

Functional Specification

Community Grids Lab, Indiana University

1. Introduction

The FIRMS implementation will provide Grid and Web Service applications with ability to interact reliably with each other based on the implementation of the WS-ReliableMessaging (WSRM) and the WS-Reliability (WSR) specification. FIRMS provide an implementation of the WSRM specification that was released in March 2004 and WSR specification that was released on August 2004. The WSRM protocol developed jointly by IBM, Microsoft and BEA can be found at <http://www6.software.ibm.com/software/developer/library/ws-reliablemessaging200403.pdf> The specification for WS-Reliability -- developed jointly by Fujitsu, Novell, Oracle, and Sun -- can be found at <http://docs.oasis-open.org/wsrn/2004/06/WS-Reliability-CD1.086.pdf>.

Please note that what we briefly describe in the subsequent section is a background and overview to the WS-ReliableMessaging and WS-Reliability specification. If one is looking for comprehensive details the best place for this is the corresponding specification. Any attempt to describe the specifications in detail, will end up making this document similar to the specification: which would be pointless.

1.1 A background on acknowledgements and reliable delivery

Entities involved in reliable messaging need to facilitate easy detection of errors in received sequences while also being able to fix these errors in sequences. In *sender-initiated* protocols a sender gets positive acknowledgments (ACKs) from all receivers periodically. A *positive acknowledgement* confirms the receipt of a specific event by a given receiver. This information along with the knowledge of the events, which an entity is supposed to receive, allows the identification of holes in the delivery sequence at any given node. The sender can then initiate retransmissions to fix these errors.

In *receiver-initiated* protocols errors in received sequences are detected at the receivers, This detection in turn triggers *negative acknowledgments* (NAK) to fix these holes in the delivered sequences and retrieve any previously undelivered events. In receiver initiated protocols the assumption at the sender is that the message has been received at the receiver unless indicated otherwise by the NAKs.

It should be noted that in sender-initiated protocols the error detection, initiation of error correction and the retransmission are all performed at the sender side. In receiver-initiated protocols the error detection and initiation of error corrections are performed at the receiver, while the retransmissions are performed by the sender. ACK based schemes can exist by themselves, while NAK based schemes cannot. This is because in a purely NAK based scheme there is no way for the sender to know for sure if a message was received and hence the sender can never clear the buffer allocated for messages that were sent by the sender.

1.2 The WSRM & WSR specifications

The specifications – WSR and WSRM – both of which are based on XML, address the issue of ensuring reliable delivery between two service endpoints. In this section we outline the similarities in the underlying principles that guide both these specifications. The similarities that we have identified are along the six related dimensions of acknowledgements, ordering and duplicate eliminations, groups of messages and quality of service, timers, security and fault/diagnostic reporting.

Both the specifications use positive acknowledgements to ensure reliable delivery. This in turn implies that error detections, initiation of error corrections and subsequent retransmissions of “missed” messages can be performed at the sender side. A sender may also proactively initiate corrections based on the non-receipt of acknowledgements within a pre-defined interval.

The specifications also address the related issues of ordering and duplicate detection of messages issued by a source. A combination of these issues can also be used to facilitate exactly once delivery. Both the specifications facilitate guaranteed exactly-once delivery of messages, a very important quality of service that is highly relevant for transaction oriented applications; specifically banking, retailing and e-commerce.

Both the specifications also introduce the concept of a *group* (also referred to as a *sequence*) of messages. All messages that are part of a group of messages share a common group identifier. The specifications explicitly incorporate support for this concept by including the group identifier in protocol exchanges that take place between the two entities involved in reliable communications. Furthermore, in both the specifications the qualities of service constraints that can be specified on the delivery of messages are valid only within a group of messages, each with its own group identifier.

The specifications also introduce timer based operations for both messages (application and control) and group of messages. Individual and group of messages are considered invalid upon the expiry of timers associated with them. Finally, the delivery protocols in the specifications also incorporate the use of timers to initiate retransmissions and to time out retransmission attempts.

In terms of security both the specifications aim to leverage the WS-Security specification, which facilitates message level security. Message level security is independent of the security of the underlying transport and facilitates secure interactions over insecure communication links.

The specifications also provide for notification and exchange of errors in processing between the endpoints involved in reliable delivery. The range of errors supported in these specifications can vary from an inability to decipher a message’s content to complex errors pertaining to violations in implied agreements between the interacting entities.

2. Product Description

This software provides an implementation of the implementation of the WS-ReliableMessaging (WSRM) and the WS-Reliability (WSR) specifications. The table below provides a comparison of the WSRM and WSR specifications.

Table 1: Comparing some of the features in WS-Reliability and WS-ReliableMessaging

	WS-Reliability	WS-ReliableMessaging
Defines	Defines elements and attributes in the header block of a SOAP envelope.	An XML based schema for elements that are needed for reliable messaging.
Related Specifications	SOAP, WS-Security	WS_addressing, WS-Policy, WS-Security
Companies Involved	Sun, Oracle, Fujitsu	IBM, Microsoft, BEA
Submission Status	Under consideration by OASIS to be a standard	Not submitted yet.
HTTP Bindings	Defines a separate HTTP binding of the protocol.	NO. Defines no such binding. However, SOAP’s HTTP binding can be leveraged.
Delivery modes supported	Unreliable, at-least-once, ordered-and-exactly-once	At most once, at least once, ordered and exactly-once.
Groups of messages	Identified by <code>GroupId</code> information associated with every message in sequence. Individual messages have numbering that increments sequentially.	Grouped together using <code>Sequence</code> element. Every <code>Sequence</code> element has a unique identifier, and a message number which increments sequentially.
Dedicated exchanges for creation and termination of message	NO	YES

groups		
Ordering/Duplication dependence	Order is always tied to Guaranteed delivery and cannot be separately specified.	Order is not necessarily tied to guaranteed delivery
Exchange/Specification of protocol constants	Through an abstract concept referred to as Agreement.	WS-Policy.
Message Numbering in a group of messages.	SequenceNumber starts at 0 for the first message in a group.	MessageNumber starts at 1 for the first message in a group.
Defines Message-Exchange patterns	Request/Response, One-way and Polling	Not defined
Negative acknowledgements	NO	YES. This enables receiver-initiated error corrections.
Acknowledging a range of messages	YES	YES
Acknowledgements for multiple Groups	YES	NO
Indication of faults in acknowledgements	YES	NO
Requesting acknowledgements	The AckRequested element is REQUIRED in every message for which reliable delivery needs to be ensured.	AckRequested is used to request the receiving entity to acknowledge the message received.
Time based expiry	Supports timer based expiry of both messages and groups.	Supports timer based expiry of both messages and groups.
Retransmissions	Triggered after receipt of a set of acknowledgements. A specified number of retry attempts are made.	Triggered by either positive or negative acknowledgments. Two other constants Retransmission Interval and exponential backoff . play a role
Security	Relies on WS-Security and assorted specifications	Relies on WS-Security and assorted specifications
Errors	Are notified through SOAP faults.	Are notified through SOAP faults. Fault processing is more sophisticated since one can leverage WS-Addressing's message information headers.

The FIRMS implementation will provide Grid and Web Service applications with ability to interact reliably with each other based on the implementation of the WS-ReliableMessaging (WSRM) and the WS-Reliability (WSR) specification.

3. Use Cases

As Web Services have matured the interactions that the services have between themselves have gotten increasingly complex and sophisticated. Web services can be composed easily from other services, and these services can be made to orchestrate with each other in dynamic fashion. Web services specifications have addressed issues such as security, trust, notifications, service descriptions, advertisements, discovery and invocations among others. These specifications can leverage, extend and interoperate with other specifications to facilitate incremental addition of features and capabilities. As web services have become dominant in the Internet and Grid systems landscape, a need to ensure guaranteed delivery of interactions (encapsulated in messages) between services has become increasingly important. This highly important and complex area was previously being addressed in the Web Services community using

homegrown, proprietary, application specific solutions. It should be noted that the terms guaranteed delivery and reliable delivery tend to be used interchangeably to signify the same concept.

Reliable delivery of messages is now a key component of the Web Services roadmap, with two promising, and competing, specifications in this area viz. WS-Reliability and WS-ReliableMessaging. We include support for both these specifications in this release of the FIRMS software.

The WSRM and WSR specifications delegate security related issues to the WS-Security specification, which facilitates message-level security. Nothing in the WSRM or WSR specification or this implementation of these specifications preclude 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.

This software will report faults (as outlined in the corresponding specifications) if there are malformed requests. We enumerate some of these below:

1. An acknowledgement received at an endpoint refers to an unknown group or sequence identifier.
2. A retransmission request refers to an unknown message number or sequence/group identifier.
3. If the group/sequence over which communications are being initiated have been terminated or have expired.

The WSRM and WSR specifications and the FIRMS software concerns itself only with the reliable delivery of messages from the source to the sink. 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

The results pertaining to our implementation of WSRM have appeared in the proceedings of the IEEE E-Science 2005 conference. For the purposes of the documentation of this software we have included the performance section of this paper below.

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. To detect outliers we first calculate the mean and standard deviation of the entire data set. This is then used to obtain a z-score for each data point,

according to following formula: $z_i = \frac{x_i - \bar{x}}{s}$ where \bar{x} is the mean and s is the standard

deviation of the original sample. If the z-value is greater than 3, the corresponding data point is deemed an outlier.

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. We have measured several relevant performance aspects of our implementation. We now proceed to discuss each of this in detail. A

synopsis of our results is also available in a separate table (Table 2) for the reader's perusal. This table lists the operation, the mean, the standard deviation, standard error, minimum and maximum values for the CPU bound latencies (in microseconds) and finally the memory utilization associated with the operation.

In our performance measurements we started off by measuring the time to create a SOAP messages within the Axis Web Services container (SOAPMessage) and using our XMLBeans representation of the SOAP schema (EnvelopeDocument). We found that the costs in terms of {latencies, memory utilization} for these operations were similar. For EnvelopeDocument the cost was {126.86 μ Secs, 2192 B} while for SOAPMessage this cost was around {117.34 μ Secs, 2192 B}. The standard deviation (and the corresponding standard error) was higher for SOAPMessage creation at 187.30 μ Secs. Since every interaction between web service endpoints are encapsulated within SOAP messages, these costs represent the minimum costs that such interactions may incur.

To facilitate deployments within Apache's Axis and Sun's JWSDP container, we have developed utilities which facilitate conversions between the SOAP representations — SOAPMessage and EnvelopeDocument. The cost to convert an EnvelopeDocument into a SOAPMessage was around {2627.54 μ Secs and 60816B} while the cost for converting a SOAPMessage into a EnvelopeDocument was around {827.58 μ Secs, 34424B}. One of the reasons for this disparity is that the Axis implementation renames the schema namespace qualifiers contained within the EnvelopeDocument.

Since WSRM heavily leverages the WS-Addressing specification we benchmarked some overheads related to WS-Addressing processing. Here, we first measured the costs associated with the creation of simple EPRs based on a simple URL String and the more elaborate EPR that contains the `wsa:ReferenceProperties` element. As might be expected the costs for the simple EPR (150.51 μ Secs, 2648B) were better than those for the more elaborate EPR (397.34 μ Secs, 7184B). Next, we measured the costs involved in the creation of a SOAP message, targeted to a specific EPR, with the most basic WSA fields — `wsa:To` and `wsa:MessageID` within the SOAP message. In the second case, we included additional elements such as `wsa:From`, `wsa:RelatesTo` and the `wsa:Action` field. In both these cases the created SOAP message conformed to the rules outlined in the WS-Addressing specification. Here we found that the cost for creating the SOAP message with basic WS-Addressing elements were (397.34, 7184B) while the cost for additional elements was (537.81 μ Secs, 13880B).

Upon receipt of a SOAP message, the first task that needs to be performed is the parsing of the SOAP message for the WS-Addressing elements. This is typically the precursor to further more specific parsing later on since the WSA elements indicate not only the semantic intent (`wsa:Action`) but also the context (`wsa:Relates`, `wsa:MessageID`) and also where errors need to be issued to in case there are problems. For example, once we have determined the semantic intent of a message from the `wsa:Action` to be a Create Sequence request, we may initiate operations to parse the `wsrc:CreateSequence` element within the Body of the SOAP message. In our benchmarks the cost for parsing the SOAP message for WS-Addressing elements was found to be {1224.752 μ Secs, 61024B}. Since this operation is performed for every SOAP message this is a cost that will be incurred during each interaction between the service endpoints.

Next, we measured the costs involved in the creation of a WSRM create sequence request (352.16 μ Secs, 16392B) and the response (335.21 μ Secs, 18160B) generated upon the receipt of this request. These costs are in addition to any costs involved due to communication overheads between the service endpoints.

For every message received from the hosting service endpoint at the SourceProcessor, the appropriate `wsrc:Sequence` is added. This contains the identifier associated with the previously created Sequence and the Message Number assigned to this message. We measured the costs involved in the creation of this `wsrc:Sequence` element (44.72 μ Secs, 2424B) and the costs involved in the addition (12.67 μ Secs, 464B) of this element to the SOAP message received at the SourceProcessor.

A WSRM sink is expected to acknowledge messages at regular intervals (based on the acknowledgement interval). We have measured the costs involved in the creation of wsrM:SequenceAcknowledgement document based on a set of Message Numbers. We found this cost to be (516.58 μSecs, 20624B). This cost includes the costs involved in the creation of the one or more wsrM:AcknowledgementRange elements which cover acknowledgements for a group of messages. Thus if one is acknowledging Message Numbers 1,2,3,4,5,7,8,9,11,12,13 there would be 3 acknowledgement ranges corresponding to 1–5, 7–9 and 11–13.

We also measured the costs involved in the creation of wsrM:TerminateSequence (24.66 μSecs, 2072B) and the time to create a WSRM Fault(519 μSecs, 18096) based on the rules outlined in the WSRM and WS-Addressing specifications.

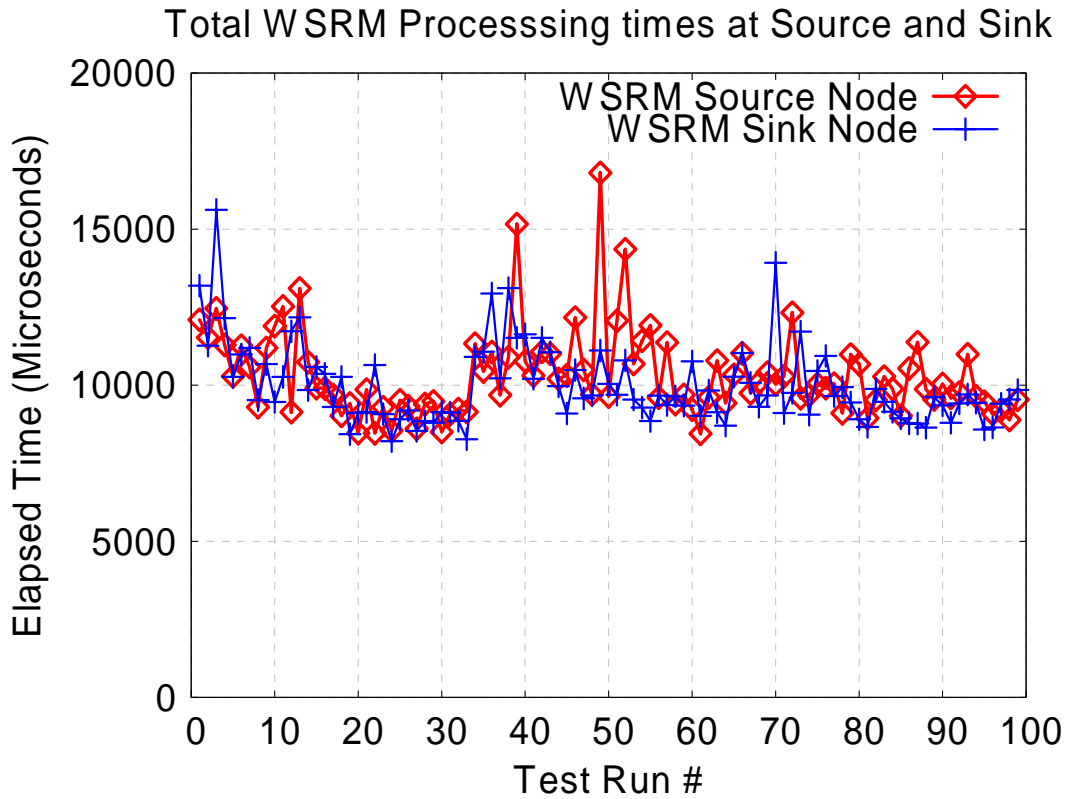


Figure 1: Total Processing times

Figure 1 depicts the total processing times at a WSRM source and sink. This includes the times for storage of message to stable storage at both source and sink. In our experiments the stable storage was based on MySQL. For MySQL we found the storage cost to be typically between 4-6milliseconds for message sizes 100B-10KB. Only after the SOAP message with the added wsrM:Sequence element has been stored onto stable storage will the message be routed to the remote sink endpoint. The graph does not include communication overheads involved in communication between the service endpoints. So these costs are in addition to the networking costs involved. In our experience we have found this cost to vary from a few milliseconds in LAN settings to a few hundred milliseconds in WAN settings.

Table 2: Summary of results (All results in Microseconds)

Operation	Mean	Standard Deviation	Standard Error	Num of Outliers	Min Value	Max Value	Memory Utilization (Bytes)
Create an XMLBeans based Envelope Document	126.864	49.395	5.041	4	108	424	2192

Create an Axis based SOAPMessage	117.340	187.302	19.017	3	34	1183	1824
Convert an EnvelopeDocument to a SOAPMessage	2627.548	905.483	93.894	7	1722	5350	60816
Convert SOAPMessage to EnvelopeDocument	827.589	586.872	60.211	5	325	2802	34424
Create a WS-Addressing EPR (Contains just a URL address)	87.562	58.590	5.979	4	71	465	2072
Create a WS-Addressing EPR (Contains WSA ReferenceProperties)	150.515	96.764	9.927	5	112	705	2648
Create an Envelope targeted to a specific WSA EPR	397.340	200.396	20.669	6	267	1276	7184
Create an Envelope targeted to a specific WSA EPR with most WSA message information headers	537.814	347.497	35.283	3	344	2123	13880
Parse an EnvelopeDocument to retrieve Wsa Message Info Headers	1224.752	727.870	73.904	3	645	4573	61024
CreateWsrSequenceRequest	352.163	260.997	26.364	2	229	1568	16392
CreateWsrSequenceResponse	335.210	226.060	23.193	5	224	1174	18160
CreateWsrSequenceDocument	44.724	4.733	0.478	2	42	75	2424
Add a WsrSequenceDocument to an existing envelope. (Contains sequence identifier and message number)	12.670	0.494	0.050	3	12	14	464
Create a WSRM SequenceAcknowledgement based on a set of message numbers	516.583	248.274	25.339	4	335	1514	20624
CreateTerminateSequence	24.666	36.203	3.638	1	19	380	2072
CreateWsrFault	519.802	294.699	30.077	4	347	1619	18096

5. Functional Design Considerations

Assumptions that were made during functional design: The WSRM and WSR specifications leverage 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 WSRand WSR specifications. 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.

Scale of deployment: We have tested the software in various configurations on different machines. In some cases we have also tested the source and the sink within the same machine using different OMII Containers.

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: We have implemented the two most dominant specifications in the area of reliable messaging viz. WS-Reliability and WS-ReliableMessaging.

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 both WSRM and WSR delegate 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 WSRM or WSR (or this implementation of the aforementioned specifications) 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 WSRM/WSR specifications. 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

package cgl.narada.wsinfra.wsrn.impl
public class WsrnAckOperationsImpl extends WsrnAckOperations

A utility class to deal with processing acknowledgements.

private void initializeWsrnElementCreation()

Description	This will initialize the WsrnElementCreation
Input arguments	-
Process	This method will initialize the Wsrn Element creation from WsrnProcessingFactory.
Output	-
Exceptions	-
Comments	-

public

SequenceAcknowledgementDocument.SequenceAcknowledgement.AcknowledgementRange[]

getAcknowledgementRanges(long[] acknowledgements)

Description	Construct acknowledgement ranges based on the specified acknowledgements.
Input arguments	long[] of acknowledgements
Process	This method will create Vectors seperateAcks and AckRanges. It will process acknowledgements. It will get the size of numOfSeperateAcks and if numOfSeperateAcks is greater than zero, it will create and add AcknowledgementRange.
Output	Array of Acknowledgement Range of Sequence Acknowledgement
Exceptions	-
Comments	-

private void processAcks(long[] acks, Vector seperateAcks, Vector ackRanges)

Description	This method will process acknowledges
Input arguments	Long[] of acks, Vector of separateAcks, Vector of ackRanges
Process	This method will check the acks for null value. It will deal with the case where there is only one message number to process. It will first sort the acks, since the algo will work on the sorted array. If successive elements are seperated by a value of 1 they can be part of a range. Finally it will create and add AcknowledgementRange using upperRange, lowerRange and ackRanges.
Output	-
Exceptions	-
Comments	-

private void createAndAddAcknowledgementRange(long upperRange, long lowerRange, Vector ackRangeVector)

Description	Create and add the acknowledgement range based on the parameters and add to the specified vector
Input arguments	long upperRange, long lowerRange, Vector ackRangeVector

Process	This method will check <code>wsrElementCreation</code> for null and <code>initializeWsrElementCreation</code> process. It will create <code>AcknowledgementRange</code> and adds to the <code>ackRangeVector</code> .
Output	-
Exceptions	-
Comments	-

`public long[] getAcknowledgements(SequenceAcknowledgementDocument seqAckDocument)`

Description	Retrieve the list of positive acknowledgements from the specified <code>sequenceAcknowledgement</code> document
Input arguments	<code>SequenceAcknowledgementDocument seqAckDocument</code>
Process	This method will get the <code>SequenceAcknowledgement</code> . It will create <code>seperateAcknowledgements</code> vector and <code>long[] seperateAcks</code> . It will process <code>Acknowledgement Range</code> if number of ranges is more than one. It will create the <code>seperateAcks</code> and sets the elements from <code>seperateAcknowledgements</code> vector.
Output	-
Exceptions	-
Comments	-

`private void processAcknowledgementRange(SequenceAcknowledgementDocument.SequenceAcknowledgement.AcknowledgementRange acknowledgementRange, Vector seperateIntoVector)`

Description	Process an acknowledgement range by trying to add the elements in the specified range into the <code>seperateIntoVector</code> . Also check to see if the element already exists in the vector, if so do not add
Input arguments	<code>AcknowledgementRange, Vector seperateIntoVector</code>
Process	This method will get the long of lower and upper values from <code>acknowledgement ranges</code> . It will add it to the <code>seperateIntoVector</code> Vector.
Output	-
Exceptions	-
Comments	-

`public long[] getMessageNumbersToAcknowledge(String sequenceIdentifier, boolean ackRequested, WsrProtocolStorageOperations wsrmProtocolOps) throws WsrStorageException`

Description	Retrieve the set of message numbers that need to be acknowledged for a sequence. The set of message numbers returned is the union of two sets (a) The message numbers that have already been acknowledged so far. (b) The set of unacknowledged message numbers. If the <code>ackRequested</code> variable is false, the message numbers included with this set are those for which the <code>ackInterval</code> has expired. If the <code>ackRequested</code> variable if it is true, THEN the <code>ackInterval</code> constraint is ignored, and ALL message numbers that have not been acknowledged are included in the set.
Input arguments	<code>String sequenceIdentifier, Boolean ackRequested, WsrProtocolStorageOperations wsrmProtocolOps</code>

Process	This method will check sequenceIdentifier and wsrnProtocolOps for null value and return null if any these are null. It will create long[] of messageNumbersToAck, ackedMessageNumbers and unackedMessageNumbers. It will combined and return the messageNumbersToAck.
Output	Long[]
Exceptions	-
Comments	-

private long[] getCombinedArrayOfLong(long[] first, long[] second)

Description	Combines the two arrays to create a unique array without any duplicate values in the combined array. This will also eliminate any duplicates that might exist in the original arrays
Input arguments	Long array of first, long array of second
Process	This method will check for first and second for null value and return the null value if any of those are null. Then it will create combined Vector and merger both first and second. It will create long[] and assign elements from combined Vector and return it.
Output	Long array
Exceptions	-
Comments	-

private static void addToVector(Vector vector, long value)

Description	Adds a message number to the vector
Input arguments	Vector, Long
Process	This method will check the given value in the vector and adds to it if not present in the vector.
Output	-
Exceptions	-
Comments	-

public class WsrnElementAdditionImpl extends WsrnElementAddition

This is a class which facilitates the addition of WSRM elements to a SOAP envelope. Depending on the exchange the elements are added either to the header or body of the SOAP envelope, with the Action Documents appropriately initialized.

public boolean addSequence(EnvelopeDocument envelopeDocument, SequenceDocument sequenceDocument)

Description	Adds a sequence document to the specified envelope. This method returns true if the operation succeed.
Input arguments	EnvelopeDocument, SequenceDocument
Process	This method checks for EnvelopeDocument and SequenceDocument for null values and it adds sequence to the envelope SoapHeader as last child element.
Output	Boolean value of true or false.
Exceptions	-
Comments	-

```
public boolean addAckRequested(EnvelopeDocument envelopeDocument,
AckRequestedDocument ackRequestedDocument)
```

Description	Adds an AckRequested document to the specified envelope. This method returns true if the operation succeed.
Input arguments	EnvelopeDocument, AckRequestedDocument
Process	This method checks for Envelope document and AckRequesteddocument for null values and adds ackRequested as a last element to the soapHeader.
Output	Boolean value of true or false.
Exceptions	-
Comments	-

```
public boolean addSequenceAcknowledgement(EnvelopeDocument envelopeDocument,
SequenceAcknowledgementDocument seqAckDocument)
```

Description	Adds a SequenceAcknowledgement element to the specified envelope. This method returns true if the operation succeed.
Input arguments	EnvelopeDocument
Process	This method will check for EnvelopeDocument, seqAckDocument for null value and return null if any one is null. It will add seqAckDocument as a last child to SoapHeader of the EnvelopeDocument.
Output	Boolean value of true or false
Exceptions	-
Comments	-

```
public boolean addTerminateSequence(EnvelopeDocument envelopeDocument,
TerminateSequenceDocument terminateSequenceDocument)
```

Description	Adds a terminate sequence to the specified envelope. This method returns true if the operation succeed.
Input arguments	EnvelopeDocument, TerminateSequenceDocument
Process	This method checks for envelopeDocument, TerminateSequenceDocument for null values and add terminate sequence to the soapBody of envelopeDocument.
Output	Boolean value of true or false
Exceptions	-
Comments	-

public class WsrnElementCreationImpl extends WsrnElementCreation

This is a class which facilitates the creation of WSRM elements that are added to exchanges.

```
public static WsrnElementCreation getInstance()
```

Description	This will returns the WsrnElementCreation class instance.
Input arguments	-
Process	This method will return the instance of this class.
Output	Instance of WsrnElementCreation
Exceptions	-
Comments	-

```
public SequenceDocument newSequence(String identifier, long messageNumber)
```

Description	Creates a sequence document based on the specified parameter
Input arguments	String identifier, long messageNumber
Process	This method will create sequenceDocument, adds new sequence to it. It adds new identifier and messageNumber.
Output	SequenceDocument
Exceptions	-
Comments	-

public SequenceDocument newSequence(String identifier, long messageNumber, boolean lastMessage, Calendar expiresAt)

Description	Creates a sequence document based on the specified parameters
Input arguments	String identifier, long messageNumber, Boolean lastNumber, Calender expiresAt
Process	This method will create the sequenceDocument and sequenceType. If the last message is true, it will new last message to sequence type. If expireAt is not null, it will create ExpiresDocument and copy from source to destination.
Output	SequenceDocument
Exceptions	-
Comments	-

public AckRequestedDocument newAckRequested(String identifier)

Description	Creates an AckRequested document based on the specified parameters
Input arguments	String of Identifier
Process	This method will get the AckRequestedDocument instance and adds the given identifier to it.
Output	AckRequestedDocument
Exceptions	-
Comments	-

public AckRequestedDocument newAckRequested(String identifier, long maxMessageNumberUsed)

Description	Creates an AckRequested document based on the specified parameters
Input arguments	String identifier, long maxMessageNumberUser
Process	This method gets the AckRequestedDocument based on given Identifier and sets the maximum message number used by setting input parameter.
Output	AckRequestedDocument
Exceptions	-
Comments	-

public SequenceAcknowledgementDocument newSequenceAcknowledgement(String identifier, long[] acknowledgementsArray, boolean positiveAcks)

Description	Creates a SequenceAcknowledgement element based on the specified parameters. If postiveAcks is true, then the SequenceACK contains positive acknowledgements otherwise it contains negative acknowledgements. The specification forbids inclusion of both positive and negative acknowledgements.
Input arguments	String of Identifier, long array of acknowledgements, Boolean of positive acks.

Process	This method gets the SequenceAcknowledgementDocument instance, adds sequenceAck and sets the Identifier. If positiveAcks exists, it will add to it.
Output	SequenceAcknowledgementDocument
Exceptions	-
Comments	-

public TerminateSequenceDocument newTerminateSequence(String identifier)

Description	Terminates a sequence identified the specified identifier
Input arguments	String of Identifier.
Process	This method will get the TerminateSequenceDocument instance and add terminateSequence to it. It will set the new Identifier to it.
Output	TerminateSequenceDocument.
Exceptions	-
Comments	-

public

SequenceAcknowledgementDocument.SequenceAcknowledgement.AcknowledgementRange
newAcknowledgementRange(long upperMessageNumber, long lowerMessageNumber)

Description	Create a new acknowledgement range based on the specified parameters
Input arguments	Long of upperMessageNumber, long of lowerMessageNumber
Process	This method gets the SequenceAcknowledgementDocument instance and sets the lower and upper message numbers to it.
Output	SequenceAcknowledgementDocument.
Exceptions	-
Comments	-

private BigInteger convertLongToBigInteger(long longValue)

Description	Converts a long into a BigInteger. This is the method which all such conversions should call. This way, if there is a more optimized way to do our conversions we just need to change it here and it impact every place. I am not very happy with this conversion, it is perhaps a little inelegant
Input arguments	Long of longValue
Process	This method will convert longValue into String and creates BigInteger object based on it.
Output	BigInteger
Exceptions	-
Comments	-

private BigInteger[] convertLongtoBigIntegerArray(long[] longValues)

Description	Converts an array of long into a Big Integer array
Input arguments	Long array of longValues
Process	This will converts long array into BigInteger array.
Output	BigInteger array
Exceptions	-
Comments	-

public class WsrElementCreationImpl extends WsrElementCreation

This is a utility class which constructs a WsrExchangeInfo based on the supplied Envelope Document.

public static WsrExchangeInfoCreator getInstance()

Description This class will returns the instance of the WsrExchangeInfoCreator class
 Input arguments -
 Process -
 Output WsrExchangeInfoCreator
 Exceptions -

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

public WsrExchangeInfo createWsrExchangeInfo(EnvelopeDocument envelopeDocument)

throws ProcessingException

Description Creates a WsrExchangeInfo based on the supplied envelope
 Input arguments EnvelopeDocument
 Process This method will get the addressing headers from envelopeDocument and creates WsrExchangeInfoImpl from it. It also checks for wsrActionElements and wsrElements in it.
 Output WsrExchangeInfo
 Exceptions ProcessingException

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

private void checkWsrActionElements(WsrExchangeInfoImpl wsrExchangeInfo, AddressingHeaders addressingHeaders)

Description Inspects the Action element contained within the Envelope to determine if it contains any of Wsr action headers
 Input arguments WsrExchangeInfoImpl, addressingHeaders
 Process -
 Output This method checks actionDocument and actions for null values. This also checks for createSequence, createSequenceResponse and terminateSequence.
 Exceptions -

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

private void checkForWsrElements(WsrExchangeInfoImpl wsrExchangeInfo, EnvelopeDocument envelopeDocument)

Description Inspects the WsrElements contained with in the Envelope.
 Input arguments WsrExchangeInfoImpl, Envelopedocument
 Process This method checks for WsrElements in the Envelope.
 Output -
 Exceptions -

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

public class WsrExchangeInfoImpl implements WsrExchangeInfo

This is a utility class which constructs a WsrExchangeInfo based on the supplied Envelope Document.

public boolean isValidExchange()

Description Checks to see if this is a valid exchange
 Input arguments -
 Process This method will check for hasSequence, has AckRequired, hasSeqAcknowledgement
 Output Boolean value of true or false
 Exceptions -

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

public class WsrnFaultsImpl implements WsrnFaults

This class provides a one-stop for creating all the faults that occur during WSRM processing. 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 SequenceFaultDocument getSequenceTerminated(String identifier)

Description	This fault is sent by either the RM Source or the RM Destination to indicate that the endpoint that generates the fault has either encountered an unrecoverable condition, or has detected a violation of the protocol and as a consequence, has chosen to terminate the sequence.
Input arguments	String of Identifier
Process	This will get the SequenceFaultDocument instance, it adds new sequence fault and Identifier to sequence fault.
Output	SequenceFaultDocument
Exceptions	-

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

public String getSequenceTerminatedReason()

Description	Retrieve the reason associated with the fault
Input arguments	-
Process	It will return reason for the sequence termination
Output	String value of fault
Exceptions	-

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

public SequenceFaultDocument getUnknownSequence(String identifier)

Description	This fault is sent by either the RM Source or the RM Destination in response to a message containing an unknown sequence identifier.
Input arguments	String value of Identifier
Process	This will creates new instance of SequenceFaultDocument and add Identifier to SequenceFault
Output	SequenceFaultDocument
Exceptions	-

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

public SequenceFaultDocument getInvalidAcknowledgement(SequenceAcknowledgementDocument seqAckDocument)

Description	This fault is sent by the RM Source in response to a <SequenceAcknowledgement> that violates the cumulative acknowledgement invariant. An example of such a violation would be a SequenceAcknowledgement covering messages that have not been sent.
Input arguments	SequenceAcknowledgementDocument
Process	It will create SequenceFaultDocument instance and set fault code as invalidacknowledgement.
Output	SequenceFaultDocument
Exceptions	-

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

public String getInvalidAcknowledgementReason()

Description Retrieve the reason associated with the fault
 Input arguments -
 Process It will return the String of Acknowledgement.
 Output String of fault
 Exceptions -

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

public SequenceFaultDocument getMessageNumberRollover(String identifier)

Description This fault is sent by the RM Source to indicate that it has run out of message numbers for a sequence. It is an unrecoverable error and terminates the Sequence.
 Input arguments String of Identifier
 Process It will create SequenceFaultDocument instance, sets Fault code and add Identifier to sequence fault.
 Output SequenceFaultDocument
 Exceptions -

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

public String getMessageNumberRolloverReason()

Description Retrieve the reason associated with the fault
 Input arguments -
 Process It will return message number rollover reason.
 Output String of fault
 Exceptions -

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

public SequenceFaultDocument getLastMessageNumberExceeded(String identifier)

Description This fault is sent by the RM Source to indicate that it has run out of message numbers for a sequence. It is an unrecoverable error and terminates the Sequence.
 Input arguments String of Identifier
 Process This gets the SequenceFaultDocument instance, adds new sequence fault and adds Identifier to the sequence fault.
 Output SequenceFaultDocument
 Exceptions -

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

public String getLastMessageNumberExceededReason()

Description Retrieve the reason associated with the fault
 Input arguments -
 Process This will return the fault message associated with last message number exceed reason.
 Output String of fault
 Exceptions -

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

public SequenceFaultDocument getSequenceRefused(String identifier)

Description This fault is sent by an RM Destination to indicate that it cannot begin a requested Sequence.

Input arguments	String of Identifier
Process	This will create SequenceFaultDocument, add new sequence fault to it and adds Identifier to it.
Output	SequenceFaultDocument
Exceptions	-
Comments	-

public String getSequenceRefusedReason()

Description	Retrieve the reason associated with the fault
Input arguments	-
Process	This will return fault associated with sequence refused reason.
Output	String of fault
Exceptions	-
Comments	-

public SequenceFaultDocument getCreateSequenceRefused()

Description	This fault is sent by an RM Destination to indicate that it cannot begin a requested Sequence.
Input arguments	-
Process	This will create new instance of SequenceFaultDocument, adds new sequence fault to it and sets fault code.
Output	SequenceFaultDocument
Exceptions	-
Comments	-

public SequenceFaultDocument getInvalidMessage()

Description	This is a fault send by an RM node when it encounters an invalid message.
Input arguments	-
Process	This will create SequenceFaultDocument instance, add new sequence fault and sets fault code to it.
Output	SequenceFaultDocument
Exceptions	-
Comments	-

public String getCreateSequenceRefusedReason()

Description	Retrieve the reason associated with the fault
Input arguments	-
Process	This will return the reason for create sequence refused.
Output	String of Fault
Exceptions	-
Comments	-

public QName getSoapSenderQName()

Description	Retrieves the Fault:Code for the sender
Input arguments	-
Process	This will return getSoapSender QName
Output	QName
Exceptions	-
Comments	-

public QName getSoapReceiverQName()

Description Retrieves the Fault:Code for the receiver
 Input arguments -
 Process This will return getSoapReceiver QName.
 Output QName
 Exceptions -

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

private void addIdentifierToSequenceFault(String identifier, SequenceFaultDocument sequenceFaultDocument)

Description This will add Identifier to Sequence Fault.
 Input arguments String of Identifier, SequenceFaultDocument.
 Process This will IdentifierDocument instance, add s new Identifier to it and set sequenceFault.

Output -
 Exceptions -

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

public class WsrnNodeFactoryImpl

This is a factory class which ensures that there is only instance of the WsrnSourceNode for a given configuration file. This class is primarily useful for deployment purposes.

public static WsrnNodeFactoryImpl getInstance()

Description This will return the instance of the WsrnNodeFactoryImpl
 Input arguments -
 Process This method will return the instance of this class.
 Output WsrnNodeFactoryImpl
 Exceptions -

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

public WsrnSourceNode getWsrnSourceNode(String configInfo) throws WsrnStorageException, DeploymentException

Description Gets a sequence monitor for the specified configuration file. For a given configuration file (based on the absolute path name) only one instance of the WsrnNode will be running within a given JVM instance.
 Input arguments String of config file information.
 Process This method will check for the configInfo for possible null values, existence and for absolute path. If a monitor does not exist, proceed to create one and add it to the monitors list.

Output -
 Exceptions WsrnStorageException, DeploymentException

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

public class WsrnNodeUtilImpl extends WsrnNodeUtils

This is a utility class which facilitates the retrieval of elements pertaining to WSRM that reside in the SOAP envelope.

public static WsrnNodeUtils getInstance()

Description	This class will returns the instance
Input arguments	-
Process	This will return the instance of the class
Output	WsrnNodeUtils
Exceptions	-
Comments	-

public CreateSequenceDocument getCreateSequenceDocument(EnvelopeDocument envelopeDocument)

throws WsFaultException

Description	Retrieves element from the body of the SOAP message
Input arguments	EnvelopeDocument
Process	This method gets the bodyType from the EnvelopeDocument, it creates Sequence Document and returns it.
Output	CreateSequencDocument
Exceptions	WsFaultException
Comments	-

public CreateSequenceResponseDocument
getCreateSequenceResponseDocument(EnvelopeDocument envelopeDocument) throws WsFaultException

Description	Retrieves element from the body of the SOAP message.
Input arguments	EnvelopeDocument
Process	This method gets the curser from BodyType, creates sequence response document based on curser text. It checks for null values and creates HandShake from it. It checks HandShake for null values and faults.
Output	CreateSequenceResponseDocument
Exceptions	WsFaultException
Comments	-

public TerminateSequenceDocument getTerminateSequenceDocument(EnvelopeDocument envelopeDocument) throws WsFaultException

Description	Retrieves element from the body of the SOAP message.
Input arguments	EnvelopeDocument
Process	This method gets the Curser from Body type, from EnvelopeDocument. It creates TerminateSequenceDocument from body curser and checks for null values.
Output	TerminateSequencDocument
Exceptions	WsFaultException
Comments	-

public IdentifierDocument getIdentifierDocument(XmlCursor xmlCursor) throws WsFaultException

Description	Retrieves the IdentifierDocument from within the speicified XmlCursor
Input arguments	XmlCursor
Process	This method gets the Identifier from WsrnQNames, creates IdentifierDocument from the xmlicursor text and returns the Identifier Document
Output	IdentifierDocument
Exceptions	WsFaultException
Comments	-

```

public                               SequenceAcknowledgementDocument
getSequenceAcknowledgement(EnvelopeDocument  envelopeDocument)  throws
WsFaultException
    Description      Retrieves the Sequence Acknowledgement document from within the
                    specified XmlCursor
    Input arguments  EnvelopeDocument
    Process          This method gets the headerType from EnvelopeDocument. It will retrieve
                    XmlCursor from HeaderType. It will create
                    SequenceAcknowledgementDocument from XmlCursor text.
    Output          SequenceAcknowledgementDocument
    Exceptions      WsFaultException
    Comments        -
    
```

```

public AckRequestedDocument  getAckRequested(EnvelopeDocument  envelopeDocument)
throws WsFaultException
    Description      Retrieves the AckRequested document from within the specified XmlCursor
    Input arguments  EnvelopeDocument
    Process          This method gets the headerType from EnvelopeDocument. It will retrieve
                    XmlCursor from HeaderType. It will then create AckRequestedDocument
                    from XmlCursor text.
    Output          AckRequestedDocument
    Exceptions      WsFaultException
    Comments        -
    
```

```

public SequenceDocument  getSequenceDocument(EnvelopeDocument  envelopeDocument)
throws WsFaultException
    Description      Retrieves the SequenceDocument from within the specified XmlCursor
    Input arguments  EnvelopeDocument
    Process          This method gets the headerType from EnvelopeDocument. It will retrieve
                    XmlCursor from HeaderType. It will then create SequenceDocument from
                    XmlCursor text.
    Output          SequenceDocument
    Exceptions      WsFaultException
    Comments        -
    
```

```

private void checkForElement(XmlCursor xmlCursor, QName qName) throws WsFaultException
    Description      Check to see if the element exists and position the cursor at the right location
                    if it does
    Input arguments  XmlCursor, QName
    Process          This will try to locate the QName given xmlCursor and qName. If it didn't
                    find the qName, it will through the exception.
    Output          -
    Exceptions      WsFaultException
    Comments        -
    
```

```

private BodyType  getEnvelopeBody(EnvelopeDocument  envelopeDocument)  throws
WsFaultException
    Description      Checks for problems with the envelope and returns the envelope body
    Input arguments  EnvelopeDocument
    
```

Process	This method will check the EnvelopeDocument for null values. It will get the BodyType from given EnvelopeDocument. It will check BodyType for null values and returns the BodyType.
Output	BodyType
Exceptions	WsFaultException
Comments	-

private HeaderType getEnvelopeHeader(EnvelopeDocument envelopeDocument) throws WsFaultException

Description	Checks for problems with the envelope and returns the envelope body
Input arguments	EnvelopeDocument
Process	This method will check the EnvelopeDocument for null values. It will get the HeaderType from given EnvelopeDocument. It will also check the HeaderType for null values and returns HeaderType.
Output	HeaderType
Exceptions	WsFaultException
Comments	-

private void throwInvalidMessageFault(String reason) throws WsFaultException

Description	Throw a WsFaultException, initialized with the invalid message QName and the specified reason
Input arguments	String of reason
Process	This method will create WsFaultException from given reason and QName from wsrmlFaults. It will add this fault to SoapHeader.
Output	-
Exceptions	WsFaultException
Comments	-

public class WsrmlRequestCreatorImpl implements WsrmlRequestCreator

A class which is responsible for creating WSRM requests

public static WsrmlRequestCreator getInstance()

Description	This will return the instance of this class
Input arguments	-
Process	This method will return the instance of this class
Output	-
Exceptions	-
Comments	-

public EnvelopeDocument getCreateSequenceRequest(ToDocument to, EndpointReferenceType wsrmlSourceEpr)

Description	Create a CreateSequenceRequest based on the specified headers. The To, From and ReplyTo, FaultTo are appropriately constructed.
Input arguments	ToDocument to, EndpointReferenceType wsrmlSourceEpr

Process	This method will check the to and wsrmsourceEpr for null values. It will create FromDocument, ActionDocument, ReplyToDocument, ReplyAfterDocument and FaultToDocument. Then It will create EnvelopeDocument from above documents.
	Then it will create SequenceDocument and adds to the body of the EnvelopeDocument
Output	EnvelopeDocument
Exceptions	-
Comments	-

public class WsrmsResponseCreatorImpl implements WsrmsResponseCreator

A class which is responsible for creating WSRM requests

public static WsrmsResponseCreator getInstance()

Description	This will return the instance of this class
Input arguments	-
Process	This method will return the instance of this class
Output	-
Exceptions	-
Comments	-

public EnvelopeDocument createSequenceResponse(AddressingHeaders createSequenceRequestHeaders, EndpointReferenceType wsrmsSinkEpr, String sequenceIdentifier)

Description	This will create sequence response envelope Document
Input arguments	AddressingHeaders createSequenceRequestHeaders, EndpointReferenceType wsrmsSinkEpr, String sequenceIdentifier
Process	This method will creates EndpointReferenceType, FromDocument, ActionDocument, FaultToDocument and RelatesToDocument from the given parameters. Then It will create EnvelopeDocument from those documents. It will create SequenceResponseDocument and it creates HandShakeType and adds sequenceIdentifier to it. Finally it will add SequenceResponseDocument to SoapBody of EnvelopeDocument.
Output	EnvelopeDocument
Exceptions	-
Comments	-

public void addCreateSequenceResponse(EnvelopeDocument envelopeDocument, String sequenceIdentifier)

Description	Create a CreateSequenceResponse element and add to a previously created envelope document.
Input arguments	EnvelopeDocument, String sequenceIdentifier
Process	This method will create CreateSequenceResponseDocument, it will get the HandshakeType from it. It will set the sequenceIdentifier to the handshaketype to it. Then it adds to soapbody of EnvelopeDocument.
Output	-
Exceptions	-
Comments	-

public class WsrSequenceMonitorFactoryImpl implements WsrSequenceMonitorFactory

This is a factory class which ensures that there is only instance of the SequenceMonitorThread running for a given database. If there are multiple databases with multiple sources/sinks this factory class will still work fine.

public static WsrSequenceMonitorFactory getInstance()

Description This will return the instance of this class
 Input arguments -
 Process This method will return the instance of this class
 Output -
 Exceptions -

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

public WsrSequenceMonitor getWsrSequenceMonitor(String configInfo, WsMessageFlow wsMessageFlow) throws WsrStorageException, DeploymentException

Description Gets a sequence monitor for the specified configuration file. For a given configuration file (based on the absolute path name) only one instance of the wsrSequenceMonitor will be running within a given JVM instance.
 Input arguments String configInfo, WsMessageFlow
 Process This method will check configInfo and wsMessageFlow for null values and throws exception if they are null. It checks for configFile location and throws exception if the location doesn't exists. It gets the absolute path of configFile. Then It creates WsrSoapMonitor. If a monitor does not exist, proceed to create one and add it to the monitors list.
 Output WsrSequenceMonitor
 Exceptions WsrStorageException, DeploymentException

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

public class WsrSequenceMonitorImpl extends Thread implements WsrSequenceMonitor

This processor processes Sequences and determines if (a) Acknowledgements/Retransmissions need to be issued. (b) See if the Inactivity timeout on a sequence has expired, if so proceed to terminate the sequence

public void startServices()

Description Begin services related to sequence monitoring viz. issue acknowledgements and initiate retransmissions for sequences.
 Input arguments -
 Process This will start the services.
 Output -
 Exceptions -

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

public void stopServices()

Description Stop services related to sequence monitoring.
 Input arguments -
 Process This will stop the services.
 Output -
 Exceptions -

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

public void run()

Description	This will run the services.
Input arguments	-
Process	This will set the Sleep Interval, cycle through the sequences. It will print the termination Information.
Output	-
Exceptions	-
Comments	-

private void cycleThroughSequences() throws MessageFlowException, WsrmsStorageException

Description	This will cycle through the sequences.
Input arguments	-
Process	This method will get the list of Sequences from WsrmsSequenceInfoOps and checks for sentSequences for null values. If sentSequences not null, it will create SequenceInfo from sequenceIdentifier. It will check for inactive timeout and issue retransmission. It will get the received Sequences from list activeSequences and check for null values. If received sequence is not null, it will create WsrmsSequenceInfo from sequenceId. It will issue acknowledgements.
Output	-
Exceptions	MessageFlowException, WsrmsStorageException
Comments	-

public boolean checkInactivityTimeout(WsrmsSequenceInfo wsrmsSequenceInfo) throws WsrmsStorageException

Description	Checks to see if the inactivity interval has expired on the sequence in question.
Input arguments	WsrmsSequenceInfo
Process	This method will get the sequenceId, WsrmsSequenceInfoPolicies from wsrmsSequenceInfo. It will get the inactivity timeout from wsrmsSequencePolicies and timeOfLastActivity from WsrmsSequenceInfo. If the inactive time is greater than current time minus timeOfLastActivity, it will terminate and timeout is TRUE else FALSE.
Output	Boolean true or false
Exceptions	WsrmsStorageException
Comments	-

public boolean checkIfMessagesAvailable(WsrmsSequenceInfo wsrmsSequenceInfo) throws WsrmsStorageException

Description	Checks to see if any messages have been stored for this sequence in question.
Input arguments	WsrmsSequenceInfo
Process	This method will get the Identifier from WsrmsSequenceInfo, checks for availability in stored elements of WsrmsProtocolOps.
Output	Boolean true or false
Exceptions	WsrmsStorageException
Comments	-

public void checkToIssueAcknowledgements(WsrmsSequenceInfo wsrmsSequenceInfo) throws MessageFlowException, WsrmsStorageException

Description	This method checks to see if acknowledgements should be issued on a specific sequence. If there is a need to do so, it proceeds to issue acknowledgements on the sequence.
Input arguments	WsrSequenceInfo
Process	This method gets the Sequence Identifier from the WsrSequenceInfo. It will get the acknowledged message numbers, unacknowledged message numbers from wsrProtocolOps by passing identifier. It combines both message numbers. It prints the acknowledgements info. It will create SequenceAcknowledgementDocument. It will process the acknowledgements.
Output	-
Exceptions	MessageFlowException, WsrStorageException
Comments	-

private void printAcknowledgementsInfo(String sequenceId, long[] unackedMessageNumbers, long[] ackno

Description	Print diagnostic information regarding the unackedMessageNumbers and the set of messages that will be acknowledged
Input arguments	String of sequenceId, long [] of unackedMessageNumbers, long [] of acknowledgements.
Process	This method will print unacknowledged Message Numbers if they are not null.
Output	-
Exceptions	-
Comments	-

private void issueAcknowledgement(WsrSequenceInfo wsrSequenceInfo, SequenceAcknowledgementDocument sequenceAckDocument) throws MessageFlowException

Description	This method creates an envelope based on the specified wsrSequenceInfo and the SequenceAcknowledgement Document.
Input arguments	WsrSequenceInfo, SequenceAcknowledgementDocument
Process	This method will get the Source, Sink endpoint references from WsrSequenceInfo and it will create FromDocument and RelatesToDocument. It will create EnvelopeDocument based on those parameters. It will add sequenceAcknowledgements to it. It will enrout to network above EnvelopeDocument.
Output	-
Exceptions	MessageFlowException
Comments	-

public void checkToIssueRetransmissions(WsrSequenceInfo wsrSequenceInfo) throws WsrStorageException, MessageFlowException

Description	This method checks to see if retransmissions need to be issued on the specified sequence. If a need arises retransmissions are issued and the retransmission interval is reset. There should also be a way to ensure exponential backoff.
Input arguments	WsrSequenceInfo
Process	This method will get the SequenceId from the WsrSequenceInfo, messagenumbers not Acknowledgements from WsrProtocolOps. It will get WsrStorageWidgets from WsrProtocolOps and noOfWidgets To Retransmit. It will add AckRequested element to the envelope document and it will start retransmission.

Output	-
Exceptions	WsrnStorageException, MessageFlowException
Comments	-

private void retransmitMessage(WsrnStorageWidget wsrnStorageWidget, AckRequestedDocument ackRequestedDocument) throws MessageFlowException

Description	This will retransmit the message.
Input arguments	WsrnStorageWidget wsrnStorageWidget, AckRequestedDocument ackRequestedDocument
Process	This will get the EnvelopeDocument from the storagewidget. It will check for acknowledge Requested option, and then it will retransmit using EnrouteToNetwork method.
Output	-
Exceptions	MessageFlowException
Comments	-

private void enrouteToNetwork(EnvelopeDocument envelopeDocument) throws MessageFlowException

Description	This will send EnvelopeDocument over the network.
Input arguments	EnvelopeDocument
Process	This will get the SoapEnvelopeConversion instance and creates SoapMessage using it. It will send the SoapMessage using WsMessageFlow enrouteNetwork method.
Output	-
Exceptions	MessageFlowException
Comments	-

private long computeNewRetransmissionInterval(WsrnSequenceInfo wsrnSequenceInfo)

Description	Computes a new retransmission interval based on a sequence's policies.
Input arguments	WsrnSequenceInfo
Process	This method will create retransmission interval based wsrnsequencePolices and returns it.
Output	Long
Exceptions	-
Comments	-

private long[] getCombinedArrayOfLong(long[] first, long[] second)

Description	Combines the two arrays to create a unique array without any duplicate values in the combined array. This will also eliminate any duplicates that might exist in the original arrays
Input arguments	Long[] First, Long[] second
Process	This method will check for first and second long [] for null values and adds to Vector. It will convert again into long[] and returns it
Output	Long[]
Exceptions	-
Comments	-

private boolean hasAckRequestedElement(EnvelopeDocument envelopeDocument)

Description	Checks to see if the AckRequested element is present in the envelope.
Input arguments	EnvelopeDocument

Process	This method will get the HeaderType, XmlCursor from EnvelopeDocument. It will local the QName using WsrnQnames and return true if exists else false.
Output	Boolean true or false
Exceptions	-
Comments	-

public class WsrnSequencePoliciesImpl implements WsrnSequencePolicies

This interface encapsulates the WsrnPolicies associated with a sequence. Among the elements that can be set in the Policy associated with a Sequence are

- (a) Sequence creation, expiration and Termination
- (b) This also provides information on the retransmission interval and the inactivity timeout.

public boolean hasSequenceExpiration()

Description	Check to see if sequence expiration has been specified
Input arguments	-
Process	It will check for null values.
Output	Boolean true or false.
Exceptions	-
Comments	-

public Calendar getSequenceExpiration()

Description	Retrieve the specified sequence expiration
Input arguments	-
Process	It will return sequenceExpiration Calender
Output	Calender
Exceptions	-
Comments	-

protected void resetSequenceExpiration(Calendar sequenceExpiration)

Description	It will reset the SequenceExpiration.
Input arguments	Calender sequenceExpiration
Process	This method will reset the SequenceExpiration with input argument.
Output	-
Exceptions	-
Comments	-

public boolean hasInactivityTimeout()

Description	Check to see if an inactivity timeout has been specified
Input arguments	-
Process	This method will check inactiveTimeout for null values.
Output	Boolean true or false.
Exceptions	-
Comments	-

public InactivityTimeoutDocument getInactivityTimeout()

Description	Retrieve specified inactivity timeout info
Input arguments	-
Process	This will return the inactive timeout
Output	InactivityTimeoutDocument

Exceptions	-
Comments	-

protected void setInactivityTimeout(InactivityTimeoutDocument inactivityTimeout)

Description	This will set the inactive Timeout
Input arguments	InactiveTimeOutDocument
Process	This method will set the inactive timeout with given input parameter.
Output	-
Exceptions	-
Comments	-

public boolean hasRetransmissionInterval()

Description	Check if retransmission related policies have been specified
Input arguments	-
Process	This method will checks retransmission interval for null values.
Output	Boolean true or false.
Exceptions	-
Comments	-

public BaseRetransmissionIntervalDocument getRetransmissionInterval()

Description	Retrieve the retransmission interval related policies
Input arguments	-
Process	This will return the BAsEReTranmissonInterval Document.
Output	BaseRetrnasmissionIntervalDocument
Exceptions	-
Comments	-

protected void setRetransmissionInterval(BaseRetransmissionIntervalDocument retransmissionInterval)

Description	This will set the retransmission interval Document
Input arguments	BaseRetrnasmissionIntervalDocument
Process	This method will set the BaseRetrnasmissionIntervalDocument with given parameter.
Output	-
Exceptions	-
Comments	-

public boolean hasExponentialBackoff()

Description	Check if the exponential back off has been specified
Input arguments	-
Process	This method will check for Exponential BackOff for null values
Output	Boolean true or false.
Exceptions	-
Comments	-

public ExponentialBackoffDocument getExponentialBackoff()

Description	Retrieve the specified exponential back off
Input arguments	-
Process	This method will return the ExponentialBackOffDocument.
Output	ExponentialBackOffDocument
Exceptions	-

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

protected void setExponentialBackoff(ExponentialBackoffDocument exponentialBackoff)

Description	This will set the ExponentialBackoffDocument
Input arguments	ExponentialBackoffDocument
Process	This method will set the ExponentialBackOffDocument.
Output	-
Exceptions	-

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

public boolean hasAcknowledgementInterval()

Description	Check if an acknowledgement interval has been specified
Input arguments	-
Process	This method will check for Acknowledgement Interval for null values.
Output	Boolean true or false
Exceptions	-

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

public AcknowledgementIntervalDocument getAcknowledgementInterval()

Description	Retrieve the specified acknowledgement interval
Input arguments	-
Process	This method will return the AcknowledgementIntervalDocument.
Output	AcknowledgementIntervalDocument
Exceptions	-

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

protected void setAcknowledgementInterval(AcknowledgementIntervalDocument acknowledgementInterval)

Description	This will set the Acknowledgement Interval Document
Input arguments	AcknowledgementIntervalDocument
Process	This method will return the AcknowledgementIntervalDocument.
Output	-
Exceptions	-

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

public byte[] getBytes()

Description	Get the byte stream representation
Input arguments	-
Process	This method will create ByteArrayOutputStream, DataOutputStream instances. It will marshal into bytes stream and return it back.
Output	Byte[]
Exceptions	-

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

public class WsrSequencePoliciesImpl implements WsrSequencePolicies

This is a utility class which facilitates the creation of WsrSequence Policies, and also the addition of policy elements besides the serialization/de-serialization of the policies.

public static WsrSequencePolicyFactory getInstance()

Description This will return the instance
 Input arguments -
 Process This method will return the instance of this class
 Output WsrSequencePolicyImpl
 Exceptions -

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

public WsrSequencePolicies createWsrSequencePolicies(XmlObject xmlObject)

Description Construct a sequence policies instance using the XmlCursor instance that has been provided.
 Input arguments XmlObject
 Process This method will create XmlCursor, WsrSequencePoliciesImpl, AcknowledgementIntervalDocument, BaseRetransmissionIntervalDocument, InactivityTimeoutDocument, ExponentialBackoffDocument and Calendar of sequence expiration. Then it creates WsrSequencePolicies using above one.
 Output WsrSequencePolicies
 Exceptions -

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

public WsrSequencePolicies getDefaultWsrSequencePolicies()

Description Get a default wsr sequence policies element
 Input arguments -
 Process This method will create String of defaultTimingProfile using getDefaultTimingProfile() and creates PolicyDocument using defaultTimingProfile. It will create WsrSequencePolicies using PolicyDocument.
 Output WsrSequencePolicies
 Exceptions -

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

public void updateExpiry(WsrSequencePolicies wsrSequencePolicies, Calendar expiresAt)

Description Update the expiry associated with a sequence. This can happen due to a new expiry value being specified within a Sequence element.
 Input arguments WsrSequencePolicies, Calendar of expiresAt
 Process This method will check WsrSequencePolicies and expiresAt for null values. It will create WsrSequencePoliciesImpl using policies and update with Calendar for expire time
 Output -
 Exceptions -

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

public AcknowledgementIntervalDocument getAcknowledgementInterval(XmlCursor xmlCursor) throws ParsingException

Description Retrieves the acknowledgement interval from within the specified XmlCursor
 Input arguments XmlCursor
 Process This method will create String element "[AcknowledgementInterval] element" and QName from it. It will check for this QName. It will create AcknowledgementIntervalDocument using XmlCursor text and send it back.

Output	AcknowledgementIntervalDocument
Exceptions	ParsingException
Comments	-

public BaseRetransmissionIntervalDocument getRetransmissionInterval(XmlCursor xmlCursor) throws ParsingException

Description	Retrieves the retransmission interval from within the specified XmlCursor
Input arguments	XmlCursor
Process	This method creates String of Element "[BaseRetransmissionInterval] element." And QName. It will check QName and throws Exception if not present. It will create BaseRetransmissionIntervalDocument based on XmlCursor.
Output	BaseRetransmissionIntervalDocument
Exceptions	ParsingException
Comments	-

public InactivityTimeoutDocument getInactivityTimeout(XmlCursor xmlCursor) throws ParsingException

Description	Retrieves the Inactivity timeout from within the specified XmlCursor
Input arguments	XmlCursor
Process	This method will create String element "[InactivityTimeout] element." and QName. It will check QName and throws Exception if not present. It will create InactivityTimeoutDocument from XmlCursor.
Output	InactivityTimeoutDocument
Exceptions	ParsingException
Comments	-

public ExponentialBackoffDocument getExponentialBackoff(XmlCursor xmlCursor) throws ParsingException

Description	Gets the exponential from within the specified XmlCursor
Input arguments	XmlCursor
Process	This method will create String element "[ExponentialBackoff] element." and QName. It will check QName and throws exception if not present. It will create ExponentialBackoffDocument using XmlCursor.
Output	ExponentialBackoffDocument
Exceptions	ParsingException
Comments	-

public SequenceRefDocument getSequenceRef(XmlCursor xmlCursor) throws ParsingException

Description	Gets the SequenceRef Document from within the specified XmlCursor
Input arguments	XmlCursor
Process	This method will create String element "[SequenceRef] element." and QName. It will check QName and throws exception if not present. It will create SequenceRefDocument using XmlCursor text.
Output	SequenceRefDocument
Exceptions	ParsingException
Comments	-

public Calendar getExpires(XmlCursor xmlCursor) throws ParsingException

Description	Gets the expire from XmlCursor
Input arguments	XmlCursor

Process	This method will create String element "[Expires] element." and QName. It will check for QName and throws exception if not present. It will create ExpiresDocument from XmlCursor and Calendar from it.
Output	Calendar
Exceptions	ParsingException
Comments	-

private void fillInMissingPolicies(WsrSequencePoliciesImpl parsedPolicies)

Description	The first time defaultsForMissingPolicies is being created, it will also call this method. DO this check to account for that scenario.
Input arguments	- WsrSequencePoliciesImpl parsedPolicies
Process	This method will check defaultsForMissingPolicies for null values. It will create check for parsedPolicies for sequenceExpiration and creates Calendar sequence expiry and adds to parsedPolicies. This method also checks parsedPolicies for Acknowledgement interval, inactivity timeout, retransmission interval and exponential backoff.
Output	-
Exceptions	-
Comments	-

private String getDefaultTimingProfile()

Description	The efficiency of WS-ReliableMessaging implementations can be improved using widely known timing profiles. The following profile can be found at: http://schemas.xmlsoap.org/ws/2004/03/rm/baseTimingProfile.xml
Input arguments	-
Process	This method will return the String representation of " <code><wsp:Policy xmlns:wsp=\"http://schemas.xmlsoap.org/ws/2004/09/policy\" xmlns:wsm=\"http://schemas.xmlsoap.org/ws/2004/03/rm\"> <wsm:BaseRetransmissionInterval Milliseconds=\"3000\" wsp:Usage=\"wsp:Observed\" /> <wsm:ExponentialBackoff wsp:Usage=\"wsp:Observed\" /> <wsm:InactivityTimeout Milliseconds=\"86400000\" wsp:Usage=\"wsp:Observed\" /> <wsm:AcknowledgementInterval Milliseconds=\"1000\" wsp:Usage=\"wsp:Observed\" /> </wsp:Policy></code> "
Output	String of defaultTimeProfile.
Exceptions	-
Comments	-

public WsrSequencePolicies getWsrSequencePolicies(byte[] marshalledBytes)

Description	Create a sequence policies instance based on its marshaled representation.
Input arguments	Byte[] of marshaled Bytes.
Process	This method will check marshalledBytes for null values and creates WsrSequencePolicies by passing marshalledBytes.
Output	WsrSequencePolicies
Exceptions	-
Comments	-

public SequenceRefDocument getSequenceRefDocument(EnvelopeDocument envelopeDocument)

Description	Get the sequenceRef document. The way this method works is as follows. SequenceRef elements should be contained within the wsp:AppliesTo element which in turn must be encapsulated within a PolicyAttachment element. If any of the predecessors are missing, this method will return a NULL.
Input arguments	EnvelopeDocument
Process	This method will create SequenceRefDocument and XmlCursor. It will check for foundPolicyAttachment. If it is not found, it will return the created sequenceRefDocument. Then it will create PolicyAttachmentDocument and gets the AppliesToDocument.
Output	SequenceRefDocument
Exceptions	-
Comments	-

public class WsrnSinkNode extends WsProcessor implements WsrnSink

This interface outlines the functionality of a WSRM sink.

public void setMessageFlow(WsMessageFlow wsMessageFlow) throws DeploymentException

Description	Sets the message flow which the processor should use
Input arguments	WsMessageFlow
Process	This method will set the WsMessageFlow to the SinkNodeHelper class.
Output	-
Exceptions	DeploymentException
Comments	-

public WsMessageFlow getMessageFlow()

Description	Gets the message flow which the processor should use.
Input arguments	-
Process	This method will return the WsMessageFlow
Output	WsMessageFlow
Exceptions	-
Comments	-

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 will create WsrnExchangeInfo from EnvelopeDocument and AddressingHeaders from WsrnExchangeInfo. It will call SinkNode Helper CheckExchangeInfo() method for valid exchanges. If the direction is from Application, It will not process that message and returns true. If the message originated from Network, it will check for appropriate action and call the method associated with that action. If there is no action associated with message, it will through Fault and return true back.
Output	Boolean of true or false
Exceptions	UnknownExchangeException, IncorrectExchangeException, ProcessingException, MessageFlowException

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

public void processCreateSequence(EnvelopeDocument envelopeDocument, AddressingHeaders addressingHeaders) throws WsFaultException, MessageFlowException, WsrsmStorageException, ProcessingException

Description	Process CreateSequence received over the network, from the source
Input arguments	EnvelopeDocument envelopeDocument, AddressingHeaders addressingHeaders
Process	This method will prepare the response envelope based on the specified addressing headers, report problems if there are any. It will proceed to issue the envelope document to Network by calling enroutetoNetwork() method.
Output	-
Exceptions	WsFaultException, MessageFlowException, WsrsmStorageException, ProcessingException

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

public CreateSequenceResponseDocument processCreateSequence(CreateSequenceDocument createSequenceDocument) throws WsFaultException, MessageFlowException

Description	Process a CreateSequence request received from the source
Input arguments	CreateSequenceDocument createSequenceDocument
Process	It will return default value of NULL
Output	CreateSequenceResponseDocument
Exceptions	WsFaultException, MessageFlowException

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

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

Description	Process wsrsm message received over the network, from the source
Input arguments	EnvelopeDocument envelopeDocument, AddressingHeaders addressingHeaders
Process	This method will create SequenceDocument and check to see if there are any problems whatsoever within the sequence element. Specifically, check for logic related to the following wsrsm sub-elements viz. sequenceIdentifier, messageNumber and lastMessage. It will check for duplicate sequenceDocument and gets the messagenumber if the duplicate is present.
Output	-
Exceptions	WsFaultException, MessageFlowException, WsrsmStorageException

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

private void storeWsrsmMessage(EnvelopeDocument envelopeDocument, WsrsmSequenceInfo wsrsmSequenceInfo, long messageNumber) throws WsrsmStorageException

Description	Store the Envelope document corresponding to the messageNumber and sequenceIdentifier onto stable storage.
Input arguments	EnvelopeDocument envelopeDocument, WsrsmSequenceInfo wsrsmSequenceInfo, long messageNumber
Process	This method will create WsrsmSequencePolicies from wsrsmSequenceInfo. It will get the acknowledgement interval from sequence policies. It gets the sequence Identifier from wsrsmSequenceInfo and stores it in the database using WsrsmStorageWidget class.
Output	-
Exceptions	WsrsmStorageException

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

public void processAckRequested(EnvelopeDocument envelopeDocument, AddressingH

Description	Process AckRequested received over the network, from the source
Input arguments	EnvelopeDocument envelopeDocument, AddressingHeaders addressingHeaders
Process	This method will create AckRequestedDocument from EnvelopeDocument and check for any problems that may exist within the [wsrm:AckRequested] element. It creates the Sequence Identifier from wsrmSequenceInfo and Long [] of acknowledgements. It will then create SequenceAcknowledgementDocument, acknowledgementEnvelope and adds seqAckDocument to envelope. It will enroute to the network created Envelope. Process the acknowledgements and tags the messages to indicate that they have been acknowledged. This needs to be the last step since by the time the call reaches here, the acknowledgement is already in transit over the network.
Output	
Exceptions	WsFaultException, MessageFlowException, WsrmStorageException, ProcessingException

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

public void processTerminateSequence(EnvelopeDocument envelopeDocument, throws WsFaultException, MessageFlowException, WsrmStorageException

Description	Process TerminateSequence received over the network, from the source
Input arguments	EnvelopeDocument envelopeDocument, AddressingHeaders addressingHeaders
Process	This method will create TerminateSequenceDocument from EnvelopeDocument and check for any problems that may exist within the [wsrm:TerminateSequence] element. It will proceed to terminate sequence and update database with termination info.
Output	-
Exceptions	WsFaultException, MessageFlowException, WsrmStorageException

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

public void processTerminateSequence(TerminateSequenceDocument terminateSequenceDocument)

throws WsFaultException, MessageFlowException

Description	Process the TerminateSequence message received from the source (over the network)
Input arguments	TerminateSequenceDocument
Process	This method will process the terminate sequence.
Output	-
Exceptions	WsFaultException, MessageFlowException

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

public EndpointReferenceType getEndpointReference ()

Description	Retrieves the endpoint reference associated with this node
Input arguments	-
Process	This method will return the sinkEpr.
Output	EndPointReference
Exceptions	-

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

public void setEndpointReference(EndpointReferenceType endpointReference)

Description	Retrieves the endpoint reference associated with this node
Input arguments	EndpointReferenceType
Process	This will set the sinkEpr from given EndPointReference using SinkNodeHelper class.
Output	-
Exceptions	-

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

public class WsrmsSinkNodeHelper extends WsProcessor

This interface outlines the functionality of a WSRM sink.

public static WsrmsSinkNodeHelper getInstance()

Description	Returns the instance of the class
Input arguments	-
Process	This method will return the instance of WsrmsSinkNodeHelper class
Output	WsrmsSinkNodeHelper
Exceptions	-

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

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

Description	This will process the EnvelopeDocument.
Input arguments	EnvelopeDocument, int direction
Process	This method will process EnvelopeDocument received over the network/application.
Output	Boolean of true or false.
Exceptions	UnknownExchangeException, IncorrectExchangeException, ProcessingException, MessageFlowException

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

public void setMessageFlow(WsMessageFlow wsMessageFlow) throws DeploymentException

Description	Sets the message flow which the processor should use
Input arguments	WsMessageFlow
Process	
Output	This method will set the message flow of WsProcessor
Exceptions	DeploymentException

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

public WsMessageFlow getMessageFlow()

Description	Gets the message flow which the processor should use.
Input arguments	-
Process	This will return the messageFlow
Output	WsMessageFlow
Exceptions	-

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

public void initialize(WsrStorageService wsrStorageService) throws DeploymentException

Description	Initiates the storage services.
Input arguments	WsrStorageService
Process	This method will check WsrStorageService for null values. It will initiate the wsrProtocolOps, wsrPolicyOps and wsrAuditOps from wsrStorageService.
Output	-
Exceptions	DeploymentException
Comments	-

public EndpointReferenceType getEndpointReference ()

Description	Retrieves the endpoint reference associated with this node
Input arguments	-
Process	This method will return the sinkEpr.
Output	EndPointReference
Exceptions	-
Comments	-

public void setEndpointReference(EndpointReferenceType endpointReference)

Description	Retrieves the endpoint reference associated with this node
Input arguments	EndpointReferenceType
Process	This will set the sinkEpr from given EndPointReference
Output	-
Exceptions	-
Comments	-

public EnvelopeDocument prepareCreateSequenceResponseEnvelope(AddressingHeaders addressingHeaders) throws WsFaultException, ProcessingException, WsrStorageException

Description	<p>Prepares a create sequence response based on the addressing headers contained in the original request. This method performs the following functions.</p> <ul style="list-style-type: none"> (a) Checks to see if the required elements [wsa:To] & [wsa:MessageID] are present. If they are not a ProcessingException or FaultException is thrown respectively. (b) Prepare the SOAP envelope with the appropriate destination, [wsa:Action] and [wsa:RelatesTo] fields. (c) Create a new sequenceIdentifier, proceed to add the appropriate CreateSequenceResponse document based on this. (d) Attach policy information to the envelope. (e) Create the wsrSequenceInfo element and add it to the database.
Input arguments	AddressingHeaders
Process	<p>This method will create EndPointReferenceType to from addressingHeaders, from from fromDocument. It will create ActionDocument and actionString. It will throw exception if addressingHeaders doesn't have messageId. Then It will create MessageIdDocument and RelatesIdDocument.</p> <p>Then it will create new EnvelopeDocument based on those created classes. Create a sequence identifier, and add the create sequence document to the newly created envelope. It will create a WsrSequenceInfo object and store it. Finally returns the EnvelopeDocument.</p>

Output	EnvelopeDocument
Exceptions	WsFaultException, ProcessingException, WsrnStorageException
Comments	-

public WsrnSequencePolicies addPolicyAttachment(EnvelopeDocument envelopeDocument, String sequenceIdentifier)

Description	Adds a policyAttachment to the header of the provide envelope. Make sure that a header is present within the supplied envelope. This attachment is typically added to the header of the CreateSequenceResponse envelope document.
Input arguments	EnvelopeDocument envelopeDocument, String sequenceIdentifier
Process	This method will create PolicyAttachmentDocument and PolicyAttacment from policyAttachmentDocument. It will create ApplyToDocument and attach to the PolicyDocument. It will create SequenceRefDocument and SequenceRefType from seqRefDocument. It will set the Identifier to the sequenceRefType. Then it will create WsrnSequencePolicies and adds Policy to the newly created WsrnSequencePolicies. It will add the PolicyAttachmentDocument to the SoapHeaders and return the WsrnSequencePolicies.
Output	WsrnSequencePolicies
Exceptions	
Comments	-

private void addWsrnPolicy(PolicyDocument.Policy policy, XmlObject wsrnPolicyObject)

Description	Adds the wsrn policy object to the specified policy
Input arguments	PolicyDocument.Policy policy, XmlObject wsrnPolicyObject
Process	This method will check for policy and wsrnPolicyObject for null values. I will copy policy and wsrnPolicyObject from source to destination using xmlContentTransfer.
Output	-
Exceptions	-
Comments	-

public WsrnSequenceInfo checkSequenceElementForProblems(AddressingHeaders addressingHeaders, SequenceDocument sequenceDocument)
throws WsrnStorageException, WsFaultException

Description	Upon receipt of a sequence this method checks to see if any problems exist within the element. Specifically we check for the following problems (a)Make sure that there is a wsu:Identifier element in Sequence, and that it is not of a NULL value. (b)Make sure that we know about the sequence. (c)Make sure that this sequence is not terminated. (d)Check to see that the message number is NOT ZERO. (e)Check to see that the LastMessageNumber constraint is not exceeded. (f)Check to see that the source is not trying to OVERRIDE a PREVIOUSLY established last message number. This method returns the WsrnSequenceInfo corresponding to the element; this is done to reduce the number of database accesses.
Input arguments	AddressingHeaders addressingHeaders, SequenceDocument sequenceDocument

Process This method will create SequenceType and AttributeURI from SequenceDocument and SequenceType respectively. It will check the attributedURI for problems. It creates the SequenceIdentifier and WsrSequenceInfo. It will check for the last message and big message number for null values. If the message number is equals to '0' it will throw exception. It will check to make sure that the last message info is not exceeded. Also make sure that the lastMessageNumber is constant, that is once set no one is allowed to change it to something else.

Then it will proceed to update the sequence information if this is indeed the last message of the sequence.

Output	WsrSequenceInfo
Exceptions	WsrStorageException, WsFaultException
Comments	-

public boolean checkIfDuplicate(SequenceDocument sequenceDocument) throws WsrStorageException

Description Checks to see if there is already a widget that has been stored corresponding to messageNumber and sequenceIdentifier encapsulated within the [wsrm:SequenceDocument].

Input arguments SequenceDocument

Process This method will get the SequenceType from SequenceDocument and sequenceIdentifier from SequenceType. It gets the messageNumber from SequenceType and it checks the stored element in wsrmProtocolOps. It returns true if presents.

Output	Boolean of true or false
Exceptions	WsrStorageException
Comments	-

public WsrSequenceInfo checkAckRequestedElementForProblems(AddressingHeaders addressingHeaders, AckRequestedDocument ackRequestedDoc) throws WsrStorageException, WsFaultException

Description Upon receipt of a sequence this method checks to see if any problems exist within the element. Specifically we check for the following problems
 (a)Make sure that there is a wsu:Identifier element in Sequence, and that it is not of a NULL value.
 (b)Make sure that we know about the sequence.
 (c)Make sure that this sequence is not terminated.
 (d)Check to see that the message number is NOT ZERO.
 (e)Check to see that the LastMessageNumber constraint is not exceeded.

Input arguments AddressingHeaders addressingHeaders, AckRequestedDocument ackRequestedDoc

Process	This method will get the AckRequestedType from ackRequestedDoc and attributeURI from ackRequestedType. It will check attributedURI for problems. It gets the SequenceIdentifier from attributedURI and WsrSequenceInfo. It will set the maximum message number used. If the maximum number used equals to '0' then it will throw exception. It will get the last message number from wsrSequenceInfo. If the maximum message number used is greater than last message number, it will through the exception.
Output	WsrSequenceInfo
Exceptions	WsrStorageException, WsFaultException
Comments	-

public EnvelopeDocument prepareEnvelopeForAcknowledgement(AddressingHeaders addressingHeaders, WsrSequenceInfo wsrSequenceInfo) throws WsFaultException, ProcessingException

Description	Prepares a SOAP envelope for the acknowledgement based on the specified addressingHeaders (contained within an AckRequested element) and the SequenceIdentifier contained within the WsrSequenceInfo.
Input arguments	AddressingHeaders addressingHeaders, WsrSequenceInfo wsrSequenceInfo
Process	This method will create EndPointReferenceType to, FromDocument from and Action Document. It will check the addressingHeaders for messageId and throws exception if messageId not present. It gets the relatesToId from WsrSequenceInfo. It creates RelatesToDocument and sets the relatesToId. Then it creates EnvelopeDocument based on those created above and sends it back.
Output	EnvelopeDocument
Exceptions	WsFaultException, ProcessingException
Comments	-

public WsrSequenceInfo checkTerminateSequenceElementForProblems(AddressingHeaders addressingHeaders, TerminateSequenceDocument terminateSequenceDocument) throws WsrStorageException, WsFaultException

Description	Upon receipt of a sequence this method checks to see if any problems exist within the element. Specifically we check for the following problems (a)Make sure that there is a wsu:Identifier element in Sequence, and that it is not of a NULL value. (b)Make sure that we know about the sequence. (c)Make sure that this sequence is not terminated. (d)Make sure that the sequence being terminated DOES NOT have any pending acknowledgements to issue.
Input arguments	AddressingHeaders addressingHeaders, TerminateSequenceDocument terminateSequenceDocument
Process	This method will create HandShakeType terminateSequence from terminateSequenceDocument and attributedURI from terminateSequence. It gets the sequenceIdentifier from attributedURI. It creates wsrSequenceInfo from addressingHeaders, sequenceIdentifier and enclosingDocument. It checks the pending acknowledgements and returns the wsrSequenceInfo.
Output	WsrSequenceInfo.

Exceptions	WsrnStorageException, WsFaultException
Comments	-

private WsrnSequenceInfo checkSequenceIdentifierForProblems(AddressingHeaders addressingHeaders, String sequenceIdentifier, String enclosingElement) throws WsrnStorageException, WsFaultException

Description	Make sure that we know about the sequence. Make sure that this sequence is not terminated.
Input arguments	AddressingHeaders addressingHeaders, String sequenceIdentifier, String enclosingElement
Process	This method will check for the known sequence and creates WsrnSequenceInfo using sequenceIdentifier in wsrnSequenceInfoOps. It will check wsrnSequenceInfo for termination and returns it.
Output	WsrnSequenceInfo
Exceptions	WsrnStorageException, WsFaultException
Comments	-

private void checkAttributedURIElementForProblems(AddressingHeaders addressingHeaders, Attributed

Description	Make sure that there is a wsu:Identifier element in Sequence, and that it is not of a NULL value.
Input arguments	AddressingHeaders addressingHeaders, AttributedURI attributedUri, String enclosingElement
Process	This method will check for the attributedURI for null values and throws exception if it is null.
Output	-
Exceptions	WsFaultException
Comments	-

public void throwLastMessageNumberExceededFaultException(AddressingHeaders addressingHeaders, String sequenceIdentifier, String additionalReason) throws WsFaultException

Description	Throws a WsFaultException corresponding to the LastMessageNumberExceeded fault.
Input arguments	AddressingHeaders addressingHeaders, String sequenceIdentifier, String additionalReason
Process	This method will create EndpointReferenceType faultTo and String reason from wsrnFaults. It will add additionalReason to it. It will create WsFaultException and SequenceFaultDocuement from sequenceIdentifier. It will add sequenceFaultDocuement to addressingHeaders of EnvelopeDocument.
Output	-
Exceptions	WsFaultException
Comments	-

public void throwSequenceTerminatedFaultException(AddressingHeaders addressingHeaders, String sequenceIdentifier, String additionalReason) throws WsFaultException

Description	Throws a WsFaultException corresponding to the sequence terminated fault.
Input arguments	AddressingHeaders addressingHeaders, String sequenceIdentifier, String additionalReason

Process	This method will create EndpointReferenceType faultTo and String reason from wsrnFaults. It will create WsFaultException and SequenceFaultDocuement from sequenceIdentifier. It will add sequenceFaultDocuement to addressingHeaders of EnvelopeDocument.
Output	-
Exceptions	WsFaultException
Comments	-

public void throwCreateSequenceRefusedFaultException(AddressingHeaders addressingHeaders, String additionalReason) throws WsFaultException

Description	Throws a CreateSequenceRefused Fault exception, if additional reason is not null, it will be added to the default reason element.
Input arguments	AddressingHeaders addressingHeaders, String additionalReason
Process	This method will create EndpointReferenceType faultTo and String reason from wsrnFaults. It will create WsFaultException and SequenceFaultDocuement . It will add sequenceFaultDocuement to addressingHeaders of EnvelopeDocument.
Output	-
Exceptions	WsFaultException
Comments	-

public void throwInvalidMessageFaultException(String reason, AddressingHeaders addressingHeaders) throws WsFaultException

Description	Throws an Invalid MessageFault exception based on the specified parameters
Input arguments	String reason, AddressingHeaders addressingHeaders
Process	This method will create EndpointReferenceType faultTo and String reason from wsrnFaults. It will create WsFaultException and SequenceFaultDocuement . It will add sequenceFaultDocuement to addressingHeaders of EnvelopeDocument.
Output	-
Exceptions	WsFaultException
Comments	-

public void throwUnknownSequenceFaultException(AddressingHeaders addressingHeaders, String identifier) throws WsFaultException

Description	Throws an UnknownSequence Fault exception based on the specified parameters
Input arguments	AddressingHeaders addressingHeaders, String identifier
Process	This method will create EndpointReferenceType faultTo and String reason from wsrnFaults. It will create WsFaultException and SequenceFaultDocuement . It will add sequenceFaultDocuement to addressingHeaders of EnvelopeDocument.
Output	-
Exceptions	WsFaultException
Comments	-

public void checkExchangeType(WsrnExchangeInfo wsrnExchangeInfo, int direction)

throws UnknownExchangeException, IncorrectExchangeException

Description	Checks an exchange type for validity. If there are problems exceptions are thrown.
Input arguments	WsrnExchangeInfo wsrnExchangeInfo, int direction
Process	This method will check the direction from where the message is originated. (Network or Application)
Output	-
Exceptions	UnknownExchangeException, IncorrectExchangeException
Comments	-

private void checkExchangeFromNetwork(WsrnExchangeInfo wsrnExchangeInfo) throws UnknownExchangeException, IncorrectExchangeException

Description	ONLY CreateSequence, TerminateSequence, Sequence & AckRequested valid exchanges that should be routed here.
Input arguments	WsrnExchangeInfo wsrnExchangeInfo
Process	This method will check for message exchange from NETWORK. It will check for following exchangeType. CreateSequenceResponse, SequenceAcknowledgement then throws problems. It will check for sequence, AckRequested, createSequence and terminateSequence.
Output	-
Exceptions	UnknownExchangeException, IncorrectExchangeException
Comments	-

private void checkExchangeFromApplication(WsrnExchangeInfo wsrnExchangeInfo) throws UnknownExchangeException, IncorrectExchangeException

Description	Only CreateSequence and WsrnMessages (messages without action elements or WSRM element) are valid.
Input arguments	WsrnExchangeInfo wsrnExchangeInfo
Process	This method will check for message exchange from Application. This will check for hasAckRequested, hasSequence, hasSequenceAcknowledgement, isCreateSequence and isTerminateSequence. Throws exception if above are in exchange type.
Output	-
Exceptions	UnknownExchangeException, IncorrectExchangeException
Comments	-

private void throwIncorrectExchangeException(String element, String from) throws IncorrectExchangeException

Description	Will throw incorrect Exchange Exception
Input arguments	String element, String from
Process	This method will create String of reason = "Should NOT have received exchange with element [" + element + "] from the " + from;
Output	-
Exceptions	IncorrectExchangeException
Comments	-

private void throwUnknownExchangeException(String element, String from) throws UnknownExchangeException

Description	Will throws unknown exchange exception
-------------	--

Input arguments	String element, String from
Process	This method will create String of reason = "No idea about processing exchange without elements [" + element + "] from the " + from + ".";
Output	
Exceptions	IncorrectExchangeException
Comments	-

public class WsrmSourceNode extends WsProcessor implements WsrmSource

This interface outlines the functionality of a WSRM sink.

public void setMessageFlow(WsMessageFlow wsMessageFlow) throws DeploymentException

Description	Sets the message flow which the processor should use
Input arguments	WsMessageFlow wsMessageFlow
Process	This will sets the messageFlow.
Output	-
Exceptions	DeploymentException
Comments	-

public WsMessageFlow getMessageFlow()

Description	Gets the message flow which the processor should use.
Input arguments	-
Process	This will returns the WsMessageFlow
Output	WsMessageFlow
Exceptions	-
Comments	-

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 envelopeDocument, int direction
Process	This method creates WsrmExchangeInfo and addressing headers from EnvelopeDocument and WsrmExchangeInfo respectively. IT will check for Fault in addressingHeaders and return true. It will check for exchange type in SourceNodeHelper. If the message direction is from Application, then it will processMessageFromAppication. If the message direction is from Network, it will process message based on Action type. If the action is createSequenceResponse, it will call processCreateSequenceResponse. If the action is has Acknowledgement, it will call processSequenceAcknowledgement method.
Output	Boolean of true or false.
Exceptions	UnknownExchangeException, IncorrectExchangeException, ProcessingException, MessageFlowException
Comments	-

public void processCreateSequenceResponse(EnvelopeDocument envelopeDocument, AddressingHeaders addressingHeaders) throws WsFaultException, WsrmStorageException

Description Process a create sequence response received from the sink

Input arguments	EnvelopeDocument addressingHeaders	envelopeDocument, AddressingHeaders
Process	This method will create CreateSequenceResponseDocument from EnvelopeDocument and it will create HandshakeType createSequenceResponse. It will create Sequence Identifier and String relatesTo. It will check relatesTo for null values and throws exception if is NULL. It will check for createSequenceResponse for relateTo key and throws exception if any.	
	It will create String destination and checks the sequenceId for that destination. If it exists, it will throw duplicate exception and if not exist, it will add it to the destination. Then it will retrieve any policy related information that may be available within the envelope. If none exist, proceed to fill this information up using default policies that are available within the policyFactory. It will also create a WsrSequenceInfo object and store it.	
Output	-	
Exceptions	WsFaultException, WsrStorageException	
Comments	-	

```
public void processSequenceAcknowledgement(EnvelopeDocument envelopeDocument,
AddressingHeaders addressingHeaders) throws WsFaultException, WsrStorageException,
MessageFlowException
```

Description	Process sequence acknowledgements received over the wire from the sink	
Input arguments	EnvelopeDocument, AddressingHeaders	
Process	This method will create SequenceAcknowledgementDocument and sequenceIdentifier from EnvelopeDocument. It will create long[] acknowledgements and check for null value. It will create long[] nacks and check for null value. It will check for retransmission and if so, proceed to do handle that. It will also check to see if the sequence needs to be terminated. If so, proceed to create and issue the appropriate termination sequence exchange	
Output	-	
Exceptions	WsFaultException, WsrStorageException, MessageFlowException	
Comments	-	

```
public void processMessageFromApplication(EnvelopeDocument envelopeDocument,
AddressingHeaders addressingHeaders) throws WsFaultException, MessageFlowException,
WsrStorageException
```

Description	Process exchange received from application, and send it across reliably	
Input arguments	EnvelopeDocument, AddressingHeaders	

Process This method will create ToDocument and destination. It will check the sequeceId for destination in the wsrSequenceInfoOps. It will create messageIdOfRequest and it will put in the createSequenceRequests. Then it will create sequenceIdentifier wsrSequenceInfo, previousMessagenumber, lastMessageOfSequence, ackRequested and new messageNumber.

If the lastMessageOfSequence is true, it will store created sequence. It will create Calendar expiresAt and adds sequenceDocument to the message. If ackRequested is true, it will add AckRequested element to the envelope document.

Output	-
Exceptions	WsFaultException, MessageFlowException, WsrStorageException
Comments	-

private void storeWsrMessage(EnvelopeDocument envelopeDocument, WsrSequenceInfo wsrSequenceInfo, long messageNumber) throws WsrStorageException

Description It will store Wsrmessage to the database.
Input arguments EnvelopeDocument envelopeDocument, WsrSequenceInfo wsrSequenceInfo, long messageNumber
Process This method will check the wsrSequencePolicies from wsrSequenceInfo for null values and create wsrSequencePolicies. Then it will create retransmissionInterval and sequenceIdentifier. It will create WsrStorageWidget and store all values.

Output	-
Exceptions	WsrStorageException
Comments	-

private String issueCreateSequenceRequest(ToDocument toDocument) throws MessageFlowException

Description This method issues a create sequence based on the specified toDocument. This method will also ensure that the request is sent over the network. The method returns the messageId of the created CreateSequenceRequest that was issued. This can be used to be part of the wait-notify scheme related to pausing operations up until the time that a response has been received over the wire.
Input arguments ToDocument
Process This method will create messageId and EnvelopeDocument from ToDocument and SourceEpr. It will create MessageIDDocument and check for null values. Then it will send the envelope over the network and returns messageId.

Output	String of messageId
Exceptions	MessageFlowException
Comments	-

private void initiateWaitOperation(String identifier)

Description This method creates an Object that is used as the basis of a wait()-notify() scheme. When a CreateSequenceResponse is received, the relatesTo element in that response (which will match this id) will be used to release the waiting.

Input arguments	String identifier
Process	This method will create Object syncObject and put the identifier and syncObject in it. It will Synchronize the syncObject.
Output	-
Exceptions	-
Comments	-

private void releasePendingWait(String identifier)

Description	This method uses the relatesTo element within a CreateSequenceResponse that is received to release a prior wait() operation that had been initiated.
Input arguments	String identifier
Process	This method will create Object syncObject and check for null values. It will Synchronize syncObject and notify it.
Output	-
Exceptions	-
Comments	-

public EndpointReferenceType getEndpointReference()

Description	Return EndPointReferene.
Input arguments	-
Process	This method will return Source End Point Reference.
Output	EndPointReference.
Exceptions	-
Comments	-

public void setEndpointReference(EndpointReferenceType endpointReference)

Description	Sets the EndpointReference
Input arguments	EndPointReference
Process	This method will set the Source EndPointReference
Output	-
Exceptions	-
Comments	-

private void reportError(String error)

Description	Will report error
Input arguments	String error
Process	This method will report error and prints it to console.
Output	-
Exceptions	-
Comments	-

public class WsrmsSourceNodeHelper extends WsProcessor

This is a utility class which performs several functions that are helpful for the processing logic related to the WsrmsSourceNode class.

public void initialize(WsrmsStorageService wsrmsStorageService) throws DeploymentException

Description	It will initialize storage service
Input arguments	WsrmsStorageService

Process	This method will check new WsrnStorageService for null values and throws exception if it is null. It will get the wsrnProtocolOps, wsrnSequenceInfoOps, wsrnPolicyOps and wsrnAuditOps from wsrnStorageServices,
Output	-
Exceptions	DeploymentException
Comments	-

public void setEndpointReference(EndpointReferenceType endpointReference)

Description	Sets the EndpointReference
Input arguments	EndPointReference
Process	This method will set the Source EndPointReference
Output	-
Exceptions	-
Comments	-

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

Description	It will process Exchange
Input arguments	EnvelopeDocument envelopeDocument, int direction
Process	It will create errorReport and check error report for null values and process error report. It will return false.
Output	Boolean true or false
Exceptions	UnknownExchangeException, IncorrectExchangeException, ProcessingException, MessageFlowException
Comments	-

public void setMessageFlow(WsMessageFlow wsMessageFlow) throws DeploymentException

Description	Sets the message flow which the processor should use
Input arguments	WsMessageFlow wsMessageFlow
Process	This will sets the messageFlow.
Output	-
Exceptions	DeploymentException
Comments	-

public WsMessageFlow getMessageFlow()

Description	Gets the message flow which the processor should use.
Input arguments	-
Process	This will returns the WsMessageFlow
Output	WsMessageFlow
Exceptions	-
Comments	-

public static WsrnSourceNodeHelper getInstance()

Description	Returns the instance of this class
Input arguments	-
Process	This method will return the instance of WsrnSourceNodeHelper class
Output	WsrnSourceNodeHelper
Exceptions	-
Comments	-

<p>public WsrSequencePolicies getWsrSequencePolicies(EnvelopeDocument envelopeDocument, String sequenceIdentifier) throws WsrStorageException</p>	
Description	<p>This method locates/retrieves/creates the WsrSequencePolicies associated with the sequenceIdentifier. There are Three steps involved here.</p> <p>Check to see if there is a SequenceRef document. IFF there is one, check to see if the match is wsr:Exact or wsr:Prefix. If it is wsr:Prefix proceed to store the created WsrSequencePolicies in the policyStorage operations. If a previous entry exists for the SAME PREFIX this should replace it.</p> <p>If there is no SequenceRef document, check to see if a prefix match exists, if there is one proceed to use the previously stored sequence policies.</p> <p>If there is NO sequenceRef document and NO matching PREFIX entries proceed to simply use the defaultWsrSequencePolicies.</p>
Input arguments	EnvelopeDocument envelopeDocument, String sequenceIdentifier
Process	<p>This method will create WsrSequencePolicies and SequenceRefDocument from EnvelopeDocument. It will check sequenceRefDocument for null values and creates wsrSequencePolicies and SequenceRefType. It will create QName matchType if sequenceRefType is match. It will create String of prefixIdentifier and store if it is not null.</p> <p>If there is no SequenceRef document, check to see if a prefix match exists, if there is one proceed to use the previously stored sequence policies. If there is NO sequenceRef document and NO matching PREFIX entries proceed to simply use the defaultWsrSequence Policies</p>
Output	WsrSequencePolicies
Exceptions	WsrStorageException
Comments	-

<p>public WsrSequenceInfo getWsrSequenceInfo(AddressingHeaders addressingHeaders, String sequenceIdentifier, String relatesTo, String destination, WsrSequencePolicies wsrSequencePolicies) throws WsrStorageException</p>	
Description	<p>Creates a WsrSequenceInfo based on the specified parameters. If the [wsa:From] is NULL or [wsa:ReplyTo] is NULL proceed to create a new EPR for the sink using the specified destination.</p>
Input arguments	AddressingHeaders addressingHeaders, String sequenceIdentifier, String relatesTo, String destination, WsrSequencePolicies wsrSequencePolicies
Process	<p>This method will create EndpointReferenceType sinkEpr from addressingHeaders and check for null value. If it is null, it will create new instance and sets the destination to it. It will create WsrSequenceInfo and attach to Policy to Sequence.</p>
Output	WsrSequenceInfo
Exceptions	WsrStorageException
Comments	-

public void checkForProblems(AddressingHeaders addressingHeaders, SequenceAcknowledgementDocument sequenceAckDocument, long[] acknowledgements) throws WsrStorageException, WsFaultException

Description	This method processes a set of acknowledgements received for a sequence. It checks, and issues faults if problems are encountered, for three distinct conditions.
	(a) Is this a known sequence? (b) If the sequence acknowledgement message numbers a valid one? (c) Has this sequence expired/terminated?
Input arguments	AddressingHeaders addressingHeaders, SequenceAcknowledgementDocument sequenceAckDocument, long[] acknowledgements
Process	This method will create SequenceAcknowledgement and AttributedURI. It will check AttributedURI for null value and throw exception if it is null. It will create sequenceIdentifier and check for known sequence. It will create WsrSequenceInfo and check for termination. If the acknowledgement is not null, it will create maxAckMessageNumber and check message number for problems.
Output	-
Exceptions	WsrStorageException, WsFaultException
Comments	-

public long[] getNegativeAcknowledgements(SequenceAcknowledgementDocument sequenceAckDocument)

Description	Retrieve the negative acknowledgements that have been specified within the sequence acknowledgement as an array of longs. The conversion of the datatypes needs to be handled by the method.
Input arguments	SequenceAcknowledgementDocument sequenceAckDocument
Process	This method will create long[] nacks and SequenceAcknowledgement. It will get the nack size from SequenceAcknowledgement and return it if it is equals to null.
Output	long[] of nacks
Exceptions	-
Comments	-

private void checkMessageNumberProblem(AddressingHeaders addressingHeaders, SequenceAcknowledgementDocument seqAckDocument, String sequenceIdentifier, long maxAckMessageNumber) throws WsrStorageException, WsFaultException

Description	Check to see if the specified maxAckNumber contained in acknowledgement is a valid one. If it is not proceed to construct a WsFaultException that contains the SequenceAcknowledgementDocument and throw that exception
Input arguments	AddressingHeaders addressingHeaders, SequenceAcknowledgementDocument seqAckDocument, String sequenceIdentifier, long maxAckMessageNumber
Process	This method will get the currentMaxMessageNumber and checks the maxAckNumber is greater than currentMaxMessageNumber and throws Exception.
Output	-
Exceptions	WsrStorageException, WsFaultException
Comments	-

public void manageRetransmissions(String sequenceIdentifier, long[] nacks) throws WsrMStorageException, MessageFlowException

Description	This method manages retransmissions based on the supplied negative acknowledgements and the sequence identifier. The sequence identifier is used to determine the message numbers from which the retransmission interval has expired and no acknowledgement has been received.
Input arguments	String sequenceIdentifier, long[] nacks
Process	This method will check acks, and determine missing numbers for which the ackInterval has expired. This will create the right AckRequested element. If there is a lastMessage number available, then the maxMessageNumberUsed element should be presented within the AckRequested element. This will get the numOfRetransmissions and stores the retransmissions and creates EnvelopeDocument. It will send the created EnvelopeDocument over the network.
Output	-
Exceptions	WsrMStorageException, MessageFlowException
Comments	-

public void processSequenceTermination(String sequenceIdentifier) throws WsrMStorageException, MessageFlowException

Description	Check to see if the sequence needs to be terminated, and if so, proceed to create and issue the appropriate termination sequence exchange
Input arguments	String sequenceIdentifier
Process	This method will create WsrMSequenceInfo and check WsrMSequenceInfo if it hasLastMessageInfo. The sequence cannot be terminated since the last message number has not yet been issued. It will get the long[] of stillToBeAked and check for null values. It will Proceed to terminate sequence. It will create TerminateSequenceDocument, EndpointReferenceType to , FromDocument from, ActionDocument action and String relatesTold. Then it will create EnvelopeDocument based on those created earlier and enroute to network. Proceed to remove the sequenceId info for the destination. This will ensure that the next time a message is sent to a destination, a new sequence identifier will be created.
Output	-
Exceptions	WsrMStorageException, MessageFlowException
Comments	-

public boolean isLastMessageOfSequence(EnvelopeDocument envelopeDocument)

Description	Check to see if an element which indicates that it is last message is present. If so return true, else return false.
Input arguments	EnvelopeDocument
Process	This method will get the String value associated with QName and check for null value. If value is true, it will return true else will return false.
Output	Boolean of true or false.
Exceptions	-
Comments	-

public boolean isAckRequested(EnvelopeDocument envelopeDocument)

Description	Check to see if an acknowledgement has been requested for this message. Check to see if the element exists, if it does return true; else return false.
Input arguments	EnvelopeDocument
Process	This method will get the String value associated with QName and check for null value. If value is true, it will return true else will return false.
Output	Boolean of true or false
Exceptions	-
Comments	-

public Calendar getSequenceExpiresAt(EnvelopeDocument envelopeDocument)

Description	Check to see if an acknowledgement has been requested for this message. Check to see if the element exists, if it does return true; else return false.
Input arguments	EnvelopeDocument
Process	This method will get the String value associated with QName and check for null value. If value is true, it will return Calender expiresAt null It will create Calender expireaAt and sets the time in mille seconds and returns it.
Output	Calender expiresAt
Exceptions	-
Comments	-

private String getValueAssociatedWithQName(EnvelopeDocument envelopeDocument, QName qName)

Description	Retrieves the value associated with a QName name within an Envelope. Note that this method is used ONLY for the proprietary QNames that are added by an application. These QNames are removed by this method once it retrieves the value.
Input arguments	EnvelopeDocument envelopeDocument, QName qName
Process	This method will check EnvelopedDocument, QName for null value. If EnvelopeType or setHeader is null, it will return null. It will get headerCursor and check for QName. If QName is null, it will return null else it will create String value and remove QName after it.
Output	String
Exceptions	-
Comments	-

public void checkForProblemsInToDocument(AddressingHeaders addressingHeaders) throws WsFaultException

Description	This method enforces the rule that the [wsa:To] element should be present within a SOAP header element. If this is not present an exception needs to be thrown.
Input arguments	AddressingHeaders addressingHeaders
Process	This method will get the ToDocument toDocument from addressingHeaders and check for null values.
Output	-
Exceptions	WsFaultException

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

public String getRelatesTo(EnvelopeDocument envelopeDocument)

Description	This method parses the SOAP envelope and retrieves the [wsa:RelatesTo] element within the SOAP header element as a String. This method returns a NULL if this element is NOT found.
Input arguments	EnvelopeDocument envelopeDocument
Process	This method will get the RelatesToDocument from addressinHeaders. It will get the Relationship from relatesToDocument and check for null value. It will return the StringValue of relationship.
Output	String relatesTo
Exceptions	-

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

public void checkForMessageRolloverFault(WsrSequenceInfo wsrSequenceInfo, long previousMessageNumber) throws WsFaultException, WsrStorageException

Description	Checks to see if a message number rollover is about to occur. That is the message number has a value of Long.MAX_VALUE. If this happens, the sequence needs to be terminated (the corresponding WSRM sequence info being stored) and a WsFault exception needs to be issued.
Input arguments	WsrSequenceInfo wsrSequenceInfo, long previousMessageNumber
Process	This method will check previousMessageNumber for long MaxValue. If previousMessageNumber is reached max value, it will create terminateReason and terminate the sequence. It will store the created sequence. It will get the sequenceIdentifier, EndpointReferenceType faultTo from WsrSequenceInfo.
Output	-
Exceptions	WsFaultException, WsrStorageException

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

private void throwMessageRolloverFaultException(String identifier, EndpointReferenceType faultTo)

throws WsFaultException

Description	Throws a WsFaultException corresponding to a message number rollover exception.
Input arguments	String identifier, EndpointReferenceType faultTo
Process	This method will create the WsFaultException and SequenceFaultDocument. It adds WsFaultException to SoapHeaderElement and throws WsFaultException.
Output	-
Exceptions	WsFaultException

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

public void throwInvalidMessageFaultException(String reason, AddressingHeaders addressingHeaders)

throws WsFaultException

Description	Throws an Invalid MessageFault exception based on the specified parameters
Input arguments	String reason, AddressingHeaders addressingHeaders

Process	This method will get the EndpointReferenceType faultTo and create WsFaultException. It will also create SequenceFaultDocument and adds it to SoapHeaderElement of WsFauException. It will return the WsFaultEeception.
Output	-
Exceptions	WsFaultException
Comments	-

public void throwInvalidAcknowledgementFaultException(AddressingHeaders addressingHeaders, SequenceAcknowledgementDocument seqAckDocument, String additionalReason) throws WsFaultException

Description	Throws an Invalid SequenceAcknowledgement Fault exception based on the specified parameters
Input arguments	AddressingHeaders addressingHeaders, SequenceAcknowledgementDocument seqAckDocument, String additionalReason
Process	This method will get the EndpointReferenceType faultTo and String reason. This will create wsFaultException and gets the SequenceFaultDocument from wsrnFaults. It will set the sequenceFaultDocument to the addressingHeaders of WsFaultException.
Output	-
Exceptions	WsFaultException
Comments	-

public void throwUnknownSequenceFaultException(AddressingHeaders addressingHeaders, String identifier)

Description	Throws an UnknownSequence Fault exception based on the specified parameters
Input arguments	AddressingHeaders addressingHeaders, String identifier
Process	This method will get the EndpointReferenceType faultTo from addressingHeaders, and creates WsFaultException from wsrnFaults. It will get the SequenceFaultDocument and adds To SoapHeaderElement of WsFaultException.
Output	-
Exceptions	WsFaultException
Comments	-

public void checkExchangeType(WsrnExchangeInfo wsrnExchangeInfo, int direction) throws UnknownExchangeException, IncorrectExchangeException

Description	It will check exchange type.
Input arguments	WsrnExchangeInfo wsrnExchangeInfo, int direction
Process	It will check the direction where the message is coming from. (Network, Application) and calls the appropriate method.
Output	-
Exceptions	UnknownExchangeException, IncorrectExchangeException
Comments	See the below methods for exchangeType.

private void checkExchangeFromNetwork(WsrnExchangeInfo wsrnExchangeInfo) throws UnknownExchangeException, IncorrectExchangeException

Description	ONLY SequenceAcknowledgement(s) and CreateSequenceResponse(s) are valid exchanges that should be routed here.
Input arguments	WsrnExchangeInfo

Process	This method will check for valid exchanges from Network. This will check exchange types of AckRequested, Sequence, CreateSequence and TerminateSequence for problem elements. If the exchange type is createSequenceResponse, the exchange is a valid one. Proceed to return
Output	-
Exceptions	UnknownExchangeException, IncorrectExchangeException
Comments	-

private void checkExchangeFromApplication(WsrmExchangeInfo wsrmExchangeInfo) throws UnknownExchangeException, IncorrectExchangeException

Description	Only CreateSequence and WsrmMessages (messages without action elements or WSRM element) are valid.
Input arguments	WsrmExchangeInfo
Process	This method will check for valid exchanges from Application only. This will check exchange types of AckRequested, Sequence, CreateSequence, SequenceAcknowledgement and TerminateSequence for problem elements.
Output	-
Exceptions	UnknownExchangeException, IncorrectExchangeException
Comments	-

private void throwIncorrectExchangeException(String element, String from) throws IncorrectExchangeException

Description	It will throw incorrect exchange exception.
Input arguments	String element, String from
Process	It will create String reason of "Should NOT have received exchange with element [" + element + "]" from the " + from and throws the exception.
Output	-
Exceptions	IncorrectExchangeException
Comments	-

private void throwUnknownExchangeException(String element, String from) throws UnknownExchangeException

Description	It will throw unknown exchange exception
Input arguments	String element, String from
Process	This method will create String reason "No idea about processing exchange without elements [" + element + "]" from the " + from and throws exception.
Output	-
Exceptions	UnknownExchangeException
Comments	-

private long[] getCombinedArrayOfLong(long[] first, long[] second)

Description	Combines the two arrays to create a unique array without any duplicate values in the combined array. This will also eliminate any duplicates that might exist in the original arrays
Input arguments	long[] first, long[] second

Process	This method will check first and second for null values and return null if they are null. It will create combined Vector and adds both to it. It will create combined long[] and get the elements from Vector and returns long[].
Output	Long[]
Exceptions	
Comments	-

WSR Functional Specifications

Package: cgl.narada.wsinfra.wsr.impl

Class: public class WsrAckOperationsImpl implements WsrAckOperations

public static WsrAckOperations getInstance()

Description	Returns Instance of WsrAckOperationsImpl class.
Input arguments	-
Process	-
Output	WsrAckOperationsImpl
Exceptions	-
Comments	-

public Vector getNonSequenceReplyAcknowledgement (ResponseDocument responseDocument)

Description	Retrieves GroupIds with single acknowledged message in question from responseDocument.
Input arguments	ResponseDocument: responseDocument
Process	Retrieves NonSequenceReplies from the ResponseDocument. If NonSequenceReply is not faulted it adds GroupID associated with this NonSequenceReply to the vector. Finally returns the vector as an output.
Output	Vector of GroupIDs
Exceptions	-
Comments	-

public Vector getNonSequenceReplyMessageProcessingFailureFault(ResponseDocument responseDocument)

Description	Retrieves GroupIds with single faulted message (message processing failure fault) in question from responseDocument
Input arguments	ResponseDocument: responseDocument
Process	Retrieves NonSequenceReplies from the ResponseDocument. If NonSequenceReply is with message processing failure fault then it adds GroupID associated with this NonSequenceReply to the vector. Finally returns the vector as an output.
Output	Vector of GroupIDs
Exceptions	-
Comments	-

public Hashtable getNonSequenceReplyFault(ResponseDocument responseDocument)

Description	Retrieves Hashtable (GroupId and fault pair) for GroupIDs with single faulted message (other than message processing failure fault) in question from responseDocument
Input arguments	ResponseDocument: responseDocument

Process	Retrieves NonSequenceReplies from the ResponseDocument. If NonSequenceReply is with fault other than messageprocessingfailure fault then it adds GroupID and fault associated with this NonSequenceReply to the Hashtable and returns the Hashtable as an output.
Output	Hashtable (GroupID and Fault pair)
Exceptions	-
Comments	-

public Hashtable getSequenceRepliesAcknowledgement(ResponseDocument responseDocument)

Description	Retrieves Hashtable (GroupId and associated acknowledged sequence numbers in vector) for groups with multiple messages from Response document. Retrieves Sequence Replies acknowledgements.
Input arguments	ResponseDocument: responseDocument
Process	Retrieves Sequence Replies from the ResponseDocument. If Sequence Reply is not faulted then adds GroupID and acknowledged sequence number's vector associated with that sequence reply in Hashtable and returns the Hashtable as an output.
Output	Hashtable
Exceptions	-
Comments	-

public Hashtable getSequenceReplyMessageProcessingFailureFault(ResponseDocument responseDocument)

Description	Retrieves Hashtable(groupId and vector of sequence numbers) for groups with multiple messages and have Message Processing Failure Fault from responseDocument (Retrieves Message Processing Failure Faults for Sequence Reply from responseDocument)
Input arguments	ResponseDocument: responseDocument
Process	Retrieves Sequence Replies from the ResponseDocument. If Sequence Reply is faulted (Message Processing Failure Fault) then adds GroupID and faulted sequence number's vector associated with that sequence reply in Hashtable and returns the Hashtable as an output.
Output	Hashtable
Exceptions	-
Comments	-

public Hashtable getSequenceReplyFault(ResponseDocument responseDocument)

Description	Retrieves Hashtable (GroupId and Hashtable(sequenceNumber and fault name)) for groups with multiple messages and have faults other than Message Processing Failure Fault from responseDocument. The Hashtable contains those GroupIds which are not for retransmission.
Input arguments	ResponseDocument: responseDocument
Process	Retrieves Sequence Replies from the ResponseDocument. If Sequence Reply is faulted(other than Message Processing Failure Fault) then adds GroupID and Hashtable(sequence number and fault) associated with that sequence reply in Hashtable and returns the Hashtable as an output.
Output	Hashtable
Exceptions	-
Comments	-

`public String[] getGroupIds(ResponseDocument responseDocument)`

Description	Retrieve all GroupIds from Response Document for which receives acknowledgements and faults
Input arguments	ResponseDocument: responseDocument
Process	Retrieves NonSequenceReplies and SequenceReplies from the responseDocuent and retrieves associated GroupIDs in String array. Finally returns the String[] as an output.
Output	String[]
Exceptions	-
Comments	-

`public Hashtable createRanges(long[] seqNums)`

Description	Create ranges from an array of seqNums.
Input arguments	Long[] seqNums
Process	Creates Hashtable of ranges from sequence number's array and returns it as an output.
Output	Hashtable
Exceptions	-
Comments	-

`public Hashtable createRanges(Hashtable seqNumsAndFaults)`

Description	create ranges from Hashtable of sequence Numbers and faults
Input arguments	Hashtable seqNumsAndFaults
Process	Creates Hashtable of ranges from sequence number and fault's array and returns it as an output.
Output	Hashtable
Exceptions	-
Comments	-

`public boolean isMessageProcessingFailureFault(QName fault)`

Description	Checks whether the fault is Message Processing failure fault.
Input arguments	QName fault
Process	Returns true if fault is message processing failure fault.
Output	boolean
Exceptions	-
Comments	-

Class: public class WsrElementAdditionImpl implements WsrElementAddition

`public static WsrElementAddition getInstance()`

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

`public boolean addRequest(EnvelopeDocument envelopeDocument, RequestDocument requestDocument)`

Description	Adds requestDocument to the specified envelopeDocument.
Input arguments	EnvelopeDocument envelopeDocument, RequestDocument requestDocument
Process	This method internally calls

	addToSoapHeaderAsLastChild(EnvelopeDocument, RequestDocument) method of SoapMessageAlteration class. It returns true if operation succeeded otherwise returns false.
Output	boolean
Exceptions	-
Comments	-

public boolean addPollRequest(EnvelopeDocument envelopeDocument, PollRequestDocument pollRequestDocument)

Description	Adds pollRequestDocument to the specified envelopeDocument.
Input arguments	EnvelopeDocument envelopeDocument, PollRequestDocument pollRequestDocument
Process	This method internally calls addToSoapHeaderAsLastChild(EnvelopeDocument, PollRequestDocument) method of SoapMessageAlteration class. It returns true if operation succeeded otherwise returns false.
Output	boolean
Exceptions	-
Comments	-

public boolean addResponse(EnvelopeDocument envelopeDocument, ResponseDocument responseDocument)

Description	Adds responseDocument to the specified envelopeDocument.
Input arguments	EnvelopeDocument envelopeDocument, ResponseDocument responseDocument
Process	This method internally calls addToSoapHeaderAsLastChild(EnvelopeDocument, ResponseDocument) method of SoapMessageAlteration class. And returns true if operation succeeded otherwise returns false.
Output	boolean
Exceptions	-
Comments	-

Class: public class WsrElementCreationImpl implements WsrElementCreation

public static WsrElementCreation getInstance()

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

public RequestDocument newRequest(String groupId, long sequenceNumber, boolean lastMessage, Calendar messageExpiryTime, String replyPatternName)

Description	Creates a Request Document based on the specified mandatory parameters. This creates a request for PollRequest and Response RM reply pattern.
Input arguments	String groupId, long sequenceNumber, boolean lastMessage, Calendar messageExpiryTime,

	String replyPatternName
Process	Creates new RequestDocument by calling RequestDocument.Factory.newInstance() method and adds all the specified parameters to the newly created RequestDocument. Finally returns RequestDocument as an output.
Output	RequestDocument
Exceptions	-
Comments	-

public RequestDocument newRequest(String groupId, long sequenceNumber, boolean lastMessage, Calendar messageExpiryTime, String replyPatternName, String replyTo)

Description	Creates a Request Document based on the specified parameters. This creates a request for Callback RM reply pattern
Input arguments	String groupId, long sequenceNumber, boolean lastMessage, Calendar messageExpiryTime, String replyPatternName String replyTo
Process	Creates RequestDocument by calling method newRequest(groupId, sequenceNumber, lastMessage, messageExpiryTime, replyPatternName) and adds replyTo element in RequestDocument. Finally returns RequestDocument as an output.
Output	RequestDocument
Exceptions	-
Comments	-

public RequestDocument newRequest(String groupId, Calendar groupExpiryTime, long sequenceNumber, boolean lastMessage, Calendar messageExpiryTime, String replyPatternName)

Description	Creates a Request Document based on the specified mandatory parameters with some optional parameters. This creates a request for PollRequest and Response RM reply pattern
Input arguments	String groupId, Calendar groupExpiryTime, long sequenceNumber boolean lastMessage, Calendar messageExpiryTime, String replyPatternName
Process	Creates RequestDocument by calling method newRequest(groupId, sequenceNumber, lastMessage, messageExpiryTime, replyPatternName) and adds groupExpiryTime element in RequestDocument. Finally returns RequestDocument as an output.
Output	RequestDocument
Exceptions	-
Comments	-

public RequestDocument newRequest(String groupId, GDuration groupMaxIdleDuration, long sequenceNumber, boolean lastMessage, Calendar messageExpiryTime, String replyPatternName)

Description	Creates a Request Document based on the specified mandatory parameters with some optional parameters. This creates a request for PollRequest and Response RM reply pattern
-------------	--

Input arguments	String groupId, GDuration groupMaxIdleDuration, long sequenceNumber boolean lastMessage, Calendar messageExpiryTime, String replyPatternName
Process	Creates RequestDocument by calling method newRequest(groupId, sequenceNumber, lastMessage, messageExpiryTime, replyPatternName) and adds groupMaxIdleDuration element in RequestDocument. Finally returns RequestDocument as an output.
Output	RequestDocument
Exceptions	-
Comments	-

public RequestDocument newRequest(String groupId, Calendar groupExpiryTime, long sequenceNumber, boolean lastMessage, Calendar messageExpiryTime, String replyPatternName, String replyTo)

Description	Creates a Request Document based on the specified mandatory parameters with some optional parameters. This creates a request for Callback RM reply pattern
Input arguments	String groupId, Calendar groupExpiryTime, long sequenceNumber boolean lastMessage, Calendar messageExpiryTime, String replyPatternName, String replyTo
Process	Creates RequestDocument by calling method newRequest(groupId, groupExpiryTime, sequenceNumber, lastMessage, messageExpiryTime, replyPatternName) and adds replyTo element in RequestDocument. Finally returns RequestDocument as an output.
Output	RequestDocument
Exceptions	-
Comments	-

public RequestDocument newRequest(String groupId, GDuration groupMaxIdleDuration, long sequenceNumber, boolean lastMessage, Calendar messageExpiryTime, String replyPatternName, String replyTo)

Description	Creates a Request Document based on the specified mandatory parameters with some optional parameters. This creates a request for callback RM reply pattern
Input arguments	String groupId, GDuration groupMaxIdleDuration, long sequenceNumber boolean lastMessage, Calendar messageExpiryTime, String replyPatternName, String replyTo
Process	Creates RequestDocument by calling method newRequest(groupId, groupMaxIdleDuration, sequenceNumber, lastMessage, messageExpiryTime, replyPatternName) and adds replyTo element

	in RequestDocument. Finally returns RequestDocument as an output.
Output	RequestDocument
Exceptions	-
Comments	-

`public PollRequestDocument newPollRequest(String[] groupId)`

Description	Creates a PollRequestDocument based on the specified parameters
Input arguments	String[] groupId
Process	Creates PollRequestDocument by calling method PollRequest.Factory.newInstance(). For each groupId it adds RefToMessageIds element and sets its groupId element with one of the specified groupIds. Finally returns PollRequestDocument as an output.
Output	PollRequestDocument
Exceptions	-
Comments	-

`public PollRequestDocument newPollRequest(String groupId)`

Description	Creates a PollRequestDocument based on the specified parameters
Input arguments	String groupId
Process	Creates PollRequestDocument by calling method newPollRequest(new String[]{groupId}). And Finally returns PollRequestDocument as an output.
Output	PollRequestDocument
Exceptions	-
Comments	-

`public void processPollRequest(PollRequestType pollRequestType, String groupId, Hashtable range)`

Description	Creates a PollRequestDocument based on the specified parameters
Input arguments	PollRequestType pollRequestType, String groupId, Hashtable range
Process	Creates PollRequestDocument for given groupId and its associated sequenceNumber ranges in Hashtable. Finally returns PollRequestDocument.
Output	PollRequestDocument
Exceptions	-
Comments	-

`public PollRequestDocument newPollRequest(String groupId, Hashtable range)`

Description	Creates a PollRequestDocument based on the specified parameters
Input arguments	String groupId, Hashtable range
Process	Creates PollRequestDocument for given groupId and its associated sequenceNumber ranges in Hashtable. Finally returns PollRequestDocument.
Output	PollRequestDocument
Exceptions	-
Comments	-

`public PollRequestDocument newPollRequest(Hashtable groupIds)`

Description	Creates a PollRequestDocument based on the specified parameters
Input arguments	Hashtable groupIds
Process	Creates PollRequestDocument for given groupIds and all its associated sequenceNumbers. Finally returns PollRequestDocument.
Output	PollRequestDocument
Exceptions	-
Comments	-

`public PollRequestDocument newPollRequest(String groupId, String replyTo)`

Description	Creates a PollRequestDocument based on the specified parameters
Input arguments	String groupId, String replyTo
Process	Creates PollRequestDocument for given groupId and replyTo element. Finally returns PollRequestDocument.
Output	PollRequestDocument
Exceptions	-
Comments	-

`public PollRequestDocument newPollRequest(String[] groupIds, String replyTo)`

Description	Creates a PollRequestDocument based on the specified parameters
Input arguments	String[] groupIds, String replyTo
Process	Creates PollRequestDocument for given groupIds and replyTo element. Finally returns PollRequestDocument.
Output	PollRequestDocument
Exceptions	-
Comments	-

`public PollRequestDocument newPollRequest(String groupId, Hashtable range, String replyTo)`

Description	Creates a PollRequestDocument based on the specified parameters
Input arguments	String groupId, Hashtable range, String replyTo
Process	Creates PollRequestDocument for given groupId its associated sequence numbers and replyTo element. Finally returns PollRequestDocument.
Output	PollRequestDocument
Exceptions	-
Comments	-

`public ResponseDocument newResponse()`

Description	Creates a simple ResponseDocument with single Response element
Input arguments	-
Process	Creates ResponseDocument by calling ResponseDocument.Factory.newInstance() method and returns ResponseDocument.
Output	ResponseDocument
Exceptions	-

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

public ResponseDocument newResponse(ResponseDocument responseDocument, String groupId, String fault)

Description	Adds a NonSequenceReply element (group with single message) to given ResponseDocument.
Input arguments	ResponseDocument responseDocument, String groupId, String fault
Process	Adds a NonSequenceReply element(group with single message) to given ResponseDocument.
Output	ResponseDocument
Exceptions	-
Comments	-

public ResponseDocument newResponse(ResponseDocument responseDocument, String groupId, Hashtable rangesAndFaultTable)

Description	Adds a SequenceReply element(group with multiple messages)To given ResponseDocument Here Hashtable contains (Hashtable of ranges(from and to), fault or ack) pair
Input arguments	ResponseDocument responseDocument, String groupId, Hashtable rangesAndFaultTable
Process	Adds a SequenceReply element(group with multiple messages)To given ResponseDocument
Output	ResponseDocument
Exceptions	-
Comments	-

public ResponseDocument newResponse(ResponseDocument responseDocument, String groupId, long from, long to, String fault)

Description	Adds a SequenceReply element(group with multiple messages)To given ResponseDocument with only one ReplyRange
Input arguments	ResponseDocument responseDocument, String groupId, long from, long to, String fault
Process	Adds a SequenceReply element(group with multiple messages)To given ResponseDocument
Output	ResponseDocument
Exceptions	-
Comments	-

Class: public class WsrExchangeInfoCreatorImpl implements WsrExchangeInfoCreator
public static WsrExchangeInfoCreator getInstance()

Description	This will return instance of WsrExchangeInfoCreatorImpl class
Input arguments	-
Process	-
Output	WsrExchangeInfoCreatorImpl
Exceptions	-

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

public WsrExchangeInfo createWsrExchangeInfo(EnvelopeDocument envelopeDocument)
throws ProcessingException

Description	Creates a WsrExchangeInfo based on the supplied envelope
Input arguments	EnvelopeDocument envelopeDocument
Process	Retrieves addressingHeaders from envelopeDocument and creates WsrExchangeInfo by calling method new WsrExchangeInfo(addressingHeaders)
Output	WsrExchangeInfo
Exceptions	ProcessingException
Comments	-

private void checkWsrActionElements(WsrExchangeInfoImpl wsrExchangeInfo,
AddressingHeaders addressingHeaders)

Description	Inspects the Action element contained within the Envelope to determine if it contains any of Wsr action headers
Input arguments	WsrExchangeInfoImpl wsrExchangeInfo, AddressingHeaders addressingHeaders
Process	Retrieves action element from addressingHeaders and checks whether this action is one of the WsrActions like processRequest, processPollRequest and processResponse.
Output	-
Exceptions	-
Comments	-

private void checkForWsrElements(WsrExchangeInfoImpl wsrExchangeInfo,
EnvelopeDocument envelopeDocument)

Description	Inspects the Header of the envelopeDocument to determine if it contains any of Wsr Elements.
Input arguments	WsrExchangeInfoImpl wsrExchangeInfo, EnvelopeDocument envelopeDocument
Process	Retrieves the header of the envelopeDocument and checks whether this header contains one of the WsrElements like Request, Response and PollRequest.
Output	-
Exceptions	-
Comments	-

Class: public class WsrExchangeInfoImpl implements WsrExchangeInfo

public boolean hasRequest()

Description	Checks to see if the Request Element is available
Input arguments	-
Process	Returns hasRequest Element
Output	boolean
Exceptions	-
Comments	-

public void setRequest()

Description	Sets the hasRequest element to true
-------------	-------------------------------------

Input arguments	-
Process	Sets the hasRequest element to true
Output	-
Exceptions	-
Comments	-

public boolean hasPollRequest()

Description	Checks to see if the PollRequest Element is available
Input arguments	-
Process	Returns hasPollRequest Element
Output	boolean
Exceptions	-
Comments	-

public void setPollRequest()

Description	Sets the hasPollRequest element to true
Input arguments	-
Process	Sets the hasPollRequest element to true
Output	-
Exceptions	-
Comments	-

public boolean hasResponse()

Description	Checks to see if the Response Element is available
Input arguments	-
Process	Returns hasResponse Element
Output	boolean
Exceptions	-
Comments	-

public void setPollRequest()

Description	Sets the hasResponse element to true
Input arguments	-
Process	Sets the hasResponse element to true
Output	-
Exceptions	-
Comments	-

public boolean isProcessRequest()

Description	Checks to see if this is a Process Request exchange
Input arguments	-
Process	Returns isProcessRequest Element
Output	boolean
Exceptions	-
Comments	-

protected void setProcessRequest()

Description	Sets the isProcessRequest element to true
Input arguments	-
Process	Sets the isProcessRequest element to true

Output	-
Exceptions	-
Comments	-

public boolean isProcessPollRequest()

Description	Checks to see if this is a ProcessPollRequest exchange
Input arguments	-
Process	Returns isProcessPollRequest Element
Output	boolean
Exceptions	-
Comments	-

protected void setProcessPollRequest()

Description	Sets the isProcessPollRequest element to true
Input arguments	-
Process	Sets the isProcessPollRequest element to true
Output	-
Exceptions	-
Comments	-

public boolean isProcessResponse()

Description	Checks to see if this is a ProcessResponse exchange
Input arguments	-
Process	Returns isProcessResponse Element
Output	boolean
Exceptions	-
Comments	-

public boolean setProcessResponse()

Description	Sets the isProcessResponse element to true
Input arguments	-
Process	Sets the isProcessResponse element to true
Output	-
Exceptions	-
Comments	-

public AddressingHeaders getAddressingHeaders()

Description	Retrieve the addressing headers that were associated with this exchange
Input arguments	-
Process	Retrieve the addressing headers that were associated with this exchange
Output	AddressingHeaders
Exceptions	-
Comments	-

public String getProblemsWithExchange()

Description	Retruns the problemsWithExchange element
Input arguments	-
Process	Retruns the problemsWithExchange element

Output	String
Exceptions	-
Comments	-

private void addToProblemsWithExchange(String problems)

Description	Sets the problemsWithExchange with problems element
Input arguments	-
Process	Sets the problemsWithExchange with problems element
Output	-
Exceptions	-
Comments	-

public boolean isValidExchange()

Description	Checks to see if this is a valid exchange
Input arguments	-
Process	Returns isValidExchange element
Output	boolean
Exceptions	-
Comments	-

Class: public class WsrNodeUtilsImpl implements WsrNodeUtils

public static WsrNodeUtils getInstance()

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

public RequestDocument getRequestDocument(EnvelopeDocument envelopeDocument) throws WsFaultException

Description	Retrieves Request Document from the Header of the envelopeDocument
Input arguments	EnvelopeDocument envelopeDocument
Process	It retrieves Header from the EnvelopeDocument and retrieves Request Document from the header. If RequestDocument is null then throws WsFaultException by calling throwInvalidMessageFault() method.
Output	RequestDocument
Exceptions	WsFaultException
Comments	-

public PollRequestDocument getPollRequestDocument(EnvelopeDocument envelopeDocument) throws WsFaultException

Description	Retrieves PollRequest Document from the Header of the envelopeDocument
Input arguments	EnvelopeDocument envelopeDocument
Process	It retrieves Header from the EnvelopeDocument and retrieves PollRequestDocument from the header. If PollRequestDocument is null then throws WsFaultException by calling

	throwInvalidMessageFault() method.
Output	PollRequestDocument
Exceptions	WsFaultException
Comments	-

public ResponseDocument getResponseDocument(EnvelopeDocument envelopeDocument) throws WsFaultException

Description	Retrieves Response Document from the Header of the envelopeDocument
Input arguments	EnvelopeDocument envelopeDocument
Process	It retrieves Header from the EnvelopeDocument and retrieves ResponseDocument from the header. If ResponseDocument is null then throws WsFaultException by calling throwInvalidMessageFault() method.
Output	ResponseDocument
Exceptions	WsFaultException
Comments	-

public String getGroupId(EnvelopeDocument envelopeDocument) throws WsFaultException

Description	Retrieves GroupId Element from the Header of the EnvelopeDocument
Input arguments	EnvelopeDocument envelopeDocument
Process	It retrieves Header from the EnvelopeDocument and retrieves GroupID from the header. If GroupID is null then returns null else returns String associated with the GroupID element.
Output	String
Exceptions	WsFaultException
Comments	-

public String getGroupExpiryTime(EnvelopeDocument envelopeDocument) throws WsFaultException

Description	Retrieves GroupExpiryTime Element from the Header of the EnvelopeDocument
Input arguments	EnvelopeDocument envelopeDocument
Process	It retrieves Header from the EnvelopeDocument and retrieves GroupExpiryTime element from the header. If GroupExpiryTime element is null then returns null else returns string associated with GroupExpiryTime element.
Output	String
Exceptions	WsFaultException
Comments	-

public String getGroupMaxIdleDuration(EnvelopeDocument envelopeDocument) throws WsFaultException

Description	Retrieves GroupMaxIdleDuration Element from the Header of the EnvelopeDocument
Input arguments	EnvelopeDocument envelopeDocument
Process	It retrieves Header from the EnvelopeDocument and retrieves GroupMaxIdleDuration element from the header. If GroupMaxIdleDuration element is null then returns null else returns

	string associated with GroupMaxIdleDuration element.
Output	String
Exceptions	WsFaultException
Comments	-

public String getMExpiryTime(EnvelopeDocument envelopeDocument)throws WsFaultException

Description	Retrieves MessageExpiryTime Element from the Header of the EnvelopeDocument
Input arguments	EnvelopeDocument envelopeDocument
Process	It retrieves Header from the EnvelopeDocument and retrieves MessageExpiryTime element from the header. If MessageExpiryTime element is null then returns null else returns string associated with MessageExpiryTime element.
Output	String
Exceptions	WsFaultException
Comments	-

public String getRPattern(EnvelopeDocument envelopeDocument)throws WsFaultException

Description	Retrieves ReplyPattern Element from the Header of the envelopeDocument
Input arguments	EnvelopeDocument envelopeDocument
Process	It retrieves Header from the EnvelopeDocument and retrieves ReplyPattern element from the header. If ReplyPattern element is null then returns null else returns string associated with ReplyPattern element.
Output	String
Exceptions	WsFaultException
Comments	-

public String getLastMessage(EnvelopeDocument envelopeDocument)throws WsFaultException

Description	Retrieves LastMessage Element from the Header of the EnvelopeDocument
Input arguments	EnvelopeDocument envelopeDocument
Process	It retrieves Header from the EnvelopeDocument and retrieves LastMessage element from the header. If LastMessage element is null then returns null else returns string associated with LastMessage element.
Output	String
Exceptions	WsFaultException
Comments	-

public String getReplyTo(EnvelopeDocument envelopeDocument)throws WsFaultException

Description	Retrieves ReplyTo Element from the Header of the EnvelopeDocument
Input arguments	EnvelopeDocument envelopeDocument
Process	It retrieves Header from the EnvelopeDocument and retrieves ReplyTo element from the header. If ReplyTo element is null then returns null else returns string associated with ReplyTo element.
Output	String

Exceptions	WsFaultException
Comments	-

private void checkForElement(XmlCursor xmlCursor, QName qName) throws WsFaultException

Description	check to see if the element with the qName exists and position the cursor at the right location if it does
Input arguments	XmlCursor xmlCursor QName qName
Process	It tries to locate the qName. If qName does not exist then throws WsFaultException by calling method throwInvalidMessageFault().
Output	-
Exceptions	WsFaultException
Comments	-

public boolean hasMessageIDElement(RequestDocument requestDocument)

Description	Checks whether MessageID element exists or not
Input arguments	RequestDocument requestDocument
Process	It tries to locate the MesageID element within RequestDocument. If it exists then returns true else returns false.
Output	boolean
Exceptions	-
Comments	-

public boolean hasGroupId(RequestDocument requestDocument)

Description	Checks whether GroupID element exists or not
Input arguments	RequestDocument requestDocument
Process	It tries to locate the GroupID element within RequestDocument. If it exists then returns true else returns false.
Output	boolean
Exceptions	-
Comments	-

public boolean hasMessageExpiryTimeElement(RequestDocument requestDocument)

Description	Checks whether MessageExpiryTime element exists or not
Input arguments	RequestDocument requestDocument
Process	It tries to locate the MessageExpiryTime element within RequestDocument. If it exists then returns true else returns false.
Output	boolean
Exceptions	-
Comments	-

public boolean hasReplyPatternElement(RequestDocument requestDocument)

Description	Checks whether ReplyPattern element exists or not
Input arguments	RequestDocument requestDocument
Process	It tries to locate the ReplyPattern element within RequestDocument. If it exists then returns true else returns false.
Output	boolean
Exceptions	-
Comments	-

public boolean hasRefToMessageIdsElement(PollRequestDocument pollRequestDocument)

Description	Checks whether RefToMessageIds element exists or not
Input arguments	PollRequestDocument pollRequestDocument
Process	It tries to locate the RefToMessageIds element within PollRequestDocument. If it exists then returns true else returns false.
Output	boolean
Exceptions	-
Comments	-

private BodyType getEnvelopeBody(EnvelopeDocument envelopeDocument)throws WsFaultException

Description	Returns Envelope Body if exists
Input arguments	EnvelopeDocument envelopeDocument
Process	Returns Envelope Body if exists else throws WsFaultException by calling method throwInvalidMessageFault() method.
Output	BodyType
Exceptions	WsFaultException
Comments	-

private HeaderType getEnvelopeHeader(EnvelopeDocument envelopeDocument)throws WsFaultException

Description	Returns Envelope Header if exists
Input arguments	EnvelopeDocument envelopeDocument
Process	Returns Envelope Header if exists else throws WsFaultException by calling method throwInvalidMessageFault() method.
Output	HeaderType
Exceptions	WsFaultException
Comments	-

private void throwInvalidMessageFault(String reason)throws WsFaultException

Description	Throw a WsFaultException, initialized with the invalid message QName and the specified reason
Input arguments	String reason
Process	Creates an instance of WsFaultException by invalidMessageQName and reason parameters and throws WsFaultException
Output	-
Exceptions	WsFaultException
Comments	-

Class: public class WsrRequestCreatorImpl implements WsrRequestCreator

public static WsrRequestCreator getInstance()

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

public EnvelopeDocument getPollRequest(ToDocument to, EndpointReferenceType wsrSourceEpr, Hashtable groupIdAndRangesTable)

Description	Create an EnvelopeDocument containing PollRequest Document
-------------	--

	based on the specified parameters.
Input arguments	ToDocument to, EndpointReferenceType wsrSourceEpr, Hashtable groupIdAndRangesTable
Process	Creates EnvelopeDocument from specified to and wsrSourceEpr parameters. Creates PollRequestDocument by calling newPollRequest(groupIdAndRangesTable, wsrSourceEpr) method of WsrElementCreation class. Adds pollRequestDocument to EnvelopeDocument and returns EnvelopeDocument.
Output	EnvelopeDocument
Exceptions	-
Comments	-

```
public void addRequest(EnvelopeDocument envelopeDocument, RequestDocument requestDocument)
```

Description	Adds requestDocument to the EnvelopeDocument header.
Input arguments	EnvelopeDocument envelopeDocument, RequestDocument requestDocument
Process	Adds requestDocument to the EnvelopeDocument header by calling addToSoapHeader(EnvelopeDocument, requestDocument) of SoapMessageAlteration class.
Output	-
Exceptions	-
Comments	-

Class: public class WsrResponseCreatorImpl implements WsrResponseCreator

```
public static WsrResponseCreator getInstance()
```

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

```
public void addResponse(EnvelopeDocument envelopeDocument, ResponseDocument responseDocument)
```

Description	Adds responseDocument to the EnvelopeDocument header.
Input arguments	EnvelopeDocument envelopeDocument, ResponseDocument responseDocument
Process	Adds responseDocument to the EnvelopeDocument header by calling addToSoapHeaderAsLastChild (EnvelopeDocument, responseDocument) of SoapMessageAlteration class.
Output	-
Exceptions	-
Comments	-

```
public EnvelopeDocument getResponse(EndpointReferenceType wsrSink, EndpointReferenceType wsrSourceEpr)
```

Description	Create an EnvelopeDocument containing Response Document based on the specified parameters.
-------------	--

Input arguments	EndpointReferenceType wsrSinkEpr, EndpointReferenceType wsrSourceEpr
Process	Creates EnvelopeDocument from specified wsrSinkEpr and wsrSourceEpr parameters. Creates Empty ResponseDocument by calling newResponse() method of WsrElementCreation class. Adds ResponseDocument to EnvelopeDocument header by calling addToSoapHeaderAsLastChild (EnvelopeDocument, responseDocument) of SoapMessageAlteration class and returns EnvelopeDocument.
Output	EnvelopeDocument
Exceptions	-
Comments	-

Class: public class WsrSinkGroupTerminationImpl

public static WsrSinkGroupTerminationImpl getInstance()

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

public void processGroupTermination(String groupId)throws WsrStorageException

Description	Checks to see if the group corresponding to groupId needs to be terminated, and if so, proceed to do that
Input arguments	String groupId
Process	It retrieves Group Info and Group Termination Info for groupId. Tests groupExpiryTime, groupMaxIdleDuration and messageExpiryTime parameters for Group Closing. If group is closed then proceed to remove the group by testing messageExpiryTime of each messages of the group. If one of the message is expired then group is removed and all the group info, group termination info and messages info is deleted from the database. It throws WsrStorageException if error occurs during database operations.
Output	-
Exceptions	WsrStorageException
Comments	-

Class: public class WsrSinkNode extends WsProcessor implements WsrSink

public void setMessageFlow(WsMessageFlow wsMessageFlow)throws DeploymentException

Description	Sets the message flow which the processor should use
Input arguments	WsMessageFlow wsMessageFlow
Process	Sets the message flow which the processor should use. Also sets the message flow for sink node helper class.
Output	-
Exceptions	DeploymentException
Comments	-

public WsMessageFlow getMessageFlow()

Description	Gets the message flow which the processor should use.
-------------	---

Input arguments	-
Process	Returns wsMessageFlow
Output	WsMessageFlow
Exceptions	-
Comments	-

public void setEndpointReference(EndpointReferenceType endpointReference)

Description	Sets the endpoint reference associated with this node
Input arguments	EndpointReferenceType endpointReference
Process	Sets the sinkEpr with endpointReference and also calls the sinkNodeHelper.setEndpointReference(sinkEpr)
Output	-
Exceptions	-
Comments	-

public EndpointReferenceType getEndpointReference()

Description	gets the endpoint reference associated with this node
Input arguments	-
Process	Returns sinkEpr
Output	EndpointReferenceType
Exceptions	-
Comments	-

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 envelopeDocument, int direction
Process	Creates wsrExchangeInfo from envelopeDocument. Retrieves addressingHeaders from wsrExchangeInfo. Determines the exchange type by calling checkExchangeType (wsrExchangeInfo, direction) method of SinkNodeHelper class. If direction = wsMessageFlow.FROM_APPLICATION then calls processMessageFromApplication(envelopeDocument, addressingHeaders) method. If direction = wsMessageFlow.FROM_NETWORK and wsExchangeInfo = isProcessRequest & hasRequest then calls processRequest(envelopeDocument, addressingHeaders) method and if direction = wsMessageFlow.FROM_NETWORK and wsExchangeInfo = isProcessPollRequest and hasPollRequest then calls processPollRequest(envelopeDocument, addressingHeaders) method. If execution is successful then returns Boolean = true else returns false.
Output	boolean
Exceptions	UnknownExchangeException IncorrectExchangeException processingException MessageFlowException

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

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

Description	Process the Request.
Input arguments	EnvelopeDocument envelopeDocument, AddressingHeaders addressingHeaders
Process	Retrieves RequestDocuement from the EnvelopeDocument. Checks the problems in RequestDocument by calling checkForProblemsInRequestDocument(RequestDocument, AddressingHeaders) method of SinkNodeHelper class. If no errors in RequestDocument, then retrieves groupId from RequestDocument and retrieves Group Info for groupId. If group is already exists in database then calls processExistingGroup(groupId, EnvelopeDocument, AddressingHeaders) method. If group not exists then calls processNewGroup(groupId, EnvelopeDocument, AddressingHeaders) method.
Output	-
Exceptions	WsFaultException, MessageFlowException, WsrStorageException
Comments	-

public void processNewGroup(String groupId, EnvelopeDocument envelopeDocument, AddressingHeaders addressingHeaders) throws WsFaultException, WsrStorageException, MessageFlowException

Description	Process New Group.
Input arguments	String groupId, EnvelopeDocument envelopeDocument, AddressingHeaders addressingHeaders
Process	It retrieves the RequestDocument from the EnvelopeDocument. If RequestDocument follows all the reliability rules messageOrdering, message Duplication Elimination and also groupExpiryTime, messageExpiryTime and groupMaxIdleDuration is not passed then creates groupInfo, groupterminationInfo and messageInfo and adds it to database. Sink node also sends response(acknowledgement or fault) back to source node.
Output	-
Exceptions	WsFaultException, MessageFlowException, WsrStorageException
Comments	-

public void processExistingGroup(String groupId, EnvelopeDocument envelopeDocument, AddressingHeaders addressingHeaders) throws WsFaultException, WsrStorageException, MessageFlowException

Description	Process Existing Group.
Input arguments	String groupId, EnvelopeDocument envelopeDocument,

	AddressingHeaders addressingHeaders
Process	It retrieves the groupInfo, groupTerminationInfo for groupId from the database. If group is not expired and RequestDocument from EnvelopeDocument satisfies all the reliability parameters then updates the groupInfo, groupTerminationInfo in database. Adds messageInfo in the database. Sink node also sends response(acknowledgement or fault) back to source node.
Output	-
Exceptions	WsFaultException, MessageFlowException, WsrStorageException
Comments	-

public void processUndeliveredSequenceNumbers(String groupId, long sequenceNumber, AddressingHeaders addressingHeaders, RequestDocument requestDocument) throws WsrStorageException, MessageFlowException, WsFaultException

Description	Process undelivered SequenceNumbers.
Input arguments	String groupId, long sequenceNumber, AddressingHeaders addressingHeaders, RequestDocument requestDocument
Process	It retrieves all the undelivered sequence numbers from the database. Retrieves message info for those sequence numbers. If message is not expired then delivered all the undelivered messages in order and sends response back to source.
Output	-
Exceptions	WsFaultException, MessageFlowException, WsrStorageException
Comments	-

public EnvelopeDocument createResponseEnvelope(RequestDocument requestDocument, AddressingHeaders addressingHeaders, String groupId, String fault, boolean isNonSequenceReply, long sequenceNumber) throws MessageFlowException

Description	Creates EnvelopeDocument containing Response Document.
Input arguments	RequestDocument requestDocument, AddressingHeaders addressingHeaders, String groupId, String fault, boolean isNonSequenceReply, long sequenceNumber
Process	Creates EnvelopeDocument containing Response Document by using specified parameters and by calling newResponse() method of WsrElementCreation class.
Output	-
Exceptions	MessageFlowException
Comments	-

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

Description	Process Payload coming from Application
Input arguments	EnvelopeDocument envelopeDocument, AddressingHeaders addressingHeaders
Process	Simply enrout an EnvelopeDocument to network
Output	-
Exceptions	WsFaultException, MessageFlowException, WsrStorageException
Comments	-

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

Description	Process PollRequest Document
Input arguments	EnvelopeDocument envelopeDocument, AddressingHeaders addressingHeaders
Process	Retrieves the PollRequestDocument from EnvelopeDocument. Checks for errors. If errors not exist then retrieves all the groupIds and its associated sequence numbers from the PollRequest Document. For all the sequence numbers retrieves the message status from the database. Respond source with Response Document which contains message status for all the sequence numbers.
Output	-
Exceptions	WsFaultException, MessageFlowException, WsrStorageException
Comments	-

public boolean addStatusToResponse(ResponseDocument responseDocument, String groupId, long[] seqNums) throws WsrStorageException

Description	Adds status of the message to ResponseDocument
Input arguments	ResponseDocument responseDocument String groupId, Long[] seqNums
Process	Adds message status (Acknowledgement or Fault) to the Response Document.
Output	boolean
Exceptions	WsrStorageException
Comments	-

public long[] createLongArrayFromVector(Vector vector)

Description	creates long Number array from Vector
Input arguments	Vector vector
Process	creates long Number array from Vector
Output	long[]
Exceptions	-
Comments	-

Class: public class WsrSinkNodeHelper extends WsProcessor
public static WsrSinkNodeHelper getInstance()

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

public void setMessageFlow(WsMessageFlow wsMessageFlow) throws DeploymentException

Description	Sets the message flow which the processor should use
Input arguments	WsMessageFlow wsMessageFlow
Process	Sets the message flow which the processor should use.
Output	-
Exceptions	DeploymentException
Comments	-

public WsMessageFlow getMessageFlow()

Description	Gets the message flow which the processor should use.
Input arguments	-
Process	Returns wsMessageFlow
Output	WsMessageFlow
Exceptions	-
Comments	-

public void setEndpointReference(EndpointReferenceType endpointReference)

Description	Set the endpoint reference associated with this node
Input arguments	EndpointReferenceType endpointReference
Process	Sets the sinkEpr with endpointReference.
Output	-
Exceptions	-
Comments	-

public boolean checkForProblemsInRequestDocument(RequestDocument requestDocument, AddressingHeaders addressingHeaders) throws WsFaultException, WsrStorageException, MessageFlowException

Description	Checks the RequestDocument for any problems or errors
Input arguments	RequestDocument requestDocument, AddressingHeaders addressingHeaders
Process	Calls checkForInvalidRequest(requestDocument, addressingHeaders) method and returns true if any errors in Request Document
Output	Boolean
Exceptions	WsFaultException, WsrStorageException, MessageFlowException
Comments	-

private boolean checkForInvalidRequest(RequestDocument requestDocument, AddressingHeaders addressingHeaders) throws WsFaultException, MessageFlowException, WsrStorageException

Description	Checks the RequestDocument for any problems or errors
Input arguments	RequestDocument requestDocument,

	AddressingHeaders addressingHeaders
Process	It checks the Request Document for mandatory parameters (Message Expiry Time, Reply Pattern). If mandatory parameters are not present then calls method processRMFault(requestDocument,addressingHeaders,fault,errorReport) method with fault "Invalid Request". Finally returns true if errors else return false.
Output	Boolean
Exceptions	WsFaultException, WsrStorageException, MessageFlowException
Comments	-

private boolean checkForInvalidMessageId(RequestDocument requestDocument, AddressingHeaders addressingHeaders) throws WsFaultException, WsrStorageException, MessageFlowException

Description	Checks the MessageId element for any problems or errors
Input arguments	RequestDocument requestDocument, AddressingHeaders addressingHeaders
Process	It retrieves the MessageId element from the Request Document and checks for mandatory parameters (groupId, sequenceNumber). If mandatory parameters are not present then calls method throwInvalidMessageFaultException (errorReport , addressingHeaders) method. Finally returns true if errors else return false.
Output	boolean
Exceptions	WsFaultException, WsrStorageException, MessageFlowException
Comments	-

private boolean checkForInvalidMessageParameters(RequestDocument requestDocument, AddressingHeaders addressingHeaders) throws WsrStorageException, MessageFlowException

Description	Checks the MessageId parameters for any errors.
Input arguments	RequestDocument requestDocument, AddressingHeaders addressingHeaders
Process	It retrieves the MessageId element from the Request Document and checks for errors. E.g. If GroupExpiryTime and GroupMaxIdleDuration both present or groupExpiryTime is less than messageExpiryTime its an error. Calls the processRMFault(requestDocument, addressingHeaders, fault, errorReport) method with fault = "Invalid Message Parameters". Finally returns true if errors else return false.
Output	boolean
Exceptions	WsrStorageException, MessageFlowException
Comments	-

private boolean checkForInvalidReplyPattern(RequestDocument requestDocument, AddressingHeaders addressingHeaders) throws MessageFlowException, WsrStorageException

Description	Checks the ReplyPattern element of Request Document.
Input arguments	RequestDocument requestDocument, AddressingHeaders addressingHeaders
Process	It retrieves the ReplyPattern element from the Request Document and checks for errors. E.g. If ReplyPattern is "Callback" and replyTo element is not set then its an error. Calls the processRMFault(requestDocument, addressingHeaders, fault, errorReport) method with fault = "Invalid Reply Pattern". Finally returns true if errors else return false.
Output	boolean
Exceptions	WsrStorageException, MessageFlowException
Comments	-

```
private void processRMFault(RequestDocument requestDocument, AddressingHeaders
addressingHeaders, QName fault, String reason) throws WsrStorageException,
MessageFlowException
```

Description	Process the Fault.
Input arguments	RequestDocument requestDocument, AddressingHeaders addressingHeaders QName fault, String reason
Process	It creates Response Document with RM fault set for particular message. It also creates Envelope Document with SOAP fault in body and finally sends it back to Source.
Output	-
Exceptions	WsrStorageException, MessageFlowException
Comments	-

```
public EnvelopeDocument createResponse(AddressingHeaders addressingHeaders,
RequestDocument requestDocument, ResponseDocument responseDocument)
```

Description	Creates response based on reply pattern.
Input arguments	RequestDocument requestDocument, AddressingHeaders addressingHeaders ResponseDocument responseDocument
Process	It creates new EnvelopeDocument based on addressing headers. If reply pattern is "Response" then it adds Request and Response header both in the same EnvelopeDocument. If reply pattern is "Callback" then adds only Response header to EnvelopeDocument. Finally returns EnvelopeDocument.
Output	EnvelopeDocument
Exceptions	-
Comments	-

```
public EnvelopeDocument getSoapEnvelope(AddressingHeaders addressingHeaders, String
replyTo)
```

Description	Creates EnvelopeDocument based on specified parameters.
Input arguments	AddressingHeaders addressingHeaders, String replyTo

Process	Creates new EnvelopeDocument by calling createSoapEnvelope(to,from,action) of WsaEnvelopeCreator class. to address can be replyTo in case of "Callback" reply pattern or else it will be source address. Action attribute will be "process Response" and from address will be address of sink. Finally returns EnvelopeDocument.
Output	EnvelopeDocument
Exceptions	-
Comments	-

private void addFaultToSoapBody(EnvelopeDocument envelopeDocument, String reason)

Description	Adds Soap Fault document in the body of the EnvelopeDocument.
Input arguments	EnvelopeDocument envelopeDocument, String reason
Process	Creates Soap fault document by calling getFaultDocument(wsrFaultsImpl.getSoapSenderQName(), reason) method of FaultCreator class. Adds soap fault document in the body of the envelopeDocument by calling copyFaultIntoSOAPBody(envelopeDocument,faultDocument) method of the FaultCreator class. Finally returns the EnvelopeDocument.
Output	EnvelopeDocument
Exceptions	-
Comments	-

public WsrGroupInfoImpl createGroupInfoFromRequestDocument(RequestDocument requestDocument, AddressingHeaders addressingHeaders)

Description	create GroupInfo from Request Document
Input arguments	RequestDocument requestDocument, AddressingHeaders addressingHeaders
Process	Creates WsrGroupInfoImpl object by retrieving all the necessary parameters from the requestDocument and addressingHeaders. And returns WsrGroupInfoImpl object.
Output	WsrGroupInfoImpl
Exceptions	-
Comments	-

public WsrStorageWidgetImpl createMessageInfoFromRequestDocument(RequestDocument requestDocument)

Description	create MessageInfo from Request Document
Input arguments	RequestDocument requestDocument
Process	Creates WsrStorageWidgetImpl object by retrieving all the necessary parameters from the requestDocument and returns WsrStorageWidgetImpl object.
Output	WsrStorageWidgetImpl
Exceptions	-
Comments	-

public void updateGroupTerminationInfo(WsrGroupTerminationInfoImpl wsrGroupTerminationInfo, Calendar groupExpiryTime, GDuration groupMaxIdleDuration, Calendar messageExpiryTime)

Description	update groupTerminationInfo object according to specified parameters
Input arguments	WsrGroupTerminationInfoImpl wsrGroupTerminationInfo, Calendar groupExpiryTime, GDuration groupMaxIdleDuration, Calendar messageExpiryTime
Process	Updates the WsrGroupTerminationInfoImpl object if groupExpiryTime, groupMaxIdleDuration or messageExpiryTime parameter is expired.
Output	-
Exceptions	-
Comments	-

public void checkExchangeType(WsrExchangeInfo wsrExchangeInfo, int direction) throws UnknownExchangeException, IncorrectExchangeException

Description	Checks an exchange type for validity. If there are problems exceptions are thrown.
Input arguments	WsrExchangeInfo wsrExchangeInfo, int direction
Process	If direction == WsMessageFlow.FROM_NETWORK calls checkExchangeFromNetwork(wsrExchangeInfo) method if direction == WsMessageFlow.FROM_APPLICATION) calls checkExchangeFromApplication(wsrExchangeInfo) method
Output	-
Exceptions	UnknownExchangeException, IncorrectExchangeException
Comments	-

private void checkExchangeFromNetwork(WsrExchangeInfo wsrExchangeInfo) throws UnknownExchangeException, IncorrectExchangeException

Description	Checks the exchange coming from the network for validity.
Input arguments	WsrExchangeInfo wsrExchangeInfo,
Process	Checks the exchange coming from network. Exchanges containing ProcessRequestAction, ProcessPollRequestAction, Request & PollRequest elements should be accepted, else throws IncorrectExchangeException error.
Output	-
Exceptions	UnknownExchangeException, IncorrectExchangeException
Comments	-

private void checkExchangeFromApplication(WsrExchangeInfo wsrExchangeInfo) throws UnknownExchangeException, IncorrectExchangeException

Description	Checks the exchange coming from the application for validity.
Input arguments	WsrExchangeInfo wsrExchangeInfo,
Process	Checks the exchange coming from application. Exchange containing ProcessResponseAction is valid else throws IncorrectExchangeException error.
Output	-
Exceptions	UnknownExchangeException, IncorrectExchangeException

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

public void throwInvalidMessageFaultException(String reason, AddressingHeaders addressingHeaders) throws WsFaultException

Description	Throws an Invalid MessageFault exception based on the specified parameters
Input arguments	String reason, AddressingHeaders addressingHeaders
Process	Creates WsFaultException and throws it.
Output	-
Exceptions	WsFaultException
Comments	-

public void checkForProblemsInPollRequestDocument(PollRequestDocument pollRequestDocument, AddressingHeaders addressingHeaders) throws WsFaultException

Description	Checks problems in Poll Request Document.
Input arguments	PollRequestDocument pollRequestDocument, AddressingHeaders addressingHeaders
Process	Checks Poll Request Document for any errors(E.g there should be atleast one RefToMessageIDs element present. groupId should not be null) etc. if any error occurs calls throwInvalidMessageFaultException(reason, addressingHeaders) method.
Output	-
Exceptions	WsFaultException
Comments	-

Class: public class WsrSinkGroupMonitorFactoryImpl implements WsrSinkGroupMonitorFactory
 public static WsrSinkGroupMonitorFactory getInstance()

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

public WsrSinkGroupMonitor getWsrSinkGroupMonitor(String configInfo, WsMessageFlow wsMessageFlow) throws WsrStorageException, DeploymentException

Description	Gets a Sink Group monitor for the specified configuration file.
Input arguments	String configInfo, WsMessageFlow wsMessageFlow
Process	If configInfo or wsrMessageFlow is null It throws a Deployment Exception. If SinkGroupMonitor for given configuration file exists in Hashtable of group monitors, it will return that instance, else will create new Sink Group Monitor instance and adds that to Hashtable.
Output	WsrSinkGroupMonitor
Exceptions	WsrStorageException, DeploymentException
Comments	-

Class: public class WsrSinkGroupMonitorImpl extends Thread implements WsrSinkGroupMonitor
public void startServices()

Description	Begin services related to Group monitoring viz. termination of groups
Input arguments	-
Process	Starts the Sink group monitor thread by calling start() method
Output	-
Exceptions	-
Comments	-

public void stopServices()

Description	Stop services related to group monitoring.
Input arguments	-
Process	Stops the Sink group monitor threads and all its monitoring activities.
Output	-
Exceptions	-
Comments	-

public void run()

Description	Runs the Group Monitoring activities.
Input arguments	-
Process	Calls cycleThroughGroups() method
Output	-
Exceptions	-
Comments	-

private void cycleThroughGroups() throws MessageFlowException, WsrStorageException

Description	Process Group termination of received groups.
Input arguments	-
Process	Retrieves all the received groups from the database and process group termination for each group by calling processGroupTermination(groupId) of wsrSinkGroupTerminationImpl class.
Output	-
Exceptions	MessageFlowException, WsrStorageException
Comments	-

Class: public class WsrSourceGroupMonitorFactoryImpl implements WsrSourceGroupMonitorFactory
public static WsrSourceGroupMonitorFactory getInstance()

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

`public WsrSourceGroupMonitor getWsrSourceGroupMonitor(String configInfo, WsMessageFlow wsMessageFlow) throws WsrStorageException, DeploymentException`

Description	Gets a Source Group monitor for the specified configuration file.
Input arguments	String configInfo, WsMessageFlow wsMessageFlow
Process	If configInfo or wsMessageFlow is null It throws a Deployment Exception. If SourceGroupMonitor for given configuration file exists in Hashtable of group monitors, it will return that instance, else will create new Source Group Monitor instance and adds that to Hashtable.
Output	WsrSourceGroupMonitor
Exceptions	WsrStorageException, DeploymentException
Comments	-

Class: `public class WsrSourceGroupMonitorImpl extends Thread implements WsrSourceGroupMonitor`
`public void startServices()`

Description	Begin services related to Group monitoring viz. retransmission of messages and termination of groups
Input arguments	-
Process	Starts the Source group monitor thread by calling start() method
Output	-
Exceptions	-
Comments	-

`public void stopServices()`

Description	Stop services related to group monitoring.
Input arguments	-
Process	Stops the Source group monitor threads and all its monitoring activities.
Output	-
Exceptions	-
Comments	-

`public void run()`

Description	Runs the Group Monitoring activities.
Input arguments	-
Process	Calls cycleThroughGroups() method
Output	-
Exceptions	-
Comments	-

`private void cycleThroughGroups() throws MessageFlowException, WsrStorageException`

Description	Process retransmission and termination of sent groups.
Input arguments	-
Process	Retrieves all the sent groups from the database. Checks each group's messages for the retransmission. If retransmission requires then calls checkToIssueRetransmissions(WsrGroupInfoImpl) and finally executes termination algorithm for each group by calling processGroupTermination(groupId) method of

	WsrSourceGroupTerminationImpl class.
Output	-
Exceptions	MessageFlowException, WsrStorageException
Comments	-

public void checkToIssueRetransmissions(WsrGroupInfoImpl wsrGroupInfoImpl) throws WsrStorageException, MessageFlowException

Description	This method checks to see if retransmissions need to be issued on the specified group. If a need arises retransmissions are issued and the retransmission interval is reset.
Input arguments	WsrGroupInfoImpl wsrGroupInfoImpl
Process	This method checks to see if retransmissions need to be issued on the Specified group. If a need arises retransmissions are issued and the retransmission interval is reset.
Output	-
Exceptions	MessageFlowException, WsrStorageException
Comments	-

Class: public class WsrSourceGroupTerminationImpl

public static WsrSourceGroupTerminationImpl getInstance()

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

public void processGroupTermination(String groupId) throws WsrStorageException

Description	Checks to see if the group corresponding to groupId needs to be terminated, and if so, proceed to do that
Input arguments	String groupId
Process	It retrieves Group Info and Group Termination Info for groupId. Tests groupExpiryTime, groupMaxIdleDuration and messageExpiryTime parameters for Group Closing. If group is closed then proceed to remove the group by retrieving unacknowledged or unfaulted sequence numbers of the group. If all the messages of the group is acknowledged or faulted then group is removed and all the group info, group termination info and messages info is deleted from the database. It throws WsrStorageException if error occurs during database operations.
Output	-
Exceptions	WsrStorageException
Comments	-

Class: public class WsrSourceNode extends WsProcessor implements WsrSource

public void setMessageFlow(WsMessageFlow wsMessageFlow) throws DeploymentException

Description	Sets the message flow which the processor should use
Input arguments	WsMessageFlow wsMessageFlow

Process	Sets the message flow which the processor should use. Also sets the message flow for source node helper class.
Output	-
Exceptions	DeploymentException
Comments	-

`public WsMessageFlow getMessageFlow()`

Description	Gets the message flow which the processor should use.
Input arguments	-
Process	Returns wsMessageFlow
Output	WsMessageFlow
Exceptions	-
Comments	-

`public void setEndpointReference(EndpointReferenceType endpointReference)`

Description	Sets the endpoint reference associated with this node
Input arguments	EndpointReferenceType endpointReference
Process	Sets the sourceEpr with endpointReference and also calls the sourceNodeHelper.setEndpointReference(sourceEpr)
Output	-
Exceptions	-
Comments	-

`public EndpointReferenceType getEndpointReference()`

Description	gets the endpoint reference associated with this node
Input arguments	-
Process	Returns sourceEpr
Output	EndpointReferenceType
Exceptions	-
Comments	-

`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 envelopeDocument, int direction
Process	Creates wsrExchangeInfo from envelopeDocument. Retrieves addressingHeaders from wsrExchangeInfo. Determines the exchange type by calling checkExchangeType (wsrExchangeInfo, direction) method of SourceNodeHelper class. If direction = wsMessageFlow.FROM_APPLICATION and wsExchangeInfo = isProcessRequest then calls processMessageFromApplication(envelopeDocument, addressingHeaders) method. if direction = wsMessageFlow.FROM_NETWORK and wsExchangeInfo = hasResponse then calls processResponse(envelopeDocument, addressingHeaders) method. if direction = wsMessageFlow.FROM_NETWORK and

	wsExchangeInfo is not hasResponse then calls enroutetoApplication(envelopeDocument) If execution is successful then returns Boolean = true else returns false.
Output	boolean
Exceptions	UnknownExchangeException IncorrectExchangeException processingException MessageFlowException
Comments	-

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

Description	Process the Message receiving from application.
Input arguments	EnvelopeDocument envelopeDocument, AddressingHeaders addressingHeaders
Process	Checks the addressingHeaders for any errors, If no errors then retrieves the new groupId. If this group is already exist in database then calls processExistingGroup(groupId,envelopeDocument) method and if group does not exist then calls processNewGroup(groupId, destination, envelopeDocument) method.
Output	-
Exceptions	WsFaultException, MessageFlowException, WsrStorageException
Comments	-

public void removeHeadersFromEnvelopeDocument(EnvelopeDocument envelopeDocument)

Description	Removes the extra headers from EnvelopeDocument.
Input arguments	EnvelopeDocument envelopeDocument
Process	Removes all the extra headers non relevant to Reliability header inserted by Source application for convenience.
Output	-
Exceptions	-
Comments	-

public void createAndEnroutePollRequest(ToDocument toDocument) throws WsrStorageException, MessageFlowException

Description	Creates and enroutes Poll Request Document.
Input arguments	ToDocument toDocument
Process	Retrieves all the groupIds for given destination. It also retrieves all the unacknowledged sequence numbers for each group and creates ranges for sequenceNumbers of each group in Hashtable(groupIdAndRangesTable). And finally creates envelope document by calling getPollRequest (toDocument, sourceEpr,groupIdAndRangesTable) method of wsrRequestCreator class.
Output	-

Exceptions	WsrStorageException, MessageFlowException
Comments	-

public boolean processNewGroup(String groupId, String destination, EnvelopeDocument envelopeDocument) throws WsrStorageException, WsFaultException

Description	Process New Group.
Input arguments	String groupId, String destination, EnvelopeDocument envelopeDocument
Process	Creates groupInfo, groupTerminationInfo and messageInfo. Creates RequestDocument by calling createRequest(groupId, groupExpiryTime, groupMaxIdleDuration, sequenceNumber, hasLastSequenceNumber, messageExpiryTime, replyPattern, replyTo) of sourceNodeHelper class. Creates EnvelopeDocument containing RequestDocument by calling addRequest(envelopeDocument, requestDocument) method of wsrElementAddition class. Stores groupinfo, groupterminationinfo and messageinfo to the database. Returns true if operation is successful else return false.
Output	boolean
Exceptions	WsrStorageException, WsFaultException
Comments	-

public boolean processExistingGroup(String groupId, EnvelopeDocument envelopeDocument) throws WsrStorageException, WsFaultException

Description	Process Existing Group.
Input arguments	String groupId, EnvelopeDocument envelopeDocument
Process	Retrieves groupInfo, groupTerminationInfo for given groupId. Creates messageInfo. Checks for group closing. If group is not closed creates RequestDocument by calling createRequest(groupId, groupExpiryTime, groupMaxIdleDuration, sequenceNumber, hasLastSequenceNumber, messageExpiryTime, replyPattern, replyTo) of sourceNodeHelper class. Creates EnvelopeDocument containing RequestDocument by calling addRequest(envelopeDocument, requestDocument) method of wsrElementAddition class. Stores groupinfo, groupterminationinfo and messageinfo to the database. Returns true if operation successful else return false.
Output	boolean
Exceptions	WsrStorageException, WsFaultException
Comments	-

public WsrGroupInfoImpl createGroupInfo(String groupId, EnvelopeDocument envelopeDocument) throws WsFaultException

Description	Creates Group Info from specified parameters.
-------------	---

Input arguments	String groupId, EnvelopeDocument envelopeDocument
Process	Creates WsrGroupInfoImpl object and returns it.
Output	WsrGroupInfoImpl
Exceptions	WsFaultException
Comments	-

public WsrStorageWidgetImpl createMessageInfo(String groupId, long sequenceNumber, EnvelopeDocument envelopeDocument) throws WsFaultException

Description	CreatesMessage Info from specified parameters.
Input arguments	String groupId, long sequenceNumber, EnvelopeDocument envelopeDocument
Process	Creates WsrStorageWidgetImpl object and returns it.
Output	WsrStorageWidgetImpl
Exceptions	WsFaultException
Comments	-

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

Description	Process Response received over the network.
Input arguments	EnvelopeDocument envelopeDocument, AddressingHeaders addressingHeaders
Process	Process the Non Sequence Replies by calling processNonSequenceReply(responseDocument) method. Process Sequence Replies by calling processSequenceReply(responseDocument) method and finally process group termination by calling processTermination(responseDocument) method.
Output	-
Exceptions	WsFaultException, MessageFlowException, WsrStorageException
Comments	-

public boolean processNonSequenceReply(ResponseDocument responseDocument) throws WsrStorageException, MessageFlowException

Description	Process Non Sequence Reply.
Input arguments	ResponseDocument responseDocument
Process	Retrieves Non Sequence Reply Acknowledge Vector and process it by calling processNonSequenceReplyAcknowledgement(nonSequenceReplyAcknowledgementVector) method of Source Node Helper class. Retrieves Non Sequence Reply retransmission vector and process it by calling processNonSequenceReplyMessageProcessingFailureFault(retransmissionVector) method of Source Node Helper class. Retrieves Non Sequence Reply Fault Hashtable and process it by calling processNonSequenceReplyFault(nonSequenceReplyFault) method of Source Node Helper class. If all operations successful returns true else return false.

Output	boolean
Exceptions	WsrStorageException, MessageFlowException
Comments	-

public boolean processSequenceReply(ResponseDocument responseDocument) throws WsrStorageException, MessageFlowException

Description	Process Sequence Reply.
Input arguments	ResponseDocument responseDocument
Process	Retrieves Sequence Reply Acknowledge Hashtable and process it by calling processSequenceReplyAcknowledgement(sequenceRepliesAcknowledgement) method of Source Node Helper class. Retrieves Sequence Reply retransmission Hashtable and process it by calling processSequenceReplyMessageProcessingFailureFault(retransmissionTable) method of Source Node Helper class. Retrieves Sequence Reply Fault Hashtable and process it by calling processSequenceReplyFault(sequenceReplyFaultTable) method of Source Node Helper class. If all operations successful returns true else return false.
Output	boolean
Exceptions	WsrStorageException, MessageFlowException
Comments	-

public void processTermination(ResponseDocument responseDocument) throws WsrStorageException

Description	Process Group Termination.
Input arguments	ResponseDocument responseDocument
Process	Retrieves all GroupIds from Response Document and executes group termination algorithm by calling processGroupTermination(groupId) of WsrSourceGroupTerminationImpl class.
Output	-
Exceptions	WsrStorageException
Comments	-

private String retrieveGroupId(String destinationAddress) throws WsrStorageException, WsFaultException

Description	Retrieves groupId associated with destinationAddress.
Input arguments	String destinationAddress
Process	Retrieves groupId associated with destinationAddress.
Output	-
Exceptions	WsrStorageException WsFaultException
Comments	-

Class: public class WsrSourceNodeHelper extends WsProcessor

public static WsrSourceNodeHelper getInstance()

Description	This will return instance of WsrSourceNodeHelper class
-------------	--

Input arguments	-
Process	-
Output	WsrSourceNodeHelper
Exceptions	-
Comments	-

public void setMessageFlow(WsMessageFlow wsMessageFlow) throws DeploymentException

Description	Sets the message flow which the processor should use
Input arguments	WsMessageFlow wsMessageFlow
Process	Sets the message flow which the processor should use.
Output	-
Exceptions	DeploymentException
Comments	-

public WsMessageFlow getMessageFlow()

Description	Gets the message flow which the processor should use.
Input arguments	-
Process	Returns wsMessageFlow
Output	WsMessageFlow
Exceptions	-
Comments	-

public void setEndpointReference(EndpointReferenceType endpointReference)

Description	Sets the endpoint reference associated with this node
Input arguments	EndpointReferenceType endpointReference
Process	Sets the sourceEpr with endpointReference.
Output	-
Exceptions	-
Comments	-

public void checkExchangeType(WsrExchangeInfo wsrExchangeInfo, int direction) throws UnknownExchangeException, IncorrectExchangeException

Description	Checks an exchange type for validity. If there are problems exceptions are thrown.
Input arguments	WsrExchangeInfo wsrExchangeInfo, int direction
Process	If direction == WsMessageFlow.FROM_NETWORK calls checkExchangeFromNetwork(wsrExchangeInfo) method and if direction == WsMessageFlow.FROM_APPLICATION) calls checkExchangeFromApplication(wsrExchangeInfo) method
Output	-
Exceptions	UnknownExchangeException, IncorrectExchangeException
Comments	-

private void checkExchangeFromNetwork(WsrExchangeInfo wsrExchangeInfo) throws UnknownExchangeException, IncorrectExchangeException

Description	Checks the exchange coming from the network for validity.
Input arguments	WsrExchangeInfo wsrExchangeInfo
Process	Checks the exchange coming from network. Exchanges containing

	hasRequest and hasPollRequest elements are not accepted. If present then throws IncorrectExchangeException error.
Output	-
Exceptions	UnknownExchangeException, IncorrectExchangeException
Comments	-

private void checkExchangeFromApplication(WsrExchangeInfo wsrExchangeInfo) throws UnknownExchangeException, IncorrectExchangeException

Description	Checks the exchange coming from the application for validity.
Input arguments	WsrExchangeInfo wsrExchangeInfo,
Process	Checks the exchange coming from application. Exchange containing ProcessRequest is valid else throws IncorrectExchangeException error.
Output	-
Exceptions	UnknownExchangeException, IncorrectExchangeException
Comments	-

public void checkForProblemsInToDocument(AddressingHeaders addressingHeaders) throws WsFaultException

Description	Checks the problems in To Document.
Input arguments	AddressingHeaders addressingHeaders
Process	This method enforces the rule that the [wsa:To] element should be present within a SOAP header element. If not present an exception needs to be thrown.
Output	-
Exceptions	WsFaultException
Comments	-

public void throwInvalidMessageFaultException(String reason, AddressingHeaders addressingHeaders) throws WsFaultException

Description	Throws an Invalid MessageFault exception based on the specified parameters
Input arguments	String reason, AddressingHeaders addressingHeaders
Process	Creates WsFaultException object and throws it.
Output	-
Exceptions	WsFaultException
Comments	-

public String createGroupId()

Description	Generates GroupId
Input arguments	-
Process	Generates new GroupID by calling getRandomBasedUIDAsString() method of uuidRetrieval class.
Output	String
Exceptions	-
Comments	-

public RequestDocument createRequest(String groupId, Calendar groupExpiryTime, GDuration groupMaxIdleDuration, long sequenceNumber, boolean isLastMessage, Calendar messageExpiryTime, String replyPattern, String replyTo)

Description	Creates Request Document based on specified parameters.
Input arguments	String groupId, Calendar groupExpiryTime, GDuration groupMaxIdleDuration, long sequenceNumber, boolean isLastMessage, Calendar messageExpiryTime, String replyPattern, String replyTo
Process	Creates RequestDocument according to the replyPattern element by calling newRequest() method of WsrElementCreation class. Finally returns the RequestDocument.
Output	RequestDocument
Exceptions	-
Comments	-

public void processNonSequenceReplyAcknowledgement(Vector nonSequenceReplyAcknowledgementVector) throws WsrStorageException

Description	Process Non Sequence Reply Acknowledgements
Input arguments	Vector nonSequenceReplyAcknowledgementVector
Process	Retrieves groupIds from Non Sequence Reply Acknowledgement Vector and calls processAcknowledgements(groupId, sequenceNumber[]) method for each groupId where sequenceNumber for Non Sequence Reply is zero as there is only single message in question.
Output	-
Exceptions	WsrStorageException
Comments	-

public void processSequenceReplyAcknowledgement(Hashtable table) throws WsrStorageException

Description	Process Sequence Reply Acknowledgements
Input arguments	Hashtable table
Process	Retrieves groupIds and its associated acknowledged sequence numbers from Sequence Reply Acknowledgement Hashtable and calls processAcknowledgements(groupId, sequenceNumber[]) method for each groupId.
Output	-
Exceptions	WsrStorageException
Comments	-

public boolean processAcknowledgements(String groupId, long[] acks) throws WsrStorageException

Description	Process Acknowledgements
Input arguments	String groupId, long[] acks
Process	Calls processAcknowledgementsOnGroup(groupId, acks) method of WsrProtocolStorageOperations class. Returns true if operation is successful else returns false.
Output	boolean
Exceptions	WsrStorageException
Comments	-

public void processNonSequenceReplyMessageProcessingFailureFault(Vector retransmissionVector) throws WsrStorageException, MessageFlowException

Description	Process Non Sequence Reply Retransmissions.
Input arguments	Vector retransmissionVector
Process	Retrieves groupIds from retransmission Vector and calls processMessageProcessingFailureFault (groupId, sequenceNumber[]) method for each groupId where sequenceNumber for Non Sequence Reply is zero as there is only single message in question.
Output	-
Exceptions	WsrStorageException, MessageFlowException
Comments	-

public void processSequenceReplyMessageProcessingFailureFault(Hashtable retransmissionTable) throws WsrStorageException, MessageFlowException

Description	Process Sequence Reply retransmissions.
Input arguments	Hashtable retransmissionTable
Process	Retrieves groupIds and its associated sequence numbers from retransmission Hashtable and calls processMessageProcessingFailureFault(groupId, seqNums[]) method for each groupId.
Output	-
Exceptions	WsrStorageException MessageFlowException
Comments	-

public void processMessageProcessingFailureFault(String groupId, long[]seqNum) throws WsrStorageException, MessageFlowException

Description	Process Retransmissions.
Input arguments	String groupId, long[] seqNum
Process	Retrieves Message Info from the database for each sequence Number of given groupId. Checks the number of retries entry of each message. If number of retries is less than three and retransmission time is greater than current time then it will retransmit the message. If number of retries is = 3 or > 3 then it will set permanent processing fault for that message.
Output	-
Exceptions	WsrStorageException MessageFlowException
Comments	-

public void processNonSequenceReplyFault(Hashtable nonSequenceReplyFault) throws WsrStorageException

Description	Process Non Sequence Reply Faults other than MessageProcessingFailure Fault.
Input arguments	Hashtable nonSequenceReplyFault
Process	Retrieves groupIds and its related faults from nonSequenceReplyFault Hashtable and calls processFault (groupId, sequenceNumber, fault) method for each groupId where

	sequenceNumber for Non Sequence Reply is zero as there is only single message in question.
Output	-
Exceptions	WsrStorageException
Comments	-

public void processSequenceReplyFault(Hashtable sequenceReplyFault) throws WsrStorageException

Description	Process Sequence Reply Fault.
Input arguments	Hashtable sequenceReplyFault
Process	Retrieves groupId, sequence number and its associated fault from sequenceReplyFault Hashtable and calls processFault(groupId, sequencenumber, fault) method.
Output	-
Exceptions	WsrStorageException MessageFlowException
Comments	-

public boolean processFault(String groupId, long sequenceNumber, QName fault) throws WsrStorageException

Description	Process Fault.
Input arguments	String groupId, long sequenceNumber, QName fault
Process	Calls processFaultOnGroup(groupId, sequenceNumber, fault) method of WsrProtocolStorageOperations class.
Output	-
Exceptions	WsrStorageException MessageFlowException
Comments	-

public void throwUnknownGroupFaultException(AddressingHeaders addressingHeaders, String groupId) throws WsFaultException

Description	Throws Unknown Group Fault Exception
Input arguments	AddressingHeaders addressingHeaders, String groupId
Process	Creates WsFaultException for Unknown group and throws it.
Output	-
Exceptions	WsFaultException
Comments	-

Class: public class WsrUtil

public boolean isExpiryTimePassed(Calendar expiryTime)

Description	Checks whether time expressed by expiryTime is expired
Input arguments	Calendar expiryTime
Process	Compares expiryTime with current system time. If time is expired returns true else returns false
Output	Boolean
Exceptions	-
Comments	-

public boolean isDurationPassed(GDuration duration)

Description	Checks whether duration is expired
Input arguments	GDuration duration
Process	Compares duration with current system duration. If duration is expired returns true else returns false
Output	Boolean
Exceptions	-
Comments	-

public String getStringOfCalendar(Calendar calendar)

Description	Creates String representation of calendar time
Input arguments	Calendar calendar
Process	Returns string representation of calendar
Output	String
Exceptions	-
Comments	-

public String getStringOfDuration(GDuration duration)

Description	Creates String representation of duration
Input arguments	GDuration duration
Process	Returns string representation of duration
Output	String
Exceptions	-
Comments	-

public Calendar getCalendarOfString(String calendar)

Description	Creates Calendar object from String calendar
Input arguments	String calendar
Process	Returns Calendar object from String calendar
Output	Calendar
Exceptions	-
Comments	-

public Duration getDurationOfString(String duration)

Description	Creates Duration object from String duration
Input arguments	String duration
Process	Returns Duration object from String duration
Output	Duration
Exceptions	-
Comments	-