```
;;;;
;;;; Created       : 2006 Aug 09 (Wed) 11:13:54 by Harold Carr.
;;;; Last Modified : 2006 Oct 18 (Wed) 14:37:42 by Harold Carr.
;;;;
```

1. Introduction

1.1. Project/Component Working Name:

Web Services Interoperability Technology (WSIT).
Also known as "Project Tango."

1.2. Name(s) and e-mail address of Document Author(s)/Supplier:

Harold Carr

harold.carr@sun.com

1.3. Date of This Document:

08/09/06

2. Project Summary

2.1. Project Description:

JWS interoperability with WCF.

(JWS = Java Web Services)
(WCF = Windows Communication Foundation)

2.2. Risks and Assumptions:

WSIT cannot ship until after WCF ships.
(For WSIT to officially interop with a Microsoft product
we need to document explicit products and settings with
which we have tested interoperability.)

3. Problem Summary

3.1. Problem Area:

On-the-wire interoperability with Windows Communication
Foundation, Microsoft's consolidated communication platform.

3.2. Justification:

Our customers have heterogeneous data centers and partners. They need to be able to interoperate between Java and the Windows Environment.

Project Tango is building on the previous generation of WS-I (I = Interoperability) technology. The point here is that the Java platform is already in the business of WS interop and WSIT is a continuation of that work with a WCF focus.

4. Technical Description:

4.1. Details:

From a developer point-of-view the main features enabled WSIT are:

* bootstrapping communication
* securing communication
* optimizing communication
* enabling reliability
* enabling atomic transactions

The WSIT implementation is composed of the following subsystems:

* Metadata

  o WS-Policy: Policies express and handle requirements and capabilities of web service consumers and providers. Think of it as an XML-based configuration language.

  o WS-MetadataExchange/WS-Transfer: WS-Transfer is a protocol to enable a consumer to request a service's metadata (i.e., its WSDL and policies). The reply is formatted according to WS-MetadataExchange ("MEX"). Think of Transfer/MEX as a bootstrap mechanism for communication.

* Security

  o WS-SecurityPolicy: Defines specific policies that describe how messages are secured.

  o WS-Security: Provides message content integrity and confidentiality (even in the presence of intermediaries).

  o WS-Trust: Provides methods for issuing, renewing, and

validating security tokens used by WS-Security. It also
provides ways to establish and broker trust relationships

  - o WS-SecureConversation: Can be viewed as a security
    optimization (i.e., better message level security and
    efficiency in multiple-message exchanges).

* QoS

  - o WS-ReliableMessaging: Enables a messaging system to
    recover from failures caused by messages that are lost or
    misordered in transit.

  - o WS-Coord: A framework for providing protocols that
    coordinate the actions of distributed applications. Used by
    WS-AtomicTransactions.

  - o WS-AtomicTransactions: Supports two phase commit semantics
    such that either all operations invoked within an atomic
    transaction succeed or are all rolled back.

NOTE: the above WS-* are specifications in various stages
of standardization at Oasis and the W3C.

* Transport

  - o SOAP/TCP: A Sun-proprietary transport to increase the
efficiency
    of communication. This is *NOT* a WSIT release driver
(whereas
    all of the above are release drivers).

4.2. Bug/RFE Number(s):

NONE.

4.3. In Scope:

Shown above.

4.4. Out of Scope:

Other vendor's are implementing the WS-* specifications.
Although we hope to interoperate with other vendor's, our
resources are focues on WCF as the primary target of
interoperability for WSIT.

4.5. Interfaces:

WSIT (in general)

*** Interfaces Exported ***

Interface	Classification	Comments
<server>.xml	committed	Server configuration file when starting from Java. For a class annotated as a web service: package com.foo; @WebService() public class Bar { ... } the file will be named: wsit-com.foo.Bar.xml and will live in WEB-INF/ for web container deployments or META-INF/ for ejb (i.e, JSR 109) deployments For inner classes annotated with @WebService the name will be: wsit-com.foo.Bar$Inner.xml This file/format is the only way a user of WSIT feature configures those features. The format of this file is WSDL 1.1. It uses standard WSDL element extensibility to embed policy assertions that control the configuration.

<server>.wsdl	committed	Server configuration file when starting from WSDL.
		This file is named ,located
and		
		formatted (WSDL 1.1) exactly
as		
		specified in the JAX-WS 2.0 specification.
		The only difference is that it will contain embedded policy assertions in WSDL element extensions as specified in
[??]		
wsit-client.xml	committed	Client configuration file.
		This file is named wsit-
client.xml		
		and is located on classpath.
		Not always needed.
		Necessary to supply the location of client security keystores.
		Optionally can control Reliable Messaging parameters.
		The format of this file is WSDL 1.1 with embedded policy assertions.
??POLICY ASSERTIONS??	committed	The set of legal assertions that may be contained in the configuration files.

*** Interfaces Imported ***

Interface ---------	Classification --------------	Comments --------
com.sun.xml.ws.api.*	uncommitted	WSIT is completely dependent on the internal APIs provided by the JAX-WS 2.1 Reference

		Implementation to enable pluggable subsystems. WSIT and JAX-WS are essentially the same development engineering, management, etc., teams. Therefore there is close day-to-day cooperation and coordination.

Bootstrapping (MEX/Transfer)

*** Interfaces Exported ***

Interface	Classification	Comments
From JAX-WS 2.1: wsimport	committed	JAX-WS wsimport is extended such that, besides trying the JAX-WS standard http://...?wsdl approach to retrieving WSDL, it also uses the WS-Transfer protocol to retrieve WSDL which are returned in a format specified in WS-MetadataExchange [??]. The WSIT NetBeans module relies on this extension to retrieve WSDLs from WSIT and WCF service providers.

*** Interfaces Imported ***

Interface	Classification	Comments
From JAX-WS 2.1: com.sun.xml.ws.api.wsdl.parser. MetadataResolverFactory MetaDataResolver	uncommitted	Extended in WSIT code so JAX-WS runtime can find and execute MEX to retrieve WSDL.

```
  ServiceDescriptor.

---------------------------------------------------------------------------
---
Secure Conversation

*** Interfaces Exported ***

NONE.



*** Interfaces Imported ***

Interface                 Classification     Comments
---------                 --------------     --------
From JAX-WS 2.1:                             (SC/API)
com.sun.xml.ws.Closeable  uncommitted        Available to server
application
                                             developers.

                                             In JAX-WS, this interface is
                                             implemented by a client port
proxy
                                             or client Dispatch.

                                             SC extends the
                                             behavior of Closeable.close()
                                             to terminate the SC session
                                             with the service.

                                             This is done by having the
                                             middleware send a request
                                             to cancel the security context
                                             to the service being used.

---------------------------------------------------------------------------
---
Reliable Messaging:

*** Interfaces Exported ***

Interface                 Classification     Comments
---------                 --------------     --------
Session Key               volatile           (RM/API)
                                             Available to server
application
                                             developers.

                                             A String uniquely identifying
```

```
the
                                             client making the request
exposed
                                             as a named property of

javax.xml.ws.handler.MessageContext
                                             exposed in injected
                                             javax.xml.ws.WebServiceContext
                                             resources.

                                             The name of the property is
                                             "com.sun.xml.ws.sessionid"


Session User Data         volatile           (RM/API)
                                             Available to server
application
                                             developers.

                                             A Hashtable<String, Object>
                                             exposed as a named property of

javax.xml.ws.handler.MessageContext
                                             exposed in injected
                                             javax.xml.ws.WebServiceContext
                                             resources.

                                             The same Hashtable is exposed
in
                                             every request from the same
client
                                             instance.

                                             The name of the property is
                                             "com.sun.xml.ws.session"


com.sun.xml.ws.rm.jaxws.runtime.client.      Only used by SeeBeyond.
AcknowledgementListener
                          volatile           notify(String id, int number)
                                             called when a message
belonging
                                             to the given id is acked.

                                             (SeeBeyond usage: can discard
the
                                             message from its persistent
store
                                             when the notification is
received.)
```

```
com.sun.xml.ws.rm.jaxws.runtime.client.      Only used by SeeBeyond.
RMSource                  volatile
                                             createSequence methods to
enable
                                             to use RM SequenceIDs rather
                                             than create an unnecessary
                                             parallel set.

                                             (SeeBeyond usage: With this
                                             version, SeeBeyond creates the
                                             sequence and keeps the state
                                             necessary to reinitialize
                                             after a system crash.
                                             In a future version, SeeBeyond
                                             uses persisted data to
                                             reinitialize the sequence
after
                                             a system crash.)

com.sun.xml.ws.rm.jaxws.runtime.server.      Only used by SeeBeyond.
RMDestination             volatile
                                             createSequence message only
used
                                             to reinitialize server side of
                                             RM after server failure.

                                             (SeeBeyond usage: reinitialize
                                             sequence with persisted data
                                             after a system crash.)

RM Server RM Sequence ID  volatile           Only used by SeeBeyond.

                                            Server-side MessageContext
Property

                                             A String uniquely identifying
the
                                             RM Sequence to which the
request
                                             belongs.

                                             Exposed as a named property of
the

javax.xml.ws.handler.MessageContext

                                             exposed in injected
                                             javax.xml.ws.WebServiceContext
                                             resources.
                                             The name of the property is
                                             "com.sun.xml.ws.rm.sequenceid"

                                             (SeeBeyond usage: checks the
                                             incoming sequence id on each
                                             request, using it to access
the
                                             inbound sequence and store
enough
                                             data in persistent state to be
                                             able to reinitialize the
sequence
                                             after a system crash.)


Message Number            volatile          Only used by SeeBeyond.

                                             Client-side BindingProvider
                                             RequestContxt Property
                                             An Integer specifying the RM
                                             MessageNumber to be used on
                                             request messages from the
                                             BindingProvider.

                                             The name of the property is

"com.sun.xml.ws.rm.messagenumber"

                                             (SeeBeyond usage: passes the
                                             message number to be used on
each
                                             request message and stores the
                                             number in persistent state
                                             along with the message.
                                             After a restart it uses the
same
                                             message number on every resend
of
                                             the message.)

RM Client Sequence ID     volatile          Only used by SeeBeyond.

                                             Client-side BindingProvider
                                             RequestContxt Property
                                             An String specifying the RM
                                             SequenceID to be used on
                                             request messages from the
                                             BindingProvider.
```

The name of the property is "com.sun.xml.ws.rm.sequenceid"

(SeeBeyond usage: passes the sequence id to be used on each request message and stores the number in persistent state along with the message. After a restart it uses the same sequence id on every resend of the message.)

*** Interfaces Imported ***

Interface	Classification	Comments
From JAX-WS 2.1: com.sun.xml.ws.Closeable	uncommitted	(RM/API) Available to server application developers. In JAX-WS, this interface is implemented by a port proxy or Dispatch. RM extends the behavior of Closeable.close() to terminate the RM session.

Atomic Transactions:

*** Interfaces Exported ***

Interface	Classification	Comments
com.sun.ws.xml.api.tx.Participant	uncommitted	For use by a system level developer to create a Volatile AT Participant. Used to implement the Java side of WCF WS-AT interop. We do not expect typical Java application developers to need to create Volatile Participants. (Just as we don't expect application

```
                                             developer to use

javax.transaction.TransactionSynchronizationRegistry.
                                             Both concepts are used to
                                             flush in memory cache to
                                             persistent storage
                                             before 2 pc of durable data.)

                                             This API accomplishes the
                                             equivalent functionality as

TransactionSynchronizationRegistry
              (see http://jcp.org/aboutJava/communityprocess/maintenance/
jsr907/907ChangeLog.html#interface_top)

                                             Both JTA 1.1 functionality and
                                             Volation AT-Participants are
used
                                             to flush in-memory cache to
                                             persistent store before 2
phase
                                             commit is performed on
durable,
                                             persistent resources.
              (see http://jcp.org/aboutJava/communityprocess/maintenance/
jsr907/907ChangeLog.html)

>

                                             Used by the WSIT
implementation
                                             of the WS-AT Coordinator and
                                             WS-AT Interop Service.

                                             Equivalent to

javax.transaction.xa.XAResource
                                             for WS-Atomic Transaction.

                                             The only usage for the
participant
                                             is with
                                ATTransaction.enlistParticipant(Participant
p)
                                             (above).

                                             Used to perform auto-
enlistment of
                                             WS-AT participants (i.e.,
                                             DataSources with XAResources
```

```
are
                                             auto-enlisted as part of a
                                             transaction by app server)

                                             Java developers will *not*use
this
                                             API.

                                             Used to test interop of WS-AT
                                             (i.e, we need a capability to
                                             create a volatile WS-AT
                                             participant.  No equivalent
                                             in Java transcations).

                                             Since WSIT itself is the only
                                             client of the interface we
will
                                             probably remove this interface
                                             from the expor list.


com.sun.ws.xml.api.tx.ATTransaction
(implements javax.transaction.Transaction)
                          uncommitted        Method called
enlistParticipant(
                                             com.sun.ws.xml.tx.Participant)
                                             that is equivalent to

javax.transaction.Transaction.enlistResource(XAResource).
                                             Used to register a Volatile
                                             Participant.
ManagedConnections
                                             use this in the AS
environment.

com.sun.xml.ws.api.tx.Protocol
                          uncommitted        Enum that defines WS-Atomic
                                             Transaction protocols.  Used
                                             by Participant.

com.sun.xml.ws.api.tx.TXException
                          uncommitted        Thrown by Participant.prepare

wscoor.wsdl & wsat.wsdl   uncommitted        Two JSR-109 hosted web
services:
                                             Coordinator and
WSATCoordinator.

Files *generated* by above *.wsdl (all uncommitted):
```

```
package com.sun.org.xmlsoap.ws.coord:

RegistrationCoordinatorPortType.java          External participant registers
                                              for a coordinated activity,
                                              includes Participant's
                                              EndpointReference.

RegistrationRequesterPortType.java            Receive a register reply from
an

                                              external Coordinator, includes
                                              external Coordinator's
                                              EndpointReference.

ActivationCoordinatorPortType.java            External client request for
                                              creation of a coordinated

activity

                                              (optional).

ActivationRequesterPortType.java              Receive external coordinator's
                                              response for the creation of a
                                              coordinated activity.

(optional)

package com.sun.org.xmlsoap.ws.at:

CoordinatorPortType.java                      WS-Atomic Transaction 2 phase
                                              commit coordinator (represents

all

                                              coordinated activities for

this

                                              AS).

ParticipantPortType.java                      WS-Atomic Transaction 2 phase
                                              commit participant (represents

all

                                              participants for this AS).

CompletionCoordinatorPortType.java            Handle external client's
direction

                                              to attempt to commit or

rollback

                                              transaction scope owned by

this

                                              root coordinator (optional).

CompletionInitiatorPortType.java              Receive result of two phase
                                              commit, committed or rolled

back

                                              (optional).
```

The optional endpoints are to support a transactional client that is remote from its root WS-AT coordination service. This is not the default usage model. Typically, the root coordinator and transaction scope creator are on same platform and use local, non-web service methods to establish transactional scope and for the client to denote the transaction should commit or rollback. (Equivalent to UserTransaction.commit()/rollback() and the result returned from this call.)

MS has following transport requirements to communicate with their WS-AT Coordinator web service. SOAP 1.1, 2004 WS-Addressing, X.509 certificates (used to establish Transaction Manager Identity), client/server authentication is required. Additionally: X.509 certificates presented over the wire must have a subject name that matches the fully qualified domain name (FQDN) of the originating machine. Therefore, DNS must be functional between each sender-receiver pair in the system for X.509 subject name checks to succeed.

*** Interfaces Imported ***

Interface	Classification	Comments
javax.transaction. Synchronization, Status, Transaction, TransactionManager, TransactionSynchronizationRegistry, UserTransaction, *Exception	committed	Provides the API that defines the contract between the transaction manager and the various parties involved in a distributed transaction namely: resource manager, application, and application server.
javax.transaction.xa. XAResource, XID, XAException	committed	Provides the API that defines the contract between the transaction manager and the resource manager, which allows the transaction manager to enlist and delist resource objects (supplied by the resource manager driver) in JTA transactions.
javax.resource.spi.	committed	Contains APIs for the system

```
XATerminator                                  contracts defined in the J2EE
                                              Connector Architecture
                                              specification.


javax.naming.*             committed          For JNDI lookup of Java EE
                                              transaction manager and user
                                              transaction.

com.sun.enterprise.transaction.TransactionImport
                           uncommitted        JCA 1.5 implemented a
                                              Transaction Inflow contract to
                                              enable external transactions
to be
                                              injected into AS.
                                              To use this capability, one
needs
                                              to write a resource adapter.
                                              Since there is no way that we
                                              could inject a resource
adapter
                                              in the Tango WS-TX pipes, we
have
                                              exported the transaction
methods
                                              used to implement the
Transaction
                                              Inflow contract for JCA 1.5.
                                              Binod and Sankar have reviewed
                                              and approved this change and
the
                                              motivation behind it.
                                              (An introductory description
of
                                              JCA 1.5 Transaction Inflow
                                              Contract can ve found at
              http://www.phptr.com/articles/article.asp?
p=383047&seqNum=2&rl=1)


---------------------------------------------------------------------------
---
Security Policy:

*** Interfaces Exported ***

NONE


*** Interfaces Imported ***
```

NONE

Security:

*** Interfaces Exported ***

Interface	Classification	Comments
XWSS 2.0 Exported Interfaces	committed	XWSS 3.0 will be backward compatible with XWSS 2.0, ARC CASE: 2005/245: http://sac.eng/arc/WSARC/2005/245/commitment.materials/xws-security/XWS_2_0.sxw
com.sun.xml.wss.impl.callback.SAMLCallback, SAMLAssertionValidator	committed	Handle SAMLPolicy Assertion scenarios.
Proprietary policy assertions	uncommitted	Used in WSIT config files to specify KeyStores and CallbackHandlers. (http://wsinterop.sfbay.sun.com/wssecurity/Keystore_Configuration.html)
Security profiles Profiles defined for evolving	uncommitted	Simplifies security config. Used by WSIT NetBeans module. (http://wsinterop.sfbay.sun.com/wssecurity/Profiles_For_WSSecurity.html)

*** Interfaces Imported ***

Interface	Classification	Comments
XWSS 2.0 Imported Interfaces	uncommitted	XWSS 3.0 will be backward compatible with XWSS 2.0, ARC CASE: 2005/245: http://sac.eng/arc/WSARC/2005/245/commitment.materials/xws-security/XWS_2_0.sxw Part of these interfaces are controlled by Apache Software.

```
javax.security.auth.callback.CallbackHandler Used to access AS 9.1
keystores
                          committed          and trustmanager.


SJSXP (https://sjsxp.dev.java.net/)          ??
                          ??


From StreamBuffer (https://xmlstreambuffer.dev.java.net/):
com.sun.xml.stream.buffer.MutableXMLStreamBuffer
com.sun.xml.stream.buffer.XMLStreamBuffer
com.sun.xml.stream.buffer.XMLStreamBufferException
com.sun.xml.stream.buffer.XMLStreamBufferMark
com.sun.xml.stream.buffer.stax.StreamReaderBufferCreator
                          ??                 Used to cache incoming message
                                             headers and replay them for
                                             security processing.


From StAX-Ex (https://stax-ex.dev.java.net/):
org.jvnet.staxex.Base64Data
org.jvnet.staxex.XMLStreamReaderEx
org.jvnet.staxex.NamespaceContextEx
                          ??                 Used to (efficiently) read
binary
                                             data when MTOM is enabled.


From JAXWS 2.1:
com.sun.xml.ws.api.SOAPVersion
com.sun.xml.ws.api.message.HeaderList
com.sun.xml.ws.api.message.Header
com.sun.xml.ws.api.message.Message
com.sun.xml.ws.encoding.TagInfoset
com.sun.xml.ws.message.AttachmentSetImpl
com.sun.xml.ws.message.stream.StreamMessage
com.sun.xml.ws.protocol.soap.VersionMismatchException
com.sun.xml.ws.streaming.XMLStreamReaderUtil
com.sun.istack.NotNull
com.sun.istack.Nullable
                          ??                 Used to wrap secured message
and
                                             headers into JAXWS Message
format.
                                             Also used to create an
incoming
                                             message after security
processing.
```

```
----------------------------------------------------------------------------
---
Trust:

*** Interfaces Exported ***

Interface                 Classification     Comments
---------                 --------------     --------
com.sun.xml.ws.security.trust.WSTrustContract Only used by AccessManager.
                          uncommitted
                                             Used to issue, validate,
cancel,
                                             renew customer tokens.

?? Needs to move up one level (not in impl).
?? Needs to be an interface for AccessManager.
com.sun.xml.ws.security.trust.impl.IssuedSAMLTokenContract
                          uncommitted        Only used by AccessManager.

                                             SAML implementation of
                                             WSTrustContract.
                                             Three methods:
                                             abstract CreateSAMLAssertion()
                                             abstract isAuthorized()
                                             abstract
getClaimedAttributes()

?? Needs to move up one level (not in impl).
?? Needs to be an interface for AccessManager.
com.sun.xml.ws.security.trust.impl.IssuedSAMLTokenContractImpl
                          uncommitted        Only used by AccessManager.

                                             Implementation of
                                             IssuedSAMLTokenContract.
                                             Provides the
CreateSAMLAssertion()
                                             method and defaults for other
two.
                                             The AccessManager product
                                             (?? URL) will extend this
                                             implementation and override
                                             isAuthorized() and
                                             getClaimedAttributes().

                                             QUESTION ??: Need more detail
                                             on what happens in the
override.


*** Interfaces Imported ***
```

NONE.

Policy:

*** Interfaces Exported ***

NONE

*** Interfaces Imported ***

Interface	Classification	Comments
JSR 109 Deployment Descriptor	committed	WSIT references the following elements: WS:WSDL-SERVICE (name), WS:WSDL-PORT.
javax.servlet.ServletContext	committed	When deploying apps in web containers this is used to get the location of WEB-INF to load the WSIT server config file.

SOAP/TCP

Note: this is the only WSIT subsystem that does *NOT* interop with
WCF. We tried licensing their SOAP/TCP and binary encoding technology
but the terms were not acceptable. WSIT includes this feature (as
well as the FastInfoset binary encoding built into JAX-WS) to be on
par with WCF.

*** Interfaces Exported ***

Interface	Classification	Comments
com.sun.xml.ws.transport.tcp.server.glassfish.WSTCPLifeCycleModule	committed	For SOAP/TCP to receive requests sent via TCP Sockets, it needs to register a listener on Grizzly

```
                                             instance(s) via AS LifeCycle
                                             module via domain.xml (see
below).



*** Interfaces Imported ***

Interface                 Classification     Comments
---------                 --------------     --------
Grizzly V1 (in appserv-rt.jar):
com.sun.enterprise.web.connector.grizzly.ByteBufferFactory
com.sun.enterprise.web.connector.grizzly.SelectorFactory
com.sun.enterprise.web.connector.grizzly.SelectorThread
com.sun.enterprise.web.connector.grizzly.Handler
com.sun.enterprise.web.connector.grizzly.algorithms.StreamAlgorithmBase
                          uncommitted        Used to implement SOAP TCP
port.


Glassfish (in appserv-tr.jar):
com.sun.appserv.server.LifecycleEvent
com.sun.appserv.server.LifecycleListener
com.sun.appserv.server.ServerLifecycleException
com.sun.enterprise.deployment.WebServiceEndpoint
com.sun.enterprise.webservice.NewEjbRuntimeEndpointInfo
com.sun.enterprise.webservice.JAXWSAdapterRegistry
com.sun.enterprise.webservice.WebServiceEjbEndpointRegistry
com.sun.enterprise.webservice.monitoring.Endpoint
com.sun.enterprise.webservice.monitoring.EndpointLifecycleListener
com.sun.enterprise.webservice.monitoring.WebServiceEngineFactory
com.sun.enterprise.webservice.monitoring.WebServiceEngine
                          committed          Used to expose SOAP TCP port.

domain.xml                committed          Changes to register a listener
                                             on Grizzly instance(s) via
                                             AS LifeCycle.

<domain application-root="${com.sun.aas.instanceRoot}/applications"
        log-root="${com.sun.aas.instanceRoot}/logs">
  <applications>
    <lifecycle-module
      class-
name="com.sun.xml.ws.transport.tcp.server.glassfish.WSTCPLifeCycleModule"
      enabled="true"
      is-failure-fatal="false"
      name="WSTCPConnectorLCModule">
    </lifecycle-module>
  ...
  </applications>
```

```
...
<servers>
  <server config-ref="server-config"
          lb-weight="100"
          name="server">
    <application-ref disable-timeout-in-minutes="30"
                     enabled="true"
                     lb-enabled="false"
                     ref="WSTCPConnectorLCModule"/>
  ...
  </server>
</servers>
```

4.6. Doc Impact:

WSIT has a tutorial on how to build JAX-WS providers and consumers that use WSIT features.

4.7. Admin/Config Impact:

WSIT has a NetBeans pluggin that is used to configure WSIT-enabled JAX-WS providers and consumers.

4.8. HA Impact:

NONE.

4.9. I18N/L10N Impact:

Error messages are localized in the same manner as WSIT's underlying JAX-WS platform.

4.10. Packaging & Delivery:

Unknown if WSIT impacts existing packages, clusters or metaclusters.

No impact on install nor upgrade.

4.11. Security Impact:

WSIT includes the implementation of XWSS 3.0 security.

4.12. Compatibility Impact

WSIT is new so has no compatibility issues.

XWSS 3.0 must be backward compatible with XWSS 2.0.

4.13. Dependencies:

WSIT depends on JAX-WS 2.1 (the "rearchitected" implementation). If JAX-WS or JAXB MRs slip WSIT will slip.

WS-AtomicTransactions/WS-Coordination depend on public and private interfaces exposed by the application server's transaction subsystem.

5. Reference Documents:

All WSIT related material can be found at the following web sites:

http://wsit.dev.java.net/ (code, how-tos, documentation, ...)
http://java.sun.com/webservices/interop/ (articles, ...)

For this first release of WSIT, the goal is interoperability with Microsoft's WCF regardless of standards. The WSIT implementation uses the following specifications as guidelines for WCF interoperability. However, WCF does not follow these specifications completely nor exactly. We therefore do whatever it takes, regardless of the specification, to ensure interoperability with WCF.

The following specifications are in various stages of pre-submission, submission and voting at different standards bodies (e.g., W3C, Oasis). None are final. Future WSIT releases may incorporate the resulting standards based on these specifications.

The following list is included for completeness. We are *NOT* claiming interoperability with anyone implementing these specifications. We *ARE* claiming interoperability with WCF's implementation/interpretation of these specifications.

Bootstrapping:

WS-MetadataExchange:
  http://wsinterop.sfbay/wsmex/presos/wsmex.pdf
WS-Transfer (only the part referenced by wsmex):

```
    http://www.w3.org/Submission/2006/SUBM-WS-Transfer-20060315/


Security Optimization:
  WS-SecureConversation
    http://msdn.microsoft.com/library/en-us/dnglobspec/html/WS-
secureconversation.pdf


Reliable Messaging:
  WS-ReliableMessaging
    http://msdn.microsoft.com/library/en-us/dnglobspec/html/WS-
ReliableMessaging.pdf
  WS-ReliableMessaging Policy
    http://msdn.microsoft.com/library/en-us/dnglobspec/html/WS-RMPolicy.pdf


Atomic Transactions:
  WS-Coordination
    http://wsinterop.sfbay/wscoord/spec/WS-Coordination.pdf (local copy)
    ftp://www6.software.ibm.com/software/developer/library/WS-
Coordination.pdf (external location)
  WS-Atomic Transaction
    http://wsinterop.sfbay/wstx/at/spec/WS-AtomicTransaction.pdf  (local
copy)
    ftp://www6.software.ibm.com/software/developer/library/WS-
AtomicTransaction.pdf (external location)


Security:
  WS-SecurityPolicy:
    http://specs.xmlsoap.org/ws/2005/07/securitypolicy/ws-
securitypolicy.pdf
  WS-Trust
    http://msdn.microsoft.com/library/en-us/dnglobspec/html/WS-trust.pdf
  WS-Security:
    ?? ws-security spec
    http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-
os-SOAPMessageSecurity.pdf
    http://www.oasis-open.org/committees/download.php/16782/wss-v1.1-spec-
os-UsernameTokenProfile.pdf
    http://www.oasis-open.org/committees/download.php/16785/wss-v1.1-spec-
os-x509TokenProfile.pdf
    http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-spec-
os-SAMLTokenProfile.pdf


Policy (used to configure the above):
  Web Services Policy 1.2 - Framework (WS-Policy):
    http://www.w3.org/Submission/WS-Policy/
  Web Services Policy 1.2 - Attachment (WS-PolicyAttachment):
    http://www.w3.org/Submission/WS-PolicyAttachment/


6. Schedule:
    6.1. Projected Availability:

          Milestone 2: Sept 2006
          Milestone 3: Oct 2006
          Feature Complete: Oct 2006
          Will be available from AppServer 9.1 is released


;;; End of file.
```