.doomdark.uuid.UUID generateRandomBasedUUID()

Description	Generates Random base UUID
Input arguments	-
Process	Calls the method generateRandomBasedUUID(SecureRandom secureRandom)of class UUIDGenerator
Output	Returns an instance of org.doomdark.uuid.UUID
Exceptions	-
Comments	-

public String getTimeBasedUUIDAsString()

Description	Generates time based UUID as String
Input arguments	-
Process	Calls the method generateTimeBasedUUID(). Which returns instance of org.doomdark.uuid.UUID and converts it to string.
Output	Returns string representation of org.doomdark.uuid.UUID
Exceptions	-
Comments	-

public String getRandomBasedUUIDAsString()

Description	Generates Random base UUID as String
Input arguments	-
Process	Calls the method generateRandomBasedUUID() which returns instance of org.doomdark.uuid.UUID and converts it to string.
Output	Returns string representation of org.doomdark.uuid.UUID
Exceptions	-
Comments	-

public class UniqueIntGenerator extends Object

Some operations require the generation of unique integer values. Within a given VM we need to ensure this uniqueness. This class achieves that.

public static UniqueIntGenerator getInstance()

Description	Returns an instance of Class UniqueIntGenerator
Input arguments	-
Process	Returns an instance of Class UniqueIntGenerator
Output	Instance of UniqueIntGenerator
Exceptions	-
Comments	-

public int getNext()

Description	Increments the tracker element and returns it
Input arguments	-
Process	Increments the tracker element and returns it
Output	Returns integer
Exceptions	-
Comments	-

public class SoapPrinter extends Object

A utility class that provides string based representations of SOAP messages.

public static String getStringRepresentation(SOAPMessage soapMessage) throws Exception

Description	Generates string representation of SOAP messages.
Input arguments	SOAPMessage soapMessage
Process	Creates ByteArrayOutputStream Object. Creates DataOutputStream object by passing ByteArrayOutputStream Object as a parameter. Writes soap message to DataOutputStream by using method soapMessage.writeTo(DataOutputStream) and returns string representation of ByteArrayOutputStream.
Output	Returns string.
Exceptions	Exception
Comments	-

public class SOAPMessageModifier extends Object

This is a utility class that performs several modifications to SOAPMessage.

Field Summary:

public static SOAPMessageModifier **instance**

Methods summary:

public static SOAPMessageModifier getInstance()

Description	Returns an instance of Class SOAPMessageModifier
Input arguments	-
Process	Returns an instance of Class SOAPMessageModifier
Output	Instance of SOAPMessageModifier
Exceptions	-
Comments	-

public void addSOAPElementToHeader(SOAPMessage soapMessage, SOAPElement soapElement) throws Exception

Description	Adds the soapElement to the header of the specified SOAPMessage
Input arguments	SOAPMessage soapMessage SOAPElement soapElement
Process	This +method first adds NamespaceDeclaration to the soapEnvelope retrieved from soapMessage. And finally adds soapElement and its child elements to the soapHeader retrieved from SOAPMessage.

Output	-
Exceptions	Exception
Comments	-

public class SoapMessageAlteration extends Object

This is a utility class which facilitates the addition of elements to Header or Body of the SOAP envelope.

public static SoapMessageAlteration getInstance()

Description	Returns an instance of Class SoapMessageAlteration
Input arguments	-
Process	Returns an instance of Class SoapMessageAlteration
Output	Instance of SoapMessageAlteration
Exceptions	-
Comments	-

public void addToSoapBody(EnvelopeDocument envelopeDocument, XmlObject xmlObject)

Description	Adds the specified XmlObject to the Body of the SOAP message
Input arguments	EnvelopeDocument envelopeDocument, XmlObject xmlObject
Process	This method retrieves the bodyType from envelopeDocument and finally calls method copyFromSourceToDestination(XmlObject xmlObject, XmlObject bodyType) of XmlContentTransfer class.
Output	-
Exceptions	-
Comments	-

public void addToSoapHeader(EnvelopeDocument envelopeDocument, XmlObject xmlObject)

Description	Adds the specified XmlObject to the Header of the SOAP message
Input arguments	EnvelopeDocument envelopeDocument, XmlObject xmlObject
Process	This method retrieves the headerType object from envelopeDocument and finally calls method copyFromSourceToDestination(XmlObject xmlObject, XmlObject headerType) of XmlContentTransfer class.
Output	-
Exceptions	-
Comments	-

public void addToSoapHeaderAsLastChild(EnvelopeDocument envelopeDocument, XmlObject xmlObject)

Description	Adds the specified XmlObject to the Header of the SOAP message as a LastChild
Input arguments	EnvelopeDocument envelopeDocument, XmlObject xmlObject
Process	This method retrieves the headerType object from envelopeDocument and finally calls method copyFromSourceToDestinationAsLastChild(XmlObject xmlObject, XmlObject headerType) of XmlContentTransfer class.
Output	-
Exceptions	-

Comments	-
-----------------	---

public boolean addToSoapHeader(EnvelopeDocument envelopeDocument, QName qName, String value)

Description	Adds a QName and the corresponding value to a SOAP Header. This method returns <i>true</i> , if the operation was successful and <i>false</i> otherwise
Input arguments	EnvelopeDocument envelopeDocument QName qName String value
Process	If any of the above mentioned parameters are null then this method returns false. It retrieves envelopeType from the envelopeDocument, retrieves headerType from the envelopeType and associate XmlCursor(headerCursor) with headerType. It also positions the cursor at appropriate place and calls the method headerCursor.insertElementWithText(qName, value).
Output	Returns true if the operation is successful else false
Exceptions	-
Comments	-

public boolean removeFromSoapHeader(EnvelopeDocument envelopeDocument, QName qName)

Description	Removes a QName and corresponding text value encapsulated by it from the specified document. This method returns <i>true</i> , if the operation was successful and <i>false</i> otherwise.
Input arguments	EnvelopeDocument envelopeDocument QName qName
Process	If any of the above mentioned parameters are null then this method returns false. It retrieves envelopeType from the envelopeDocument, retrieves headerType from the envelopeType and associate XmlCursor(headerCursor) with headerType. It also locates the qName by using method QNameLocator.locateQName(headerCursor, qName); If it returns true then removes the xml by using method headerCursor.removeXml()
Output	Returns true if the operation is successful else false
Exceptions	-
Comments	-

public class SoapMarshaller extends Object

This is a utility class to marshall and un-marshall SOAP messages.

public static byte[] marshallSoapMessage(SOAPMessage soapMessage) throws Exception

Description	This method converts a SOAPMessage into a stream of bytes
Input arguments	SOAPMessage soapMessage
Process	Creates ByteArrayOutputStream Object. Creates DataOutputStream object by passing ByteArrayOutputStream Object as a parameter. Writes soap message to DataOutputStream by using method soapMessage.writeTo(DataOutputStream) and returns array of bytes of ByteArrayOutputStream by using method ByteArrayOutputStream.toByteArray() method
Output	Returns array of bytes
Exceptions	Exception
Comments	-

public static SOAPMessage unmarshallSoapMessage(byte[] marshalledBytes) throws Exception

Description	This method unmarshals a SOAPMessage from a stream of bytes
Input arguments	byte[] marshalledBytes
Process	Creates ByteArrayInputStream Object from marshaledBytes. Creates DataInputStream object by passing ByteArrayOutputStream Object as a parameter and finally creates SOAPMessage by calling method MessageFactory.newInstance().createMessage(new MimeHeaders(), DataInputStream)
Output	Returns SOAPMessage
Exceptions	Exception
Comments	-

public static SOAPMessage unmarshallSoapMessage(InputStream in) throws Exception

Description	This method unmarshals a SOAPMessage from the specified InputStream
Input arguments	InputStream in
Process	Creates DataInputStream object by passing InputStream in Object as a parameter and finally creates SOAPMessage by calling method MessageFactory.newInstance().createMessage(new MimeHeaders(), DataInputStream)
Output	Returns SOAPMessage
Exceptions	Exception
Comments	-

public class SoapHeaderProcessor extends Object

This class performs utility functions pertaining to the processing of SOAP headers

public static boolean hasHeaderElement(SOAPEnvelope soapEnvelope, String localName, String prefix, String prefixUri) throws Exception

example:

```
<wombat:GetLastTradePrice xmlns:wombat="http://www.wombat.org/trader">
("xmlns" stands for "XML namespace".) prefix = wombat, localName = getLastTradePrice, prefixURI=
http://www.wombat.org/trader
```

Description	This method returns true if element with localName, prefix and prefixUri exist in soapEnvelope otherwise false.
Input arguments	SOAPEnvelope soapEnvelope, String localName, String prefix, String prefixUri
Process	Retrieves soapHeader from soapEnvelope. Creates Name element from localName, prefix and prefixUri. Finally it retrieves all the child elements for the Name element from soapHeader by using method soapHeader.getChildElements(Name elementName). If child element presents then return true, else return false.
Output	Returns true if element exist else return false.
Exceptions	Exception
Comments	-

public static SOAPElement getHeaderElement(SOAPEnvelope soapEnvelope, String localName, String prefix, String prefixUri) throws Exception

Description	Retrieves the SOAPElement corresponding to the localName, prefix and prefixUri that have been specified
Input arguments	SOAPEnvelope soapEnvelope, String localName, String prefix, String prefixUri
Process	Retrieves soapHeader from soapEnvelope. Creates Name element from localName, prefix and prefixUri. Finally it retrieves all the child elements for the Name element from soapHeader by using method soapHeader.getChildElements(Name elementName). If child element presents then return the element, else return null
Output	Returns SOAPElement if element exist else return null.
Exceptions	Exception
Comments	-

public class SoapEnvelopeConversion extends Object

This is a utility class which performs conversions between SOAPMessage and Envelope document. This class is typically used in scenarios where such conversions are necessary.

public static SoapEnvelopeConversion getInstance()

Description	Returns an instance of Class SoapEnvelopeConversion
Input arguments	-
Process	Returns an instance of Class SoapEnvelopeConversion
Output	Instance of SoapEnvelopeConversion
Exceptions	-
Comments	-

public SOAPMessage getSOAPMessage(EnvelopeDocument envelopeDocument) throws ProcessingException

Description	This method converts an EnvelopeDocument into a javax.xml.SOAPMessage.
Input arguments	EnvelopeDocument envelopeDocument
Process	If envelopeDocument which is passed as an argument is null then it throws ProcessingException. It retrieves byte array from envelopeDocument and converts it to SOAPMessage by calling method unmarshallSoapMessage(byte[]) of SoapMarshaller class.
Output	Returns SOAPMessage
Exceptions	ProcessingException generates when envelopeDocument passed as an argument is null or while there is any problems in converting EnvelopeDocument to SOAPMessage.
Comments	-

public EnvelopeDocument getEnvelopeDocument(SOAPMessage soapMessage) throws ProcessingException

Description	This method converts a SOAPMessage into an Envelope Document.
Input arguments	SOAPMessage soapMessage
Process	If soapMessage which is passed as an argument is null then it throws ProcessingException. It retrieves string representation of soapMessage by calling method getStringRepresentation(soapMessage) of SoapPrinter class. Finally it creates envelopeDocument by passing string representation of soapMessage to method EnvelopeDocument.Factory.parse(soapMessageString).
Output	Returns EnvelopeDocument
Exceptions	ProcessingException generates when soapMessage passed as an argument is null or while there is any problems in converting SOAPMessage to EnvelopeDocument.
Comments	-

public class FaultCreator extends Object

This class enables the creation of faults. Note that currently this class creates on SOAP 1.1 faults, it should be no big deal to have a similar capability for SOAP 1.2. If a need arises one will be provided.

public static FaultCreator getInstance()

Description	Returns an instance of Class FaultCreator
Input arguments	-
Process	Returns an instance of Class FaultCreator
Output	Instance of SoapEnvelope FaultCreator
Exceptions	-
Comments	-

Public EnvelopeDocument createFault(QName faultCodeQName, String reason, EndpointReferenceType faultTo)

Description	This method creates EnvelopeDocument for given fault represented by faultCodeQName, reason and faultTo elements
Input arguments	QName faultCodeQName, String reason, EndpointReferenceType faultTo
Process	It creates EnvelopeDocument by calling method getInitializedEnvelopeDocument(faultTo). It also creates FaultDocument by calling method getFaultDocument(faultCodeQName, reason). And finally copies FaultDocument to EnvelopeDocument by calling method copyFaultIntoSOAPBody(EnvelopeDocument, FaultDocument)
Output	Returns EnvelopeDocument
Exceptions	-
Comments	-

public EnvelopeDocument createFault(QName faultCodeQName, String reason, XmlObject detailObject, EndpointReferenceType faultTo)

Description	This method creates EnvelopeDocument for given fault represented by faultCodeQName, reason, detailObject and faultTo elements
--------------------	---

Input arguments	QName faultCodeQName, String reason, XmlObject detailObject, EndpointReferenceType faultTo
Process	It creates EnvelopeDocument by calling method getInitializedEnvelopeDocument(faultTo). It also creates FaultDocument by calling method getFaultDocument(faultCodeQName, reason). Then it retrieves faultType from FaultDocument and process Detail Element by calling method processFaultDetailElement(faultType, detailObject). Finally copies FaultDocument to EnvelopeDocument by calling method copyFaultIntoSOAPBody(EnvelopeDocument, FaultDocument)
Output	Returns EnvelopeDocument
Exceptions	-
Comments	-

public EnvelopeDocument createFault(QName faultCodeQName, String reason, String faultActor, EndpointReferenceType faultTo)

Description	This method creates EnvelopeDocument for given fault represented by faultCodeQName, reason, faultActor and faultTo elements
Input arguments	QName faultCodeQName, String reason, String faultActor, EndpointReferenceType faultTo
Process	It creates EnvelopeDocument by calling method getInitializedEnvelopeDocument(faultTo). It also creates FaultDocument by calling method getFaultDocument(faultCodeQName, reason). Then it retrieves faultType from FaultDocument and copies faultActor to the faultType by calling method faultType.setFaultactor(faultActor). Finally it copies FaultDocument to EnvelopeDocument by calling method copyFaultIntoSOAPBody(EnvelopeDocument, FaultDocument)
Output	Returns EnvelopeDocument
Exceptions	-
Comments	-

public EnvelopeDocument createFault(QName faultCodeQName, String reason, XmlObject detailObject, String faultActor, EndpointReferenceType faultTo)

Description	This method creates EnvelopeDocument for given fault represented by faultCodeQName, reason, detailObject, faultActor and faultTo elements
Input arguments	QName faultCodeQName, String reason, XmlObject detailObject, String faultActor, EndpointReferenceType faultTo
Process	It creates EnvelopeDocument by calling method getInitializedEnvelopeDocument(faultTo). It also creates FaultDocument by calling method getFaultDocument(faultCodeQName, reason). Then it retrieves faultType from FaultDocument and copies faultActor to the faultType by calling method faultType.setFaultactor(faultActor). It also processes Detail Element by calling method processFaultDetailElement(faultType, detailObject). Finally it copies FaultDocument to EnvelopeDocument by calling method copyFaultIntoSOAPBody(EnvelopeDocument, FaultDocument)

Output	Returns EnvelopeDocument
Exceptions	-
Comments	-

public class QNameLocator extends Object

This class helps to determine if a QName is present within the XML. This uses the XmlCursor interface to help with this operation. Note that if indeed there is such an element, the cursor is appropriately positioned at the right location. One can then use xmlText() to retrieve the XML text and parse accordingly to access sub-elements if any within the element.

public static QNameLocator getInstance()

Description	Returns an instance of Class QNameLocator
Input arguments	-
Process	Returns an instance of Class QNameLocator
Output	Instance of QNameLocator
Exceptions	-
Comments	-

public boolean locateQName(XmlCursor cursor, QName qname)

Description	Locate the QName based on the specified XmlCursor.
Input arguments	XmlCursor cursor, QName qname
Process	Locates the QName based on the specified xmlCursor
Output	true If successfully locate the QName else false
Exceptions	-
Comments	Please note that the cursor is positioned at the location where the element is successfully located. Note that since the cursor is positioned at the location at which the element is found, this method does not reposition the cursor's temporarily saved location <i>iff</i> it returns a true. Please make sure you use <i>cursor.pop()</i> to erase the stack storage of the temporary location if your program's logic so demands. Thus, if you did <i>cursor.push()</i> prior to calling this method, and if a QName has been located by this method. <i>cursor.pop()</i> at a subsequent time can lead to aberrant results. You need to do <i>cursor.pop()</i> again in such a scenario.

public int locateQNameAndDepth(XmlCursor cursor, QName qname)

Description	Locate the QName and returns depth based on the specified XmlCursor.
Input arguments	XmlCursor cursor, QName qname
Process	Locate the QName and returns depth based on the specified XmlCursor.
Output	Returns integer representing the depth of element
Exceptions	-

Comments	Please note that the cursor is positioned at the location where the element is successfully located. Note that since the cursor is positioned at the location at which the element is found, this method does not reposition the cursor's temporarily saved location <i>iff</i> it returns a true. Please make sure you use <i>cursor.pop()</i> to erase the stack storage of the temporary location if your program's logic so demands. Thus, if you did <i>cursor.push()</i> prior to calling this method, and if a QName has been located by this method. <i>cursor.pop()</i> at a subsequent time can lead to aberrant results. You need to do <i>cursor.pop()</i> again in such a scenario.
-----------------	--

public class MarshallingUtils extends Object

A utility class to Marshall and Unmarshall object and object bytes

public static MarshallingUtils getInstance()

Description	Returns an instance of Class MarshallingUtils
Input arguments	-
Process	Returns an instance of Class MarshallingUtils
Output	Instance of MarshallingUtils
Exceptions	-
Comments	-

public byte[] marshall(Object object) throws IOException

Description	Marshall an object into a byte stream
Input arguments	Object object
Process	Creates ByteArrayOutputStream Object. Creates ObjectOutputStream object by passing ByteArrayOutputStream Object as a parameter. Writes object to ObjectOutputStream by using method ObjectOutputStream.writeObject(object) and returns array of bytes of ByteArrayOutputStream by using method byteArrayOutputStream.toByteArray() method
Output	Returns array of bytes
Exceptions	IOException
Comments	-

public Object unmarshall(byte[] objectBytes) throws IOException, ClassNotFoundException

Description	This method Unmrhalls an object from a byte stream
Input arguments	byte[] objectBytes
Process	Creates ByteArrayInputStream Object from objectBytes. Creates ObjectInputStream object by passing ByteArrayInputStream Object as a parameter and finally creates Object by calling method ObjectInputStream.readObject()
Output	Returns instance of Object
Exceptions	IOException, ClassNotFoundException
Comments	-

public void writeObjectToStream(Object object, DataOutputStream dout) throws IOException

Description	This method Writes an object onto a DataOutputStream
Input arguments	Object object, DataOutputStream dout
Process	Creates a array of bytes from object by calling method marshall(object) and write array of bytes to DataOutputStream dout by using method dout.write(byte []);
Output	-
Exceptions	IOException
Comments	-

public Object readObjectFromStream(DataInputStream din) throws IOException, ClassNotFoundException

Description	This method Reads an object that was written using the writeObject(object, DataOutputStream) from a DataInputStream
Input arguments	DataInputStream din
Process	This method reads the bytes from DataInputStream and stores in the array of bytes, and finally returns the instance of Object by calling the method unmarshall(byte[]);
Output	Returns instance of Object
Exceptions	IOException ClassNotFoundException
Comments	-

public String readStringFromStream(DataInputStream din) throws IOException

Description	This method Reads a String from the underlying dataInput Stream
Input arguments	DataInputStream din
Process	This method reads the bytes from DataInputStream and stores in the array of bytes. and finally creates the String Object by calling method new String(byte [])
Output	Returns String Object
Exceptions	IOException
Comments	-

public void writeStringToStream(String toWrite, DataOutputStream dout) throws IOException

Description	This method Writes a String to the underlying stream
Input arguments	String toWrite DataOutputStream dout
Process	This method retrieves the bytes from String "toWrite" and stores in the array of bytes. and finally writes the array of bytes to DataOutputStream dout by using method dout.write(byte[]);
Output	-
Exceptions	IOException
Comments	-

public void writeEprToStream(EndpointReferenceType eprType, DataOutputStream dout) throws IOException

Description	This method Writes an endpoint reference to the underlying stream
Input arguments	EndpointReferenceType eprType, DataOutputStream dout

Process	This method creates EndpointReferenceDocument and sets its endpoint reference with eprType. Then it generates string representation of EndpointReferenceDocument and calls method writeStringToStream(EndpointReferenceDocument, dout) to write to underlying DataOutputStream.
Output	-
Exceptions	IOException
Comments	-

public EndpointReferenceType readEprFromStream(DataInputStream din) throws IOException

Description	This method Reads an EndpointReference from the underlying datainput stream
Input arguments	DataInputStream din
Process	This method calls readStringFromStream(din) method which reads the String from underlying stream. It creates EndpointReferenceDocument by calling method EndpointReferenceDocument.Factory.parse(String) by passing string returned from method readStringFromStream(din) method. And finally returns EndpointReferenceType by calling method EndpointReferenceDocument.getEndpointReference();
Output	Returns EndpointReferenceType
Exceptions	IOException
Comments	-

public void writeEnvelopeToStream(EnvelopeDocument envelopeDocument, DataOutputStream dout) throws IOException

Description	This method Writes an envelope document to the underlying stream
Input arguments	EnvelopeDocument envelopeDocument, DataOutputStream dout
Process	This method converts envelopeDocument to the string representation by calling method envelopeDocument.toString() and finally calls the method writeStringToStream(String, dout);
Output	-
Exceptions	IOException
Comments	-

public EnvelopeDocument readEnvelopeFromStream(DataInputStream din) throws IOException

Description	This method Reads an envelope document from the underlying stream
Input arguments	DataInputStream din
Process	It calls the method readStringFromStream(din) which reads and returns string from DataInputStream din. And finally creates and returns envelopeDocument by calling method EnvelopeDocument.Factory.parse(String)
Output	Returns EnvelopeDocument
Exceptions	IOException
Comments	-