WS-I Testing Tools version 1.1 User Guide

Document Type:

Technical User Guide

Editors:

Jacques Durand, Fujitsu
Brian Macker, Computer Associates
Shrikant Wagh, Optimyz Software, Inc.
Simeon Greene, Oracle Corporation
Keith Stobie, Microsoft Corporation
David Lauzon, IBM

Last Edit Date:

11/10/2004 11:00:00 PM

Document Status:

Version 1.1

This document is a Working Group Draft; it has been accepted by the Working Group as reflecting the current state of discussions. It is a work in progress, and should not be considered authoritative or final; other documents may supersede this document.

Acknowledgement from the Editors: Our thanks to all the members of the Testing Work Group who developed the tool specifications and contributed valuable input and comments on this User Guide.

Table of Contents:

1 Overview

For questions like:

- Can testing tools certify that a Web Service is conforming to the Profile?
- Can testing tools verify all the requirements of a Profile?
- Are there some restrictions in using the testing tools?

See the 7 Frequently Asked Questions (FAQ) section.

1.1 General Testing Process and Architecture

The Web Services Interoperability Organization (WS-I) has developed testing tools that evaluate Web services conformance to Profiles. These tools test Web service implementations using a non-intrusive, black box approach. The tools focus is on the interaction between a Web service and user applications.

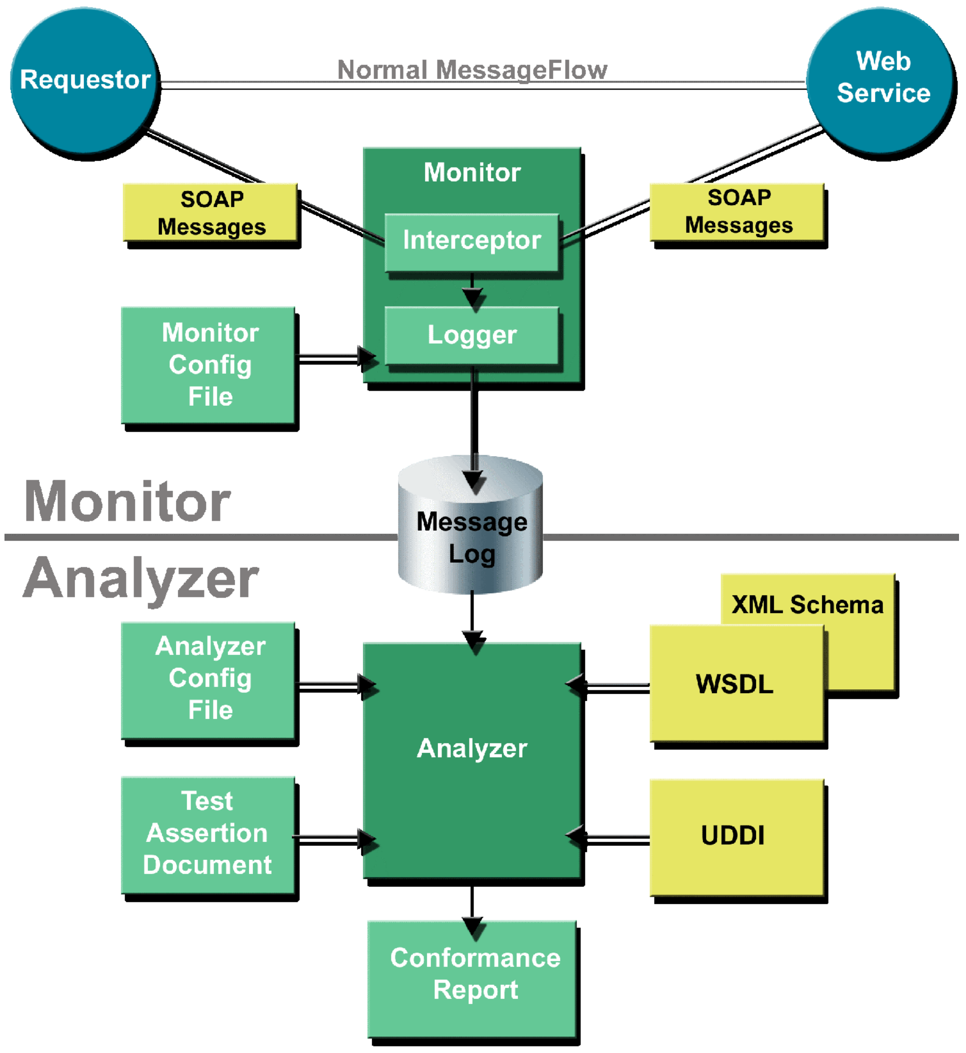


Figure 1 - Testing Tools Architecture.

The testing infrastructure is comprised of the Monitor and the Analyzer (see Figure 1) and a variety of supporting files:

- **Monitor** – This is both a message capture and logging tool. The interceptor captures the messages and the logger re-formats them and stores them for later analysis in the message log. The monitor is implemented using a man in the middle approach to intercept and record messages.

- **Analyzer –** This is an analysis tool that verifies the conformance of Web Services artifacts to Profiles. For example, it analyzes the messages sent to and from a Web service, after these have been stored in the message log by the Monitor.

- **Configuration Files** – These are XML files used to control the execution of the Testing Tools:
    - **Monitor Configuration File –** controls the execution of the monitor
    - **Analyzer Configuration File –** controls the execution of the analyzer
    - **Test Assertion Document** – defines the test assertions that will be processed by the analyzer

Other files or data artifacts will be accessed, which are not part of the test framework, but dependent on the Web Service to be tested:

- **Web Service artifacts –** these inputs to the Analyzer are target material for testing, and will be reported on:
    - **Message Log** – contains the monitoring trace of messages captured at transport level.
    - **WSDL definitions** - contains the definitions related to the Web Service
    - **UDDI entries** - contains references to Web Service definitions, as well as bindings.

- **Generated Files** – These are XML files produced by Testing Tools, that are specific to the Web Service being tested:
    - **Message Log –** (also a “Web Service artifact”)
- **Conformance Report** – contains the complete conformance analysis from the specified inputs.

2 Installation

2.1 *System Requirements*

Both the Monitor and Analyzer tools are available in C# and Java versions from the WS-I web site. The requirements for each are listed as follows:

- **C#** - The Microsoft .NET Framework release 1.1 must be installed. This can be obtained from the Microsoft download web site at: http://www.microsoft.com/downloads
- **Java** – The Java 2 Runtime Environment release 1.3.1 or later must be installed. This can be obtained from the Java web site at http://java.sun.com/j2se/downloads.html

2.2 *Installation Procedure*

The WSI Test tools are available for download from www.ws-i.org.

2.2.1 C# Version

Unzip the installation package into a working directory.

2.2.2 Java Version

Unzip the installation package into a working directory. Before running any of the tools, you must set the WSI_HOME environment variable to the location of the installed files.

Example: **set WSI_HOME=C:\wsi-test-tools**

In the example above, c:\wsi-test-tools is the installation directory for the WS-I testing tools

3 WS-I Monitor Tool

3.1 *Brief Description*

The WS-I Monitor Tool is implemented with a "man in the middle" approach so that it can intercept all the SOAP messages between the consumer (Web Services Client) and the instance (Web Services). The monitor configuration file controls the operation of the monitor and defines the parameters to ensure the SOAP messages are properly routed. In this release, WS-I monitor and analyzer tools are designed to handle HTTP traffic only. The complete WS-I testing tools specifications are available at www.ws-i.org.

3.2 *Configuration Setup*

The user can specify the WS-I Monitor tool configuration using the monitorConfig.xml file. The monitorConfig.xml file contains the list of configuration options for WS-I Monitor Tool. monitorConfig.xsd file describes the XML schema for the WS-I Monitor tool configuration file and this schema file can be located in <wsi-test-tools-home>\common\schemas folder. Following table describes the list of available options and their usage in monitorConfig.xml file.

Element	**Description**	**Usage**
configuration	This is the root element for the configuration file. This root element encloses all configuration parameters for the WS-I Monitor Tool.	Mandatory
comment	Provides descriptive information about the monitor configuration document. It does not affect the execution of the Monitor Tool. The element can be used as the immediate child of <configuration> and/or <redirect> element.	Optional
logFile	Using this element the user can specify the location of the log file that will contains the SOAP messages which then can be processed by the WS-I Analyzer tool.	Mandatory
logFile[@**replace**]	"**replace**" is an attribute of "logFile" element. Using this attribute the user can specify whether the existing log file specified by the value of "location" attribute can be overwritten. The allowed values for "**replace**" attribute are as follows: • true	Optional Default value – "false"

Element	**Description**	**Usage**
	Indicates that the already existing log file can be replaced. • false Indicates that if the log filename with the same name specified by the value of "location" attribute already exists then it cannot be replaced. If the values is set to "false" then the WS-I Monitor tool will terminate with an error message, if the log file with specified name already exists.	
logFile[@**location**]	"**location**" is an attribute of "logFile" element. Using this attribute the user can specify the location and the filename of the log file. The user can specify the filename as absolute path or the path relative to the current folder.	Mandatory
addStyleSheet	The user can specify the appropriate values for the attributes of this element to indicate whether the style sheet reference should be included in the log file which contains the SOAP messages intercepted by WS-I Monitor tool. This is an optional element and if this element is not specified in the configuration file, then the following comment line will be inserted in the log file after the XML declaration statement: `<!-- ?xml-stylesheet type="text/xsl" href="..\common\xsl\traceLog.xsl"? -->`	Optional
addStyleSheet[**@href**]	"**href**" is an attribute of "addStyleSheet" element. Using this attribute the user can specify the location and the filename of the style sheet. The user can specify the filename as absolute path or the path relative to the current folder. The specified style sheet will be used to render the report in HTML format.	Mandatory
addStyleSheet[**@type**]	"**type**" is an attribute of "addStyleSheet" element. Using this attribute the user can specify the content type for the style sheet.	Optional Default value – "text/xsl"
addStyleSheet[**@title**]	"**title**" is an attribute of "addStyleSheet" element. Using this attribute the user can specify the brief description or advisory information about the style sheet.	Optional

Element	Description	Usage
addStyleSheet[**@media**]	"**media**" is an attribute of "addStyleSheet" element. Using this attribute the user can specify the intended destination medium for the style sheet.	Optional
addStyleSheet[**@charset**]	"**charset**" is an attribute of "addStyleSheet" element. Using this attribute, the user can specify the character encoding for the style sheet.	Optional
addStyleSheet[**@alternate**]	"**alternate**" is an attribute of "addStyleSheet" element. The user can specify the appropriate value for this attribute to indicate the use of alternate style sheet. The allowed values for "**alternate**" attribute are as follows: • true Indicates that the use of alternate style sheet. • false Indicates that the alternate style sheet is not used.	Optional
logDuration	Using this element the user can specify the number of seconds for which the WS-I Monitor tool will accept the new client connections. After this specified duration the WS-I Monitor tool will stop accepting any new clients and write the intercepted messages to the log file.	Mandatory
cleanupTimeoutSeconds	Using this element the user can specify the time interval (in terms of seconds) to allow the existing communication to be over. Once logDuration time ends, the monitor begins to shutdown and accepts no new connections. To allow any existing conversations to finish, the monitor leaves the active ports alive for the number of seconds set in cleanupTimeoutSeconds. When this period ends, all ports are shut down.	Mandatory
manInTheMiddle	This is the wrapper element for all <redirect> elements and contains the configuration information for one or more manInTheMiddle port monitors.	Optional
redirect	The <redirect> elements define where the monitor will listen for traffic and how it will	Mandatory

Element	**Description**	**Usage**
	forward that traffic. The user can specify one or more redirects as desired.	
listenPort	Using this element the user can specify the listen port for the manInTheMiddle.	Mandatory
schemeAndHostPort	Using this element the user can specify the redirect URL for the received messages. When traffic is received on listenPort, it is sent to the URL specified by schemeAndHostPort element. The URL can be specified as: • *http://<host>:<port>* A subset form of an HTTP URL as specified in RFC1738 [2], section 3.3: *http://<host>:<port>/<path>?<searchpart>* The port is optional and defaults to 80.	Mandatory
maxConnections	Using this element the user can specify the maximum number of connections that the port will queue up before it begins refusing connections.	Mandatory
readTimeoutSeconds	Using this element the user can specify how long a listenPort should wait for a read operation before assuming that the connection timed out and releasing the connection. This timeout occurs when no data has been received by the client or server during this duration. If either end does send data, then neither connection is assumed to have timed out.	Mandatory

3.3 Sample Configuration File

This section illustrates the sample configuration file for usage of WS-I Monitor Tool.
The sample configuration file explained in this section assumes the following scenario:

- The user is intended to monitor the messages for two web services deployed at location URL http://www.coldrooster.com port 80 and http://www.tempuri.com port 80 respectively.
- The above mentioned two web services URL and ports are mapped the monitor URL and the monitor ports used for these two web services are port 9090 and port 8080 respectively.
- In order to insert the monitor between the requestor and web service some method of redirection is required. In this example it is assumed that the requestor is modified to redirect its request to the appropriate monitor ports.

The sample configuration file for WS-I Monitor tool for the above mentioned scenarios appears as listed in Fig. #2.

```
<?xml version="1.0" encoding="utf-8" ?>
<wsi-monConfig:configuration

  xmlns:wsi-monConfig=
    "http://www.ws-i.org/testing/2003/03/monitorConfig/">
  <wsi-monConfig:comment>This is a comment</wsi-monConfig:comment>
  <wsi-monConfig:logFile replace="true" location=”c:\traceLog.xml”>
        <wsi- monConfig:addStyleSheet href="../../common/xsl/log.xsl"/>
  </wsi-monConfig:logFile>
  <wsi-monConfig:logDuration>900</wsi-monConfig:logDuration>
  <wsi-monConfig:cleanupTimeoutSeconds>120</wsi-monConfig:cleanupTimeoutSeconds>
  <wsi-monConfig:manInTheMiddle>
    <wsi-monConfig:redirect>
      <wsi-monConfig:comment>Redirect for port 9090</wsi-monConfig:comment>
      <wsi-monConfig:listenPort>9090</wsi-monConfig:listenPort>
      <wsi-monConfig:schemeAndHostPort>http://www.coldrooster.com</wsi-
monConfig:schemeAndHostPort>
      <wsi-monConfig:maxConnections>1000</wsi-monConfig:maxConnections>
      <wsi-monConfig:readTimeoutSeconds>30</wsi-monConfig:readTimeoutSeconds>
    </wsi-monConfig:redirect>
    <wsi-monConfig:redirect>
      <wsi-monConfig:comment> Redirect for port 8080</wsi-monConfig:comment>
      <wsi-monConfig:listenPort>8080</wsi-monConfig:listenPort>
      <wsi-monConfig:schemeAndHostPort
         >http://www.tempuri.org:80</wsi-monConfig:schemeAndHostPort>
      <wsi-monConfig:maxConnections
         >1000</wsi-monConfig:maxConnections>
      <wsi-monConfig:readTimeoutSeconds
         >30</wsi-monConfig:readTimeoutSeconds>
    </wsi-monConfig:redirect>
  </wsi-monConfig:manInTheMiddle>
</wsi-monConfig:configuration>
```

Figure 2 - Sample Configuration file for WS-I Monitor tool.

The sample configuration file listed in Fig #2 instructs the monitor to perform the following tasks:

1. Log all messages to a file named **c:\traceLog.xml**. Replacing any existing file with the same name.
2. Set the duration of the testing session to **900** seconds
3. Run a man in the middle listener.
4. Open up listener connections on port: **9090** and **8080**.
5. Forward any traffic received on port 9090 to **www.coldrooster.com**, port **80**.
   Note: port 80 is the default and is not specified

6. Forward any traffic received on port 8080 to **www.tempuri.org**, port **80**.
7. Allow up to **1000** connections on ports 9090 and 8080.
8. Set the timeout value for a connection for ports 9090 and 8080 to **30** seconds.

3.4 WS-I Monitor Tool Command Line Syntax

Apart from configuration file, the user can also use the command line options for WS-I Monitor tool. All command line options override the options that are specified in the configuration file. Both – C# and Java – version of WS-I Monitor tools supports –config options, while Java version of WS-I Monitor tool support additional options.

The simplified command line syntax to invoke the WS-I Monitor tool is as follows:

```
Monitor -config <configFile>
```

3.4.1 C# WS-I Monitor Tool Command Line Options

The command line options for C# version of WS-I Monitor tool are described in detail in the following table:

Sr No	Option	Description
1	**-config**	The user can specify the configuration file to be used by the WS-I Monitor tool

3.4.2 Java WS-I Monitor Tool Command Line Options

The command line options for Java version of WS-I Monitor tool are described in detail in the following table:

Sr No	Option	Description
1	**-config** or **-c**	The user can specify the configuration file to be used by the WS-I Monitor tool.
2	**-verbose** or **-v**	Display diagnostic messages on the console.
3	**-comments** or **-C**	Using this option the user can provides descriptive information about the monitor configuration document. It does not affect the execution of the Monitor Tool.
4	**-logFile** or **-l**	Using this option the user can specify the location of the log file that will contains the SOAP messages which then can be processed by the WS-I Analyzer tool.
5	**-replace** or **-r**	Using this option the user can specify whether the existing log file specified by the value of “location” attribute can be overwritten.

Sr No	Option	Description
6	**-logDuration** or **-d**	Using this option the user can specify the number of seconds for which the WS-I Monitor tool will accept the new client connections. After this specified duration the WS-I Monitor tool will stop accepting any new clients and write the intercepted messages to the log file.
7	**-timeout** or **-t**	Using this option the user can specify how long a listenPort should wait for a read operation before assuming that the connection timed out and releasing the connection. This timeout occurs when no data has been received by the client or server during this duration. If either end does send data, then neither connection is assumed to have timed out.

3.4.3 Executing the C# Version of WS-I Monitor Tool

To run the C# version of WS-I Monitor tool, change directory to the <wsi-test-tool-home>\cs\bin folder and execute the following command:

Monitor [-config < configFilename >]

Example:

```
cd c:\wsi-test-tools\cs\bin
Monitor -config ..\samples\monitorConfig.xml
```

Note: If no configuration file is defined, the WS-I Monitor tool will default to monitorConfig.xml file in the current directory. In order to exit the WS-I Monitor tool, the user must either wait for the logDuration time to expire or manually exit the WS-I Monitor tool to properly close and complete the monitor log file. To manually exit the application, press <Ctrl-C> in the command prompt from where WS-I Monitor tool is running.

3.4.4 Executing the Java Version of WS-I Monitor Tool

To run the Java version of WS-I Monitor tool, change directory to the <wsi-test-tool-home>\java\bin folder and execute the following command:

Monitor [-config < configFilename >]

Example:

```
cd C:\wsi-test-tools\java\bin
Monitor -config ..\samples\monitorConfig.xml
```

Note: In order to exit the WS-I Monitor tool, the user must either wait for the logDuration time to expire or manually exit the WS-I Monitor tool to properly close and complete the monitor log file. To manually exit the application, press <Ctrl-C> in the command prompt from where WS-I Monitor tool is running.

3.4.5 How to Deploy the Monitor

The monitor uses the man-in-the-middle approach to monitor and record SOAP messages between clients and services. In order to integrate the monitor with a deployed Web Service, one of the three following techniques should be used:

1. Alter the Requestor
2. Move the Service
3. Alter the UDDI Registry entry

- **Alter the Requestor:** This technique involves altering the requestor, or client, to direct its requests to an alternative URL and/or port. This approach is usually the easiest where a testing program or test harness is being used to drive the server. In this case no modifications are required to the server, and the monitor can be used to test both internal and external Web services.

- **Move the Service:** In this case the service is moved to new location and/or port, and the monitor takes its place. This approach is best used where the client code cannot be modified, and the service can be conveniently modified or relocated.

- **Alter the UDDI Registry entry:** For applications where the connection (end point) is dynamically established through a UDDI registry, the monitor can be integrated by updating the UDDI entry. In this case the updated UDDI entry can refer to a WSDL definition where the address location in the service name has been updated to refer to the monitor, or, if the endpoint is specified in the accessPoint of the bindingTemplate, then the user can just modify this accessPoint.

3.4.6 The Monitor Output

WS-I Monitor tool logs all intercepted messages and the configuration parameters used, to a XML file. The monitor writes out each message pair (request and response) as it is processed, and the final </log> tag is not added until the monitor exits. When using the WS-I Monitor tool, you must terminate the monitor before you can view the log file. To manually exit,
press <Ctrl-C>.

It is important to note that the WS-I Monitor logs the HTTP conversation only, and makes no assumptions about the content of an HTTP body. WS-I Monitor does not check whether a message element contains a SOAP message. If a message contains an HTTP Content-Length header, it will not record the message until the number of bytes specified in that header is received. This feature is included primarily to deal with HTTP messages that issue an HTTP 100-continue request separate from sending the message body. (It is legal to send the HTTP headers independently of the message body with a HTTP Expect Header of 100-continue.)

In the following sections the log files generated by WS-I Monitor tool will be explained in detail with examples.

3.4.7 Message Log File Description

The log file created by WS-I Monitor tool contains the list of configuration options for WS-I Monitor Tool, and the request and response messages intercepted by WS-I Monitor. The structure of log file is defined by XML Schema. log.xsd file describes the XML schema for the WS-I Monitor tool log file and this schema file can be located in <wsi-test-tools-home>\common\schemas folder.

Following table describes the list of elements and their significance in log file.

Element	**Description**
log	This element is the root element for the WS-I Monitor log file. This root element encloses all configuration parameters, environment information for the WS-I Monitor Tool and the intercepted messages.
log[**@timestamp**]	"timestamp" is an attribute of "log" element. It specifies the time at which the log file was created. This value is less than or equal to the timestamp of the first messageEntry in the log.
monitor	"Monitor" element contains versioning information of the Monitor tool used. This element also contains the Monitor Tool configuration specified in the monitorConfig.xml file and the environment information.
monitor[@**version**]	"version" is an attribute of "monitor" element and displays the version of the WS-I Monitor tool.
monitor[@**releaseDate**]	"releaseDate" is an attribute of "monitor" element and displays the release date of the WS-I Monitor tool.
implementer	"implementer" is a child element of "monitor" element and identifies the organization that created the monitor tool.
implementer[@**name**]	"name" is as attribute of "implementer" element and displays the name of the organization that created the monitor tool.
implementer[@**location**]	"location" is as attribute of "implementer" element and displays the URL of the Web site from where the monitor tool can be obtained.
environment	"environment" is a child element of "monitor" element and describes the operating system and runtime environment used to run the WS-I Monitor application.

Element	Description
runtime	“runtime” is a child element of “environment” element and describes the runtime environment used by WS-Monitor tool.
runtime[@**name**]	“name” is as attribute of “runtime” element and displays the name of the name of the runtime environment. Ex: Java, .NET, etc.
runtime[@**version**]	“version” is as attribute of “runtime” element and displays the version of the runtime in use.
operatingSystem	“operatingSystem” is a child element of “environment” element and describes the operating system used to run the WS-I Monitor application.
operatingSystem[@**Name**]	“name” is as attribute of “operatingSystem” element and displays the name of the operating system.
operatingSystem[@**version**]	“version” is as attribute of “operatingSystem” element and displays the version of the operating System is use.
xmlParser	“xmlParser” is a child element of “environment” element and describes the XMP parser used by the WS-I Monitor application. This element is optional and will only be included if the XML Parser makes a difference.
xmlParser[@**name**]	“name” is as attribute of “xmlParser” element and displays the name of the XML parser used by WS-I Monitor tool.
xmlParser[@**version**]	“version” is as attribute of “xmlParser” element and displays the version of the XML parser used by WS-I Monitor tool.
configuration	“configuration” is a child element of “monitor” element and describes the configuration parameter that are specified in the monitor configuration file used to run the WS-I Monitor application.
messageEntry	“messageEntry” is a child element of “log” element and describes the messages intercepted by WS-I Monitor tool. Each “messageEntry” describes one intercepted request/response message.
messageEntry[@**timestamp**]	“timestamp” is an attribute of “messageEntry” element and displays the time when the message was intercepted by WS-I Monitor tool.
messageEntry[@ **conversationID**]	“conversationID” is an attribute of “messageEntry” element and this identifier is used to group messages received between the time that the client connects to the monitor port and when that connection is closed. The string is unique for each connection.

Element	Description
messageEntry[@**ID**]	"ID" is an attribute of "messageEntry" element and displays unique identifier for each intercepted message within the log.
messageEntry[@**type**]	"type" is an attribute of "messageEntry" element and displays the type information for the intercepted messages. The valid values for this attribute are as follows: • "request" Identifies the entry as a HTTP request. • "response" Identifies the entry as a HTTP response.
messageContent	"messageContent" is a child element of "messageEntry" element and contains the HTTP Body message in the HTTP request or HTTP response messages intercepted by WS-I Monitor tool. Special characters like '<' and '>' will be converted to their entity reference equivalents.
messageContent[@**BOM**]	"BOM" is an attribute of "messageContent" element and displays Byte Order Mark (if any) that was originally in the HTTP payload.
senderHostAndPort	"senderHostAndPort" is a child element of "messageEntry" element and identifies the host name and TCP port of the machine from where the message was sent. In a "request" type of messages, this will match the value of the port and host where the Web Services client is running. In a "response" type of messages, this will match the value of the port and host where the Web Services are deployed. When a port value is not specified, the default value of port 80 is assumed.
receiverHostAndPort	"receiverHostAndPort" is a child element of "messageEntry" element and identifies the host name and TCP port of the machine where the message will be received. In a "request" type of messages, this will match the value of the port and host where the Web Services are deployed. In a "response" type of messages, this will match the value of the port and host where the Web Services client is running. When a port value is not specified, the default value of port 80 is assumed.
httpHeaders	"httpHeaders" is a child element of "messageEntry" element and contain the HTTP headers of the corresponding request/response message.

3.4.8 Sample Message Log File

Fig #3 illustrates the sample log file generated by WS-I Monitor tool.

```
<?xml version="1.0" encoding="utf-8" ?>
<?xml-stylesheet href="..\..\common\xsl\log.xsl" type="text/xsl" alternate="no" ?>
<!-
Copyright (C) 2002-2003 by The Web Services-Interoperability Organization (WS-I) and
Certain of its Members. All Rights Reserved.
. . .  -->
<wsi-log:log
xmlns:wsi-log="http://www.ws-i.org/testing/2003/03/log/"
xmlns:wsi-monConfig="http://www.ws-i.org/testing/2003/03/monitorConfig/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" timestamp="2003-06-
13T13:10:03-07:00 ">
<wsi-log:monitor version="0.95.0.0" releaseDate="2003-06-04">
<wsi-log:implementer name="WS-I Organization"
location="http://www.ws-i.org/Testing/Tools/2003/06/WSI_Test_CS_0.95_bin.zip " />
<wsi-log:environment>
<wsi-log:runtime name=".NET" version="1.1.4322.573" />
<wsi-log:operatingSystem name="Win32NT" version="5.1.2600.0" />
<wsi-log:xmlParser name=".NET" version="1.1.4322.573" />
</wsi-log:environment>

<wsi-monConfig:configuration>
<wsi-monConfig:comment>This is a sample monitor config file
</wsi-monConfig:comment>
<wsi-monConfig:logFile location="traceLog.xml" replace="true">
<addStyleSheet alternate="false" href="..\..\common\xsl\log.xsl" type="text/xsl"
xmlns="http://www.ws-i.org/testing/2003/03/common/" />
</wsi-monConfig:logFile>
<wsi-monConfig:logDuration>900</wsi-monConfig:logDuration>
<wsi-monConfig:cleanupTimeoutSeconds>120</wsi-monConfig:cleanupTimeoutSeconds>
<wsi-monConfig:manInTheMiddle>
<wsi-monConfig:redirect>
<wsi-monConfig:comment>This redirects to local machine
</wsi-monConfig:comment>
<wsi-monConfig:listenPort>9090</wsi-monConfig:listenPort>
<wsi-monConfig:schemeAndHostPort>localhost:80
</wsi-monConfig:schemeAndHostPort>
<wsi-monConfig:maxConnections>1000</wsi-monConfig:maxConnections>
<wsi-monConfig:readTimeoutSeconds>30</wsi-monConfig:readTimeoutSeconds>
</wsi-monConfig:redirect>
</wsi-monConfig:manInTheMiddle>
</wsi-monConfig:configuration>
</wsi-log:monitor>
```

```
<wsi-log:messageEntry xsi:type="wsi-log:httpMessageEntry"
timestamp="2003-06-13T13:10:17.5540859-07:00"
conversationID="1" ID="1" type="request">
<wsi-log:messageContent BOM="4294851584">
&lt;?xml version="1.0" encoding="utf-16"?&gt;
&lt;soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"&gt;
&lt;soap:Body&gt;
&lt;getCatalog xmlns="http://www.ws-
i.org/SampleApplications/SupplyChainManagement/2002-08/Retailer.wsdl" /&gt;
&lt;/soap:Body&gt;
&lt;/soap:Envelope&gt;
</wsi-log:messageContent>
<wsi-log:senderHostAndPort>127.0.0.1:2302</wsi-log:senderHostAndPort>
<wsi-log:receiverHostAndPort>localhost:80</wsi-log:receiverHostAndPort>
<wsi-log:httpHeaders>POST /wsi/main/SampleApps/SupplyChain/Retailer/Retailer.asmx HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol 1.1.4322.573)
Content-Type: text/xml; charset=utf-16
SOAPAction: ""
Content-Length: 351
Expect: 100-continue
Host: 127.0.0.1:80
</wsi-log:httpHeaders>
</wsi-log:messageEntry>

<wsi-log:messageEntry xsi:type="wsi-log:httpMessageEntry" timestamp="2003-06-
13T13:10:18.0225269-07:00" conversationID="1" ID="2" type="response">
<wsi-log:messageContent BOM="15711167">
&lt;?xml version="1.0" encoding="utf-8"?&gt;
&lt;soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"&gt;
&lt;soap:Body&gt;
&lt;getCatalogResponse xmlns="http://www.ws-i.org/SampleApplications/SupplyChainManagement/2002-
08/Retailer.wsdl"&gt;
&lt;return xmlns=""&gt;
&lt;Item xmlns="http://www.ws-i.org/SampleApplications/SupplyChainManagement/2002-
08/RetailCatalog.xsd"&gt;
 . . .
&lt;/Item&gt;
&lt;/return&gt;
&lt;/getCatalogResponse&gt;
&lt;/soap:Body&gt;
&lt;/soap:Envelope&gt;
</wsi-log:messageContent>
<wsi-log:senderHostAndPort>localhost:80</wsi-log:senderHostAndPort>
<wsi-log:receiverHostAndPort>127.0.0.1:2302</wsi-log:receiverHostAndPort>
<wsi-log:httpHeaders>HTTP/1.1 200 OK
Server: Microsoft-IIS/5.1 Date: Fri, 13 Jun 2003 20:10:17 GMT
```

```
X-Powered-By: ASP.NET X-AspNet-Version: 1.1.4322
Cache-Control: private, max-age=0
Content-Type: text/xml; charset=utf-8
Content-Length: 3447
</wsi-log:httpHeaders>
</wsi-log:messageEntry>

 . . .

<wsi-log:messageEntry xsi:type="wsi-log:httpMessageEntry" timestamp="2003-06-
13T13:10:19.0225269-07:00" conversationID="2" ID="4" type="request">
<wsi-log:messageContent BOM=”15711167”>
&lt;?xml version="1.0" encoding="utf-8"?&gt;
&lt;soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"&gt;
&lt;soap:Body&gt;
&lt;logEventRequestElement xmlns="http://www.ws-
i.org/SampleApplications/SupplyChainManagement/2002-08/LoggingFacility.xsd"&gt;
&lt;DemoUserID&gt;
ee04714a-5b7f-4e2e-9d4b-0c3ceb27c715
&lt;/DemoUserID&gt;
&lt;ServiceID&gt;
ManufacturerA.submitPO
&lt;/ServiceID&gt;
&lt;EventID&gt;UC3-3&lt;/EventID&gt;
&lt;EventDescription&gt;
ManufacturerA is replenishing stock for 605008
&lt;/EventDescription&gt;
&lt;/logEventRequestElement&gt;
&lt;/soap:Body&gt;
&lt;/soap:Envelope&gt;
</wsi-log:messageContent>
<wsi-log:senderHostAndPort>127.0.0.1:2305</wsi-log:senderHostAndPort>
<wsi-log:receiverHostAndPort>localhost:80</wsi-log:receiverHostAndPort>
<wsi-log:httpHeaders>POST /wsi/main/SampleApps/SupplyChain/LoggingFacility/LoggingFacility.asmx
HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol 1.1.4322.573)
Content-Type: text/xml; charset=utf-8
SOAPAction: ""
Content-Length: 636
Expect: 100-continue
Host: 127.0.0.1:80
</wsi-log:httpHeaders>
</wsi-log:messageEntry>

<wsi-log:messageEntry xsi:type="wsi-log:httpMessageEntry" timestamp="2003-06-
13T13:10:19.1225269-07:00" conversationID="2" ID="5" type="response">
<wsi-log:messageContent />
```

```
<wsi-log:senderHostAndPort>localhost:80</wsi-log:senderHostAndPort>
<wsi-log:receiverHostAndPort>127.0.0.1:2305
</wsi-log:receiverHostAndPort>
<wsi-log:httpHeaders>HTTP/1.1 202 Accepted
Server: Microsoft-IIS/5.1
Date: Fri, 13 Jun 2003 20:10:19 GMT
X-Powered-By: ASP.NET X-AspNet-Version: 1.1.4322
Cache-Control: private
Content-Length: 0
</wsi-log:httpHeaders>
</wsi-log:messageEntry>
</wsi-log:log>
```

Figure 3 – Example of Message Log file.

3.4.9 Viewing the HTML version of the Message Log File

The XSL style sheet referred to by the XML processing instruction embedded at the beginning of the log file, control the rendering of the log file in HTML format. This XML processing instruction in XML log file appears as follows:

```
<?xml-stylesheet href="..\..\common\xsl\log.xsl" type="text/xsl" alternate="no" ?>
```

The XSL style sheet referred to by “href” attribute should be accessible from the directory where the log file is located. The common.xsl file referred to by log.xsl file should also be appropriately accessible in order to render the log file in HTML format.

When using IE to view the HTML version of the log file, it is preferred to use IE 5.5, or later version of IE. In order to view the log file in HTML format using IE, just open the log file using IE.

The user may notice the differences in namespace qualifications of the elements specified in traceLog.xml and in log.xsl. For example, an entry element in tracelog.xml is “log” whereas the entry in the log.xsl is “wsi-log:log”. There is a difference between the way that elements are namespace qualified in the traceLog.xml file and the way they are referenced in the log.xsl file. In the traceLog.xml file, the default namespace is "http://www.ws-i.org/testing/2003/03/log/", so there is no need to qualify the element names. In the log.xsl file, element references must be namespace qualified so that the XSL processor knows exactly which element to process. In the log.xsl file, "wsi-log" is the short name for the "http://www.ws-i.org/testing/2003/03/log/" namespace.

4 WS-I Analyzer Tool

4.1 *Brief Description*

WS-I Analyzer tool verifies the conformance of Web Services artifacts to WS-I Profiles. WS-I Analyzer tool analyzes the following three test targets:

- Web Services Description (i.e. WSDL document)
- Web Services Messages (i.e. SOAP Request and SOAP Responses including their HTTP Headers)
- Web Services Discovery (i.e. UDDI entries)

Thus, using the WS-I Analyzer tool the user can analyze the Web Service description (WSDL document), Envelopes and Messages (Request and/or Response messages) and Web Services Discovery (UDDI entries) for WS-I Profile conformance.

4.2 *WS-I Analyzer Processing Rules*

The Analyzer is controlled through the use of a Test Assertion Document (TAD). The TAD for the WS-I Basic Profile defines four primary artifacts: envelopes, messages, description, and discovery. These artifacts correlate to the <logFile>, <wsdlReference> and <uddiReference> elements in the configuration file. The following rules describe the expected combinations, and the behavior of the WS-I Analyzer Tool for these combinations:

- If only the <logFile> element is specified, then all envelopes and messages in a log file are processed by the analyzer. Any test assertions that had a Web Service description defined for a secondary entry type will be bypassed.
- The <wsdlReference> and <uddiReference> elements can not be specified together.
- If only the <uddiReference> element is specified, then the test assertions for both artifacts -the description artifacts and the discovery artifacts - are processed.
- If only the <wsdlReference> element is specified, then only the test assertions for the description artifact are processed.
- If the <logFile> and <wsdlReference> elements are specified, then the test assertions for both - the messages and the description - artifacts are processed. If the <wsdlElement> element contains a reference to a WSDL port or the <serviceLocation> element is specified, and there are messages for more than one Web Service in the log file, then only the messages that are associated with specified Web Service will be analyzed.
- If the <logFile> and <uddiReference> elements are specified, then the test assertions for the messages, description and discovery artifacts are processed. If the <uddiKey> element contains a reference to a bindingTemplate and there are messages for more

than one Web service in the log file, then only the messages that are associated with specified Web service will be analyzed.

- If the <logFile> element is specified with either a <wsdlReference> or a <uddiReference> and they do not contain a service location (i.e. wsdl:port element reference, uddi:bindingTemplate reference, or <serviceLocation> element), then the analyzer will terminate after processing the configuration options.
- If a <uddiReference> element contains a <wsdlElement> element, then the type attribute value is “binding”. If it is not, then the analyzer will terminate after processing the configuration options.
- If a <serviceLocation> element is specified within a <wsdlReference> element, the <wsdlElement> element contains a reference to either a <wsdl:port> or <wsdl:binding>. If it does not, then the analyzer will terminate after processing the configuration options. If the <wsdlElement> element contains a reference to a <wsdl:port>, then the value in the <serviceLocation> element is used instead of the value of the <soapbind:address> element within the <wsdl:port> element.
- If a <serviceLocation> element is specified within a <uddiReference> element, the <wsdlElement> element contains a reference to a <wsdl:binding>. If it does not, then the analyzer will terminate after processing the configuration options. If the <uddiKey> element contains a reference to a <uddi:bindingTemplate>, then the value in the <serviceLocation> element is used instead of the value of the <uddi:accessPoint> element within the <uddi:bindingTemplate>.
- A <uddiReference> element may contain a reference to a <uddi:bindingTemplate> which references more than one <uddi:tModel> that are categorized as “wsdlSpec”, or a <uddi:tModel> that references more than one <wsdl:binding>. When these conditions exist, the <wsdlElement> element with a type attribute value of “binding” will normally be used to indicate which <wsdl:binding> element to analyze.
- When a <uddiReference> element contains a valid <wsdlElement> element and the referenced <uddi:bindingTemplate> or <uddi:tModel> contains a reference to more than one <wsdl:binding>, if the specified <wsdl:binding> can not be found then the analyzer will terminate after detecting this condition.

There are certain situations where the input artifacts for the analyzer are not as complete as expected, such as an empty log file or incomplete WSDL description. The following table describes some of these cases, and explains the expected behavior of the analyzer in such cases.

Sr. No.	State of the Input Artifact	Analyzer behavior
1	Log file with no elements (no <messageEntry> element). This could happen when the monitor is started and stopped without receiving any messages.	The test assertions that target a request or response message as the primary entry, will have a result of “missingInput”.
2	A WSDL document is provided as	The analyzer will terminate with an error

	input but it does not contain the WSDL element which is specified on the <wsdlElement> element.	message which indicates that the WSDL document did not contain the expected WSDL element.
3	The analyzer configuration file contains a reference to a UDDI entry, but the UDDI entry does not exist.	The analyzer will terminate with an error message which indicates that the UDDI entry is not valid.
4	The analyzer configuration file contains a reference to a port, binding or portType, but the associated portType does not contain any operation definitions.	If a log file is not specified in the analyzer configuration file, then no special processing occurs. The test assertions with WSDL operation and WSDL message for primary entry types will not be processed. If a log file is specified and the correlation type is endpoint, then the correlation process can be done but any message-related test assertions with an additional entry type of operation or message will have a result of "notApplicable". If a log file is specified and the correlation type is namespace or operation, then there is no way to process the correlation function. When this condition is detected, then the analyzer will terminate with an error message that indicates that the WSDL service description did not contain enough information to process the correlation function.
5	The analyzer configuration file contains a reference to a WSDL or UDDI element that would indicate that other WSDL or UDDI elements should not be processed. For example, if the <wsdlElement> contains a reference to a portType element, then the binding element will not be processed.	The test assertions which contain an entry type that matches the elements that aren't processed will have a result of "missingInput".
6	A WSDL document is provided as input, but it does not contain elements that match all of the entry types in the WSDL-related test assertions. For example, a WSDL document may not contain a <types> element, but there are test assertions that have a primary entry type of "types".	The test assertions which contain an entry type which matches the missing elements will have a result of "missingInput".

4.3 Configuration Setup

The user can specify the analyzer configuration using the analyzerConfig.xml file. The analyzerConfig.xml file contains the list of configuration options for WS-I Analyzer Tool. analyzerConfig.xsd file describes the XML schema for the WS-I Analyzer tool configuration file and this schema file can be located in <wsi-test-tools-home>\common\schemas folder. Following table describes the list of available options and their usage in analyzerConfig.xml file.

Element	**Description**	**Usage**
configuration	This is the root element for the configuration file. This root element encloses all configuration parameters for the WS-I Analyzer Tool.	Mandatory
Configuration[@**name**]	"name" is an attribute of "configuration" element. The user can specify the name associated with the specified configuration using this element.	Optional
description	The user can specify the brief description for the specified configuration using this element.	Optional
verbose	The user can specify the desired value for "verbose" element to indicate whether diagnostic information should be displayed while the analyzer is running. The valid values for "verbose" element are "true" or "false". When running the analyzer tool from the command line, the diagnostic information is displayed on the console.	Optional Default value – "false"
assertionResults	The user can specify the appropriate values for the attributes of this element to indicate the type of assertion results that should be listed in the conformance report.	Mandatory
assertionResults[@**type**]	"type" is an attribute of "assertionResults" element. Using this attribute the user can specify the type of assertion results to be included in the conformance report. The allowed values for the "type" attribute and their significance are listed as below: • all List the results from all test assertions. • notPassed List all assertion test results except the ones that have a result of "passed". • onlyFailed List only the test assertion results that have a result of "failed". • notInfo List only the test assertion results that do not have a type of "informational".	Optional Default value – "all''

assertionResults[@**messageEntry**]	"messageEntry" is an attribute of "assertionResults" element. Using this attribute the user can specify whether the message entries should be included in the report file. The allowed values for "messageEntry" attribute are as follows: • true Includes the message entries in the report file. • false Does not include the message entries in the report file.	Optional Default value – "true"
assertionResults[@**assertionDescr iption**]	"assertionDescription" is an attribute of "assertionResults" element. Using this attribute the user can specify whether the description of each test assertion should be included in the report file. The allowed values for "assertionDescription" attribute are as follows: • true Includes the description of each test assertion in the report file. • false Does not include the description of each test assertion in the report file.	Optional Default value – "false"
assertionResults[@**failureMessag e**]	"**failureMessage**" is an attribute of "assertionResults" element. Using this attribute the user can specify whether the failure message defined for each test assertion should be included in the report file. The allowed values for "**failureMessage**" attribute are as follows: • true Includes the defined failure messages for each test assertion in the report file. • false Does not include the defined failure messages for each test assertion in the report file.	Optional Default value – "true"
assertionResults[@**failureDetail**]	"**failureDetail**" is an attribute of "assertionResults" element. Using this attribute the user can specify whether the failure detail message defined for each test assertion should be included in the report file. The allowed values for "**failureDetail**" attribute are as follows: • true Includes the defined failure detail messages for each test assertion in the report file. • false Does not include the defined failure detail messages for each test assertion in the report file.	Optional Default value – "true"

reportFile	The user can specify the appropriate values for the attributes of this element to provide the location of the report file and to indicate whether the existing report file with the same name should be overwritten.	Mandatory
reportFile[@**repl ace**]	"**replace**" is an attribute of "reportFile" element. Using this attribute the user can specify whether the existing report file specified by the value of "location" attribute can be overwritten. The allowed values for "**replace**" attribute are as follows: • true Indicates that the already existing report file can be replaced. • false Indicates that if the report filename with the same name specified by the value of "location" attribute already exists then it cannot be replaced. If the values is set to "false" then the analyzer will terminate with an error message, if the report file with specified name already exists.	Optional Default value – "false"
reportFile[@**locat ion**]	"**location**" is an attribute of "reportFile" element. Using this attribute the user can specify the location and the filename of the report file. The user can specify the filename as absolute path or the path relative to the current folder.	Optional Default value – "report.xml"
addStyleSheet	The user can specify the appropriate values for the attributes of this element to indicate whether the style sheet reference should be included in the conformance report created by the WS-I Analyzer tool. This is an optional element and if this element is not specified in the configuration file, then the following comment line will be inserted in the report file after the XML declaration statement: `<!-- ?xml-stylesheet type="text/xsl" href="..\common\xsl\report.xsl"? -->`	Optional
addStyleSheet[**@ href**]	"**href**" is an attribute of "addStyleSheet" element. Using this attribute the user can specify the location and the filename of the style sheet. The user can specify the filename as absolute path or the path relative to the current folder. The specified style sheet will be used to render the report in HTML format.	Mandatory
addStyleSheet[**@t ype**]	"**type**" is an attribute of "addStyleSheet" element. Using this attribute the user can specify the content type for the style sheet.	Optional Default value – "text/xsl"

addStyleSheet[**@title**]	"**title**" is an attribute of "addStyleSheet" element. Using this attribute the user can specify the brief description or advisory information about the style sheet.	Optional
addStyleSheet[**@media**]	"**media**" is an attribute of "addStyleSheet" element. Using this attribute the user can specify the intended destination medium for the style sheet.	Optional
addStyleSheet[**@charset**]	"**charset**" is an attribute of "addStyleSheet" element. Using this attribute, the user can specify the character encoding for the style sheet.	Optional
addStyleSheet[**@alternate**]	"**alternate**" is an attribute of "addStyleSheet" element. The user can specify the appropriate value for this attribute to indicate the use of alternate style sheet. The allowed values for "**alternate**" attribute are as follows: • true Indicates that the use of alternate style sheet. • false Indicates that the alternate style sheet is not used.	Optional
testAssertionFile	Using this element, the user can specify the location and the filename of the WS-I Test Assertion Document (TAD). The user can specify the filename as absolute path or the path relative to the current folder. The Analyzer tool will test for the WS-I conformance against the assertions defined in the TAD specified by this element.	Mandatory
logFile	Using this element the user can specify the location of the log file that contains the SOAP messages that need to be processed by the WS-I Analyzer tool. This is an optional element and if "logFile" element does not appear in the configuration file, then all test assertions that operate on the log entries will not be processed.	Optional

logFile[**@correlationType**]	"**correlationType**" is an attribute of "logFile" element. The user can specify the appropriate value for this attribute to define what information should be used to match the messages from the log file with the web services that are being tested. The allowed values for "**correlationType**" attribute are as follows: • endpoint Indicates a message is correlated to a Web service based only on the endpoint definition. This option is sufficient when a single Web Service is deployed on this endpoint. • namespace Indicates that the correlation process will use both - the endpoint and the namespace - to match a message to a Web service. This option is necessary when more than one Web Services are deployed on this endpoint. (The namespace allows for selecting the right one.) • ooperation Indicates that the correlation requires a match on the endpoint, namespace and a operation signature. This option is necessary when more than one Web Services are deployed on this endpoint, and they might use the same namespace (The operation allows for additional discrimination, although this will not be sufficient if both WS use same operation names.)	Optional Default value – "operation"
wsdlReference	This is a wrapper element for "wsdlElement", "wsdlURI" and "serviceLocation". Using this element the user can refer to the WSDL element and the WSDL file, which should be analyzed. At any time only of the two elements – either wsdlReference or UDDIReference - can appear in a configuration file. wsdlReference is an optional element and if this element does not appear in the configuration file, then the WSDL related test assertions will not be processed.	Optional
wsdlElement	"wsdlElement" can appear as a child element of "wsdlReference" or "uddiReference" element. Using this element, the user can specify the reference to the WSDL element that should be analyzed. The user can also specify the appropriate values for the attributes of this element to indicate the type, namespace and parent element of the referenced element.	Mandatory

wsdlElement[**@type**]	"**type**" is an attribute of "wsdlElement" element. The user can specify the appropriate value for this attribute to indicate the type of the element that is referenced by "wsdlElement". The allowed values for "**type**" attribute are as follows: • port Indicates the element that is referenced by "wsdlElement" is of type "port". • binding Indicates the element that is referenced by "wsdlElement" is of type "binding". • portType Indicates the element that is referenced by "wsdlElement" is of type "portType". • operation Indicates the element that is referenced by "wsdlElement" is of type "port". • message Indicates the element that is referenced by "wsdlElement" is of type "message".	Mandatory
wsdlElement[**@namespace**]	"**namespace**" is an attribute of "wsdlElement" element. Using this attribute, the user can specify the appropriate namespace for the element that is referenced by "wsdlElement".	Mandatory
wsdlElement[**@parentElementName**]	"**parentElementName**" is an attribute of "wsdlElement" element. The attribute is only required when the "type" attribute has a value of "port" or "operation". Specifying the appropriate value for "**parentElementName**" attribute, the user can qualify the reference to a WSDL port definition within a service element if the referenced element is of type "port". It can also be used to qualify the reference to an operation definition within a portType if the referenced element is of type "operation".	Optional
wsdlURI	"wsdlURI" is a child element of "wsdlReference". Using this element, the user can specify the location and the filename of the WSDL file for the Web Services under test. The user can specify the filename as absolute path or the path relative to the current folder. The user can also specify the location of the WSDL file as HTTP URI.	Mandatory

serviceLocation	“serviceLocation” can appear as a child element of “wsdlReference” or “uddiReference” element. Using this element, the user can specify the service location information for the element that is referenced by the corresponding “wsdlElement”. There are times when the service location is not defined in a WSDL document, but this information is required by the WS-I Analyzer tool. When this situation occurs, the <wsdlElement> element should reference a WSDL binding and “serviceLocation” element should contain the service endpoint location. If the <wsdlElement> element contains a reference to a wsdl:port and the <serviceLocation> element is specified, then the value in the <serviceLocation> element overrides the value of the “location” attribute of the <soapbind:address> element.	Optional

uddiReference	This is a wrapper element for "**uddiKey", "inquiryURL**", "wsdlElement" and "serviceLocation" elements. Using this element the user can specify a single UDDI bindingTemplate or tModel, bindingKey or tModelKey, inquiryURL and "wsdlElement" elements. At any time only of the two elements – either wsdlReference or uddiReference - can appear in a configuration file. uddiReference is an optional element and if this element does not appear in the configuration file, then the UDDI related test assertions will not be processed. If neither "uddiReference" nor "wsdlReference" element appears in the configuration file, then both - WSDL and UDDI – related test assertions will not be processed by the WS-I Analyzer tool.	Optional
uddiKey	"uddiKey" is a child element of "uddiReference" element. Using this element, the user can specify the value and type of the UDDI key. The user can also specify the appropriate values for the attributes of this element to indicate the type of the UDDI key reference by this element.	Mandatory
uddiKey[**@type**]	"**type**" is an attribute of "uddiKey" element. The user can specify the appropriate value for this attribute to indicate the type of the UDDI key referred by "uddiKey" element. The allowed values for "**type**" attribute are as follows: • bindingKey Indicates the UDDI key that is referenced by "uddiKey" is of type "bindingKey". • tModelKey Indicates the UDDI key that is referenced by "uddiKey" is of type "tModelKey".	Mandatory
inquiryURL	"**inquiryURL**" is a child element of "uddiReference" element. Using this element, the user can specify the inquiry URL that can be used to retrieve the UDDI bindingTemplate or tModel associated with the uddiKey element.	Mandatory

4.4 Sample Configuration Files

This section illustrates the sample configuration files for usage of WS-I Analyzer Tool for three different scenarios. The sample configuration files for following three different scenarios are explained in this section.

- Sample configuration file that uses direct WSDL reference.
- Sample configuration file that uses <serviceLocation> element.
- Sample configuration file that uses UDDI reference.

4.4.1 Sample configuration file that uses direct WSDL reference

The sample WS-I Analyzer tool configuration file that directly references the WSDL document as a local disk file appears as below:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsi-analyzerConfig:configuration name="Sample Basic Profile Analyzer Configuration"
   xmlns:wsi-analyzerConfig=”http://www.ws-i.org/testing/2004/07/analyzerConfig”/>

  <wsi- analyzerConfig:description xml:lang="en">
   This file contains a sample of the configuration file for
   the Basic Profile Analyzer, which can be used with the
   other sample files.
  </wsi- analyzerConfig:description>

  <wsi-analyzerConfig:verbose>false</wsi-analyzerConfig:verbose>
  <wsi-analyzerConfig:assertionResults type="all" messageEntry="false"
     failureMessage="true"/>
  <wsi-analyzerConfig:reportFile replace="true" location="report.xml">
        <wsi- analyzerConfig:addStyleSheet href="../common/xsl/report.xsl"/>
  </wsi-analyzerConfig:reportFile>
  <wsi-analyzerConfig:testAssertionsFile>
        ../common/profiles/BasicProfileTestAssertions.xml
  </wsi-analyzerConfig:testAssertionsFile>
  <wsi-analyzerConfig:logFile correlationType="endpoint">
        traceLog.xml
  </wsi-analyzerConfig:logFile>
  <wsi-analyzerConfig:wsdlReference>
        <wsi-analyzerConfig:wsdlElement type="port"
           parentElementName="RetailerService"
           namespace="http://.../RetailerService.wsdl">
                LocalIBMRetailerPort
        </wsi-analyzerConfig:wsdlElement>
   <wsi-analyzerConfig:wsdlURI>
         ../common/samples/RetailerService.wsdl
   </wsi-analyzerConfig:wsdlURI>
  </wsi-analyzerConfig:wsdlReference>
</wsi-analyzerConfig:configuration>
```

Figure 4 – Example of Configuration file Using Direct WSDL Reference.

The sample configuration file listed in Fig. #4 instructs the WS-I Analyzer tool to perform the following tasks:

1. Run without displaying diagnostic information (wsi-analyzerConfig:verbose value is **false**).
2. List all assertion results in the conformance report, as opposed to listing only a subset of those(assertionResults type="**all**").
3. Exclude log entries from the conformance report (messageEntry="**false**). The conformance of each log entry will still be reported, but the full content of the log entry will not appear.
4. Include failure messages for each test assertion in the report (failureMessage="**true**").

5. Write the report file to “report.xml” (location="**report.xml**”), and if the report file with same name already exists then replace it (replace="**true**").
6. Write the log file to “tracelog.xml” (wsi-analyzerConfig:logFile value is **traceLog.xml**).
7. Correlate messages to Web services based on the endpoint (correlationType="**endpoint**”).
8. The conformance target is a single port: (wsdlElement type="**port**" , parentElementName="**RetailerService**")
9. Refer to the WSDL “samples/retailer.wsdl” file for Web Services description (wsdlElement namespace="http://.../Retailer.wsdl”).

4.4.2 Sample configuration file that uses serviceLocation element

The configuration listed in Fig. #5 describes the sample WS-I Analyzer tool configuration file that user the serviceLocation element to specify the web service endpoint location, if this information is not specified in the WSDL document.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsi-analyzerConfig:configuration name="Sample Basic Profile Analyzer Configuration"
    xmlns:wsi-analyzerConfig=”http://www.ws-i.org/testing/2004/07/analyzerConfig”/>

  <wsi-analyzerConfig:description xml:lang="en">
    This file contains a sample of the configuration file for
    the Basic Profile Analyzer, which can be used with the
    other sample files.
  </wsi-analyzerConfig:description>

  <wsi-analyzerConfig:verbose>false</wsi-analyzerConfig:verbose>
  <wsi-analyzerConfig:assertionResults type="all" messageEntry="false"
failureMessage="true"/>
  <wsi-analyzerConfig:reportFile replace="true" location="report.xml">
            <wsi-analyzerConfig:addStyleSheet href="../common/xsl/report.xsl"/>
  </wsi-analyzerConfig:reportFile>
  <wsi-analyzerConfig:testAssertionsFile>
            ../common/profiles/BasicProfileTestAssertions.xml
  </wsi-analyzerConfig:testAssertionsFile>
  <wsi-analyzerConfig:logFile correlationType="endpoint">
            traceLog.xml
  </wsi-analyzerConfig:logFile>
  <wsi-analyzerConfig:wsdlReference>
            <wsi-analyzerConfig:wsdlElement type="binding"
                                    namespace="http://www.ws-
i.org/SampleApplications/SupplyChainManagement/2002-08/Retailer.wsdl">
                        RetailerSoapBinding
            </wsi-analyzerConfig:wsdlElement>
            <wsi-analyzerConfig:wsdlURI>
              http://www.ws-i.org/SampleApplications/SupplyChainManagement/2002-08/Retailer.wsdl
           </wsi-analyzerConfig:wsdlURI>
           <wsi-analyzerConfig:serviceLocation>
                      http://tempuri.org/services/retailerService
           </wsi-analyzerConfig:serviceLocation>
  </wsi-analyzerConfig:wsdlReference>
</wsi-analyzerConfig:configuration>
```

Figure 5 – Example of Configuration file Using Service Location

The sample configuration file listed in Fig. #5 instructs the WS-I analyzer tool to perform the following tasks:

1. The conformance target is a binding element. (<wsdlElement type="**binding**"> RetailerSoapBinding< wsdlElement>)
2. Refer to the WSDL specified using HTTP URL for Web Services description (<wsdlURI>http://www.ws-i.org/SampleApplications/SupplyChainManagement/2002-08/Retailer.wsdl</wsdlURI >).
3. Refer to http://tempuri.org/services/retailerService as service endpoint location. (<serviceLocation>http://tempuri.org/services/retailerService</ serviceLocation>). When the service location is not defined in a WSDL document, but this information is required by the WS-I Analyzer tool. When this situation occurs, the <wsdlElement> element should reference a WSDL binding and “serviceLocation” element should contain the service endpoint location.

4.4.3 Sample configuration file that uses UDDI reference

The configuration listed in Fig. #6 describes the sample WS-I Analyzer tool configuration file that uses the UDDI reference.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsi-analyzerConfig:configuration name="Sample Basic Profile Analyzer Configuration"
   xmlns:wsi-analyzerConfig= “http://www.ws-i.org/testing/2004/07/analyzerConfig” />

  <wsi-analyzerConfig:description xml:lang="en">
    This file contains a sample of the configuration file for
    the Basic Profile Analyzer, which can be used with the
    other sample files.
  </wsi-analyzerConfig:description>

  <wsi-analyzerConfig:verbose>false</wsi-analyzerConfig:verbose>
  <wsi-analyzerConfig:assertionResults type="all" messageEntry="false" failureMessage="true"/>
  <wsi-analyzerConfig:reportFile replace="true" location="report.xml">
    <wsi-analyzerConfig:addStyleSheet href="../common/xsl/report.xsl"/>
  </wsi-analyzerConfig:reportFile>
  <wsi-analyzerConfig:testAssertionsFile>
    ../common/profiles/BasicProfileTestAssertions.xml
  </wsi-analyzerConfig:testAssertionsFile>
  <wsi-analyzerConfig:logFile correlationType="endpoint">
    traceLog.xml
  </wsi-analyzerConfig:logFile>
  <wsi-analyzerConfig:uddiReference>
    <wsi-analyzerConfig:uddiKey type="bindingKey"> 22eb4f00-0ef2-11d7-a725-000629dc0a53</wsi-
analyzerConfig:uddiKey>
    < wsi-analyzerConfig:inquiryURL >
      http://uddi.ibm.com/ubr/inquiryapi
    </wsi-analyzerConfig:inquiryURL>
  </wsi-analyzerConfig:uddiReference>
</wsi-analyzerConfig:configuration>
```

Figure 6 – Example of Analyzer Configuration File Using UDDI Reference.

The sample configuration file listed in Fig. #6 instructs the WS-I analyzer tool to perform the following tasks:

1. The conformance target is UDDI reference and process the UDDI related test assertions.
2. Refer to the UDDI key of type "bindingKey" for Web Services discovery. (<uddiKey type="**bindingKey**"> **22eb4f00-0ef2-11d7-a725-000629dc0a53**</wsi-analyzerConfig:uddiKey >).
3. Refer to URL - **http://uddi.ibm.com/ubr/inquiryapi** for Web Services discovery. (<inquiryURL > **http://uddi.ibm.com/ubr/inquiryapi** </ inquiryURL >).Running the WS-I Analyzer Tool

4.5 WS-I Analyzer Tool Command Line Syntax

Apart from configuration file, the user can also use the command line options for WS-I Analyzer tool. All command line options override the options that are specified in the configuration file. Both – C# and Java – version of WS-I Analyzer tools supports –config, -verbose and –testAssertionFile options, while Java version of WS-I Analyzer tool support additional options.

The simplified command line syntax to invoke the WS-I Analyzer tool is as follows:

```
Analyzer -config <configFile> [-verbose true|false] [-testAssertionFile <testAssertionFile>]
```

4.5.1 C# WS-I Analyzer Tool Command Line Options

The command line options for C# version of WS-I Analyzer tool are described in detail in the following table:

Sr No	Option	Description
1	**-config**	The user can specify the configuration file to be used by the WS-I Analyzer tool
2	**-verbose**	Display diagnostic messages on the console.
3	**-testAssertionFile**	The user can specify the Test Assertions Document (TAD) file to be used by the WS-I Analyzer tool

4.5.2 Java WS-I Analyzer Tool Command Line Options

The command line options for Java version of WS-I Analyzer tool are described in detail in the following table:

Sr No	Option	Description
1	**-config** or **–c**	The user can specify the configuration file to be used by the WS-I Analyzer tool.
2	**-verbose** or **–v**	Display diagnostic messages on the console.
3	**-testAssertionFile** or **-t**	The user can specify the Test Assertions Document (TAD) file to be used by the WS-I Analyzer tool.
4	**-assertionResults** or **-a**	The user can specify the appropriate values for this option to indicate the type of assertion results that should be listed in the conformance report.
5	**-messageEntry** or **-M**	Using this option the user can specify whether the message entries should be included in the report file.
6	**-assertionDescription** or **-A**	Using this option the user can specify whether the description of each test assertion should be included in the report file.
7	**-failureMessage** or **-F**	Using this option the user can specify whether the failure message defined for each test assertion should be included in the report file.
8	**-failureDetail** or **-D**	Using this option the user can specify whether the failure detail message defined for each test assertion should be included in the report file.
9	**-logFile** or **-l**	Using this option the user can specify the location of the log file that contains the SOAP messages that need to be processed by the WS-I Analyzer tool.

Sr No	Option	Description
10	**-reportFile** or **-r**	The user can specify the appropriate values for this option to provide the location of the report file
11	**-replace** or **-R**	Using this option the user can specify whether the existing report file can be overwritten.
12	**-correlationType** or **-C**	The user can specify the appropriate value for this option to define what information should be used to match the messages from the log file with the web services that are being tested.
13	**-wsdlElement** or **-W**	Using this option, the user can specify the reference to the WSDL element that should be analyzed.
14	**-serviceLocation** or **-S**	Using this option, the user can specify the service location information for the element that is referenced by the corresponding "wsdlElement"
15	**-wsdlURI** or **-W**	Using this option, the user can specify the location and the filename of the WSDL file for the Web Services under test.
16	**-uddiKeyType** or **-K**	The user can specify the appropriate value for this option to indicate the type of the UDDI key referred to by "uddiKey" element.
17	**-uddiKey** or **-k**	Using this option, the user can specify the value of the UDDI Key
18	**-inquiryURL** or **-i**	Using this option, the user can specify the inquiry URL that can be used to retrieve the UDDI bindingTemplate or tModel associated with the uddiKey element.

4.5.3 Executing the C# Version of WS-I Analyzer Tool

To run the C# version of WS-I Analyzer tool, change directory to the <wsi-test-tool-home>\cs\bin folder and execute the following command:

```
Analyzer [-config < configFilename >]
```

Example:

```
cd c:\wsi-test-tools\cs\bin
Analyzer -config ..\samples\analyzerConfig.xml
```

Note: If no configuration file is defined, the analyzer will default to analyzerConfig.xml.

4.5.4 Executing the Java Version of WS-I Analyzer Tool

To run the Java version of WS-I Analyzer tool, change directory to the <wsi-test-tool-home>\java\bin folder and execute the following command:

```
Analyzer -config < configFilename >
```

Example:

```
cd C:\wsi-test-tools\java\bin
Analyzer -config ..\samples\analyzerConfig.xml
```

Note: there is no default configuration file for the Java version.

5 The Test Assertions Document (TAD)

5.1 *Test Assertion representation*

A test assertion is an encoding of a profile requirement defined in the profile document. It can represent part of a requirement, a single requirement, or more than one requirement. The set of test assertions derived from a profile requirement is scripted into an XML document called the Test Assertion Document (TAD). This document is used by the WS-I Analyzer Tool as input and it determines the set of test procedures that will be activated.

An example of Test Assertion XML element is shown in Figure 7 below:

```
<testAssertion id="BP1306" entryType="responseEnvelope" type="required"
enabled="true">
    <context>For a candidate response message containing a soap:Fault element</context>
    <assertionDescription>The soap:Fault element does not have children other than
soap:faultcode, soap:faultstring, soap:faultactor or soap:detail.</assertionDescription>
    <failureMessage>One or more soap:Fault children elements are not standard, i.e. is
neither soap:faultcode, soap:faultstring, soap:faultactor nor soap:detail.</failureMessage>
    <failureDetailDescription>SOAP message</failureDetailDescription>
    <additionalEntryTypeList/>
    <prereqList>
        <testAssertionID>BP1701</testAssertionID>
    </prereqList>
    <referenceList>
        <reference profileID="BP11">R1000</reference>
    </referenceList>
    <comments/>
</testAssertion>
```

Figure 7 – Example of Test Assertion.

5.2 *TAD Terminologies and Definitions*

The following terms are used when describing a test assertion. Each of these terms is always related to a particular test assertion:

- **Artifact**: General term used to designate the material used as input to the analyzer. See Chapter 6 (6.1) for a further definition of Artifact.
- **Entry Type:** Each artifact type can be further specialized into sub-types, called entry types. See Chapter 6 (6.1) for a further definition of Entry Type.
- **Test Assertion:** A test assertion is a translation of a WS-I profile requirement into a statement verifiable by the WS-I Analyzer. Each test assertion is defined by a testAssertion XML element in the Test Assertion Description document. Each test assertion usually relates to a specific artifact (though it may correlate several artifacts, a test assertion will always relate to one of these as its primary artifact, e.g. a

"WSDL" test assertion, or a "message" test assertion.) This artifact type is identified by the "type" attribute of the "artifact" XML element of which the test assertion is a child.

- **Informational Assertion:** The profiles have identified some material as being out of scope. Though these materials may cause problems with interoperability they are not in the strictest sense forbidden in order to achieve profile conformance. Instead, these materials are listed as Extensibility Points by the profiles. Extensibility points are addressed by Informational Assertions in the TAD. An Informational Assertion can only have two outcomes produced by the Analyzer in the report file: "passed", or "notApplicable". In the case of "passed", an extensibility point was encountered by the analyzer and its use noted. In the case of "notApplicable" no such extensibility point was encountered. Neither case has an overall effect on the report summary which can be "passed" or "failed".
- **Primary Entry (Type):** A Test Assertion will always target instances of a specific entry type, for example, a request message or a WSDL port binding. It is identified by the attribute "entryType" of the XML element "testAssertion". (Note that a TA may need to correlate other entries, e.g. may need to access WSDL definitions in order to verify messages captured on the wire). The primary entry type for a test assertion is the entry type that is the main object of the test assertion, e.g. a message request or response. Each test assertion will generate a single conformance statement within an analyzer report. The conformance statement will only concern the primary entry even though the reported errors may provide details on the associated non-primary entries. This means there will be as many pass-or-fail results for this test assertion, as there are qualified entries (instances of the primary entry type) in the input material to the Analyzer.
- **Secondary Entry (Type)**: A secondary entry is any entry that is required in addition to the primary entry, in order to process a test assertion, i.e. needs to be correlated with the primary entry. For example, the primary entry may be a request message as captured on the wire, and a secondary entry may be the message parts description in WSDL that relates to this wire message. The list of secondary entry types (if any) is specified in the XML element "additionalEntryTypeList".
- **Qualified Instance**: Artifact entry that matches the context of a test assertion.
- **Context**: Intuitively, the context of a test assertion defines which artifact material will qualify for a test assertion. A context in a test assertion is a pre-condition that entries of one or more entry types must satisfy in order for the analyzer to verify the test assertion over these entries. When more than one entry type is defined in a test assertion (primary and secondary types), the context normally defines how to correlate the entries instances of these types, so that the right secondary entries will be associated with the primary entry. The context should also single out the primary entry type. It is described by the XML element: "Context" in the TAD.
- **Assertion Description**: The assertion description for a test assertion is the actual profile requirement to which a qualified entry is expected to conform. It is described by the XML element: "assertionDescription" in the TAD.

- **Pre-requisites:** A test assertion may refer to pre-requisite test assertions. The intuitive meaning of pre-requisites is that when verifying the test assertion over an entry, in case this entry (or related secondary entries) did not satisfy the pre-requisite test assertions for this test assertion, then the outcome of the test assertion verification would be meaningless. Consequently, a test assertion should never be evaluated for an entry, if for the entry (or related secondary entries), the relevant pre-requisite test assertions failed. However, to enhance the usability of existing assertions as prerequisites, test assertion can be evaluated for an entry, if any of the prerequisite test assertions for that entry (or any related secondary entries) are notApplicable. For example, a test assertion with an entry type of “responseMessage” can be a prerequisite for a test assertion with entry type “anyMessage”. If the actual entry is a request message then the pre-requisite test assertion is just ignored. Pre-requisite test assertion Ids are listed in the XML element: “prereqList” in the TAD.
- **Referenced Profile Requirements**: A test assertion (TA) normally has references to one or more profile requirements (“reference” elements), labeled as Rxxxx in the Profile definition. Each profile requirement that is referenced by a TA falls into one of two categories or “roles”:
  - **Target profile requirement**: This type of profile requirement is verified by the TA. If the conformance report shows that some entry material fails this assertion, it means that this entry does not conform to some of the target requirements (the details of the error message will distinguish which one, in case there are several.)
  - **Collateral profile requirement**: This type of profile requirement is NOT verified by the TA, but is considered as collateral to the test assertion. A collateral profile requirement indicates conditions that must be taken into account, when verifying a target requirement. Some collateral profile requirements (e.g. such as MAY and SHOULD types of requirements), may actually never be verified by (i.e. targets of) any TA. Such requirements simply represent some possible situations and options that must be taken into account when verifying other requirements. Collateral requirements are also different from “pre-requisites” as defined here, in that they may concern other material than the primary entry for the TA (e.g. a secondary entry). If case a collateral requirement for TA 1 is the target of another test assertion TA 2, and if TA 2 fails on related artifacts, then the outcome of TA 1 should not be relied upon: in case of failure of TA 1, the artifact failing TA 2 should be investigated as a possible indirect cause of failure for TA 1.

5.3 How the Test Assertions are processed

This section provides additional information on how test assertions are processed.

- Each one of the Analyzer input options (e.g. as defined in the analyzer configuration file) maps to an artifact type: description, messages, or discovery. The Analyzer will process all the target material (entries) of a type of artifact, before processing entries of the next type of artifact. The processing order is: (1) discovery, (2) description, (3) messages.

- The entries in the input material, are processed in an ordered way, e.g. message entries are processed in the order they appear in the message log file. Each entry of an artifact type will be analyzed in sequence, which means the analysis of an entry will be complete (a report on the profile-conformance of the entry will be appended in the report document) before analyzing the next entry.
- When an entry is analyzed, all the test assertions which have a corresponding primary entry type, will be considered for verification. Only those for which (1) the primary entry has also satisfied the pre-requisite assertions, (2) the primary entry satisfies the Context, will be processed. For each processed TA, there will be a report item in the conformance report, for this entry.

When a test assertion is processed over an entry, it will complete with one of the following results:

- **passed**
  The test assertion completed the verification on the entry without detecting any errors.
- **failed**
  The entry in input failed the test assertion.
- **warning**
  The entry in input failed the test assertion, but the test assertion indicated that it was "recommended", not "required". This type of failure will not affect the overall conformance result.
- **prereqFailed**
  The test assertion was not processed because a prerequisite test assertion failed
- **notApplicable**
  The entry did not qualify for this test assertion, which means that although the entry was of the type of artifact relevant to this test assertion, it did not match the assertion context or it failed a prerequisite test assertion. In both cases, the test assertion is not relevant to this entry.

When summarizing the overall result of a test assertion over a set of entries, all individual entry results will be counted for each of the above outcomes, in the conformance report. Another case may occur, where no entry of the expected type was provided for this test assertion (e.g. no WSDL file was provided to the analyzer for a test assertion relating to WSDL.)

- **missingInput**
  The test assertion was not processed due to a lack of entries of the expected type. (i.e. the input required to process the test assertion was missing). This is also the case when there was no qualified entry for the test assertion, in the specified artifacts (e.g. a WSDL document does not contain a <types> element).

5.4 *Viewing the HTML version of Test Assertion Document (TAD)*

The XSL style sheet referred to by the XML processing instruction embedded at the beginning of the TAD file, controls the rendering of the TAD in HTML format. This XML processing instruction in XML log file appears as follows:

```
<?xml-stylesheet type="text/xsl" href="..\xsl\assertions.xsl"?>
```

The XSL style sheet referred to by “href” attribute should be accessible from the directory where the TAD file is located. The common.xsl file referred to by assertions.xsl file should also be appropriately accessible in order to render the TAD file in HTML format.

When using IE to view the HTML version of the TAD, it is preferred to use IE 5.5, or later version of IE. In order to view the TAD in HTML format using IE, just open the TAD file using IE.

6 The Profile Conformance Report

The analyzer tool produces one output file, the Conformance Report. This file contains the conformance report for the set of artifacts that were produced by a Web service. This report contains the conformance test results for the material provided as input to the Analyzer (WSDL, message log file), with regard to the set of assertions that were to be verified (in a Test Assertion Document, for example BasicProfile_1.1_TAD.xml). The conformance report also details the conformance level for each test assertion that was processed, and may list detailed information for any error that was encountered. The report also contains a summary of the test assertions results. This summary will indicate if the artifacts related to the target Web service passed or failed the profile conformance test.

The conformance report as produced by the Analyzer is an XML file. An XSL transform is provided for HTML rendering, and also for computing different views of the raw data in XML format, such as test assertion summaries, for each class of artifact.
The next section describes first the XML format of the report.
The following section comments on the HTML rendering of the report, which is the one user, may want to consult, as a more readable document for conformance assessment.

6.1 Definitions

These definitions relate to terms that appear in the conformance report, and also help to describe how the analyzer is processing test data.

- **Artifact**: General term used to designate the material used as input to the analyzer. For the Basic Profile, there are four types of artifacts, which correspond to the different inputs provided to the Analyzer:
  - envelope: designated to the SOAP 1.1 structure that transmits the message in a *message log file*.
  - messages: designated to the entries in the *message log file*.
  - description: designated to the WSDL files or parts of these.
  - discovery: designated to any material represented in UDDI, not including WSDL items
- **Entry type:** Each artifact type can be further specialized into sub-types, called entry types.
  - “requestMessage” , “responseMessage” and “anyMessage” are entry types associated with the “messages” artifact
  - “port”, “binding”, “portType”, “definitions”, “import” , “types” , “message “ are entry types associated with the “description” artifact
  - “bindingTemplate”, “tModel” are entry types associated with the “discovery” artifact

  - "requestEnvelope", "responseEnvelope" and "anyEnvelope" are entry types associated with the "envelope" artifact.

- **Entry**: An entry is an instance of an entry type, for example an HTTP request (for "requestMessage" type), or a part of a WSDL file that describes a port binding (for "binding" type).

6.2 *Elements of a Report*

The following table defines each of the elements that can be used in the conformance report file. Attributes are shown using Xpath notation (ex. report[@name] for the "name" attribute).

Element	Description	Usage
report	The root element for the profile conformance report file.	Mandatory.
report[@**name**]	The name of the conformance report	Optional
Report[@**timestamp**]	The date and time indicating when the report was generated.	Mandatory
analyzer	This element contains information about the specific implementation of the analyzer tool, and the options that were used to test a Web service for conformance to a profile.	Mandatory
analyzer[@**version**]	The version number for the implementation of the tool. This value contains a version and release number. It may also contain a major and minor number.	Mandatory
analyzer[@**releaseDate**]	The date the tool was released.	Mandatory
Implementer	The organization that implemented the analyzer tool. **Note:** The URI value for the location attribute , if present in the report, contains an indication of the analyzer version. (date or version number). Here are two examples of how this may appear in an analyzer implementation: http://hostname/2003/03/analyzer http://hostname/1.0/analyzer	Mandatory
implementer[@**name**]	The name of the organization that implemented the tool.	Optional
implementer[@**location**]	Web site where you can get more information on the implementation of the tool.	Optional
environment	The environment that was used to run the analyzer	Mandatory

	tool.	
runtime	The runtime that was used by the analyzer.	Mandatory
runtime[@**name**]	Runtime name	Mandatory
runtime[@**version**]	Runtime version	Mandatory
operatingSystem	The operating system where the analyzer tools was run.	Mandatory
operatingSystem[@**name**]	Operating system name	Mandatory
operatingSystem[@**version**]	Operating system version	Mandatory
xmlParser	The XML parser that was used when running the analyzer.	Mandatory
xmlParser[@**name**]	XML parser name	Mandatory
xmlParser[@**version**]	XML parser version	Mandatory
configuration	The configuration options which were specified when the analyzer was run. Refer to the section on the configuration file for a description of this element and its contents.	Mandatory
artifact	This element contains a reference to one of the artifacts that is listed in the test assertion document.	Optional
artifact[@**type**]	The type of artifact that is being analyzed. The value of the attribute always matches one of the valid artifact types defined in the test assertion document.	Mandatory
artifactReference	This element contains artifact reference information. For example, if the artifact is "message", then it will contain the timestamp from the message log file and it may contain the contents of the first <wsi-monConfig:comment> element that appears in the monitor configuration section of the log file if it is present.	Mandatory
artifactReference[@**timestamp**]	The timestamp from the message log file or the date and time for the WSDL file.	Mandatory
comment	The comment element that is the first child of the configuration element in the message log file.	Optional
entry	This element contains a reference to an instance of a type of entry that was analyzed. **Note:** The valid values for the type attribute are given in quotes in 6.1 Definitions above.	Mandatory
entry[@**type**]	The type of entry for the test assertion.	Optional
entry[@**referenceID**]	This attribute is optional. When it is specified, it includes a unique identifier for an instance of the type of entry (an example would be an identifier for a specific entry in a message log file). Depending on the type of artifact, the referenceID has different	Optional

	meaning (see later in this section)	
messageEntry	This element contains a reference to the log entry which was the target of a test assertion.	Optional
assertionResult	This element contains the result for a single execution of a test assertion for an entry.	Mandatory
assertionResult[@**id**]	Test assertion identifier. This value matches the value that is listed in the profile definition document.	Mandatory
assertionResult[@**result**]	This attribute contains the result from the execution of the test assertion. The values for the result attribute have the following meaning: • **passed** The test assertion completed its check without detecting any errors. • **failed** The test assertion detected an error. A description of the error appears into the <failureMessage> sub-element. • **warning** The test assertion failed, but the type attribute for the test assertion indicated that it was "recommended", not "required". • **notApplicable** The test assertion was not processed because it did not match the assertion context or a prerequisite test assertion failed. • **prereqFailed** The test assertion was not processed because a prerequisite test assertion failed **missingInput** The test assertion was not processed due to a lack of entries of the expected type. (i.e. the input required to process the test assertion was missing). This is also the case when there was no qualified entry for the test assertion, in the specified artifacts (e.g. a WSDL document does not contain a <types> element).	Mandatory
additionalEntryList	This element contains a list of additionalEntry elements.	Optional
additionalEntry	This element contains a reference to entries in addition to the primary entry defined within the <entry> element which were needed to process a test assertion.	Optional

	Note: The values for the type attribute have the same meaning as those for the <entry> element.	
additionalEntry[@**type**]	The type of entry for the test assertion.	Optional
additionalEntry[@**referenceID**]	This attribute is optional. When it is specified, it contains a unique identifier for an instance of the type of entry (an example would be an identifier for an entry in a log file).	Optional
assertionDescription	The assertion description for the test assertion.	Optional
failureMessage	The failure message that is defined for the test assertion.	Optional
failureDetail	An optional failure detail message which is specific to the implementation of the analyzer tool. As an example, this element may contain a failure detail message (or set of messages) from an implementation specific XML parser.	Optional
failureDetail[@**referenceType**]	The type of entity that caused all or part of the test assertion failure. This attribute is optional.	Optional
failureDetail[@**referenceID**]	The identifier for the entity that caused all or part of the failure. This attribute is optional.	Optional
summary	This element is the container for the conformance report summary.	Optional
summary[@**result**]	The values for the result attribute have the following meaning: • **passed** The result attribute will contain a value of "passed" only if all of the processed test assertions were successful. The result value will be "passed" even when some test assertions are not processed because the input options indicated that they should be ignored. • **failed** If at least one individual execution of a test assertion failed, then this attribute will have a value of "failed".	Mandatory
analyzerFailure	When a failure occurs that causes the analyzer tool to terminate before it has processed all of the test assertions, this element is used to indicate the source of the error. This element contains at least one <failureDetail> elements as a sub-element. The < failureDetail > element indicates the source of the error, and contains instructions on how to correct the error.	Optional

6.3 *Example of Conformance Report in XML Format*

The following figure (Figure 8) contains an example of a profile conformance report XML file, as produced by the Analyzer.

Note: This example does not contain a complete conformance report. Most of the test assertion results have been left out.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="../common/xsl/report.xsl" type="text/xsl" ?>
<report name="WS-I Basic Profile Conformance Draft Report. This is a prerelease
version and no statement can be made from this report on WS-I conformance."
   timestamp="2004-11-10T09:11:48.455"
   xmlns="http://www.ws-i.org/testing/2004/07/report/"
   xmlns:wsi-report="http://www.ws-i.org/testing/2004/07/report/"
   xmlns:wsi-log="http://www.ws-i.org/testing/2003/03/log/"
   xmlns:wsi-analyzerConfig="http://www.ws-i.org/testing/2004/07/analyzerConfig/"
   xmlns:wsi-monConfig="http://www.ws-i.org/testing/2003/03/monitorConfig/"
   xmlns:wsi-assertions="http://www.ws-i.org/testing/2004/07/assertions/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<analyzer version="1.1" releaseDate="2004-11-10">
   <implementer name="Web Services Interoperability Organization"
location="http://www.ws-i.org/testing/2003/03/Analyzer.html"/>
   <environment>
     <runtime name="Java(TM) 2 Runtime Environment, Standard Edition"
version="1.4.2"/>
     <operatingSystem name="Windows XP" version="5.1"/>
     <xmlParser name="Apache Xerces" version="XML4J 4.3.3"/>
   </environment>
   <wsi-analyzerConfig:configuration>
     <wsi-analyzerConfig:verbose>false</wsi-analyzerConfig:verbose>
     <wsi-analyzerConfig:assertionResults type="all" messageEntry="true"
         assertionDescription="false" failureMessage="true" failureDetail="true"/>
       <wsi-analyzerConfig:reportFile replace="true" location="report.xml">
         <wsi-analyzerConfig:addStyleSheet href="../common/xsl/report.xsl"
type="text/xsl" />
       </wsi-analyzerConfig:reportFile>
       <wsi-analyzerConfig:testAssertionsFile>
         ../common/profiles/ SSBP10_BP11_TAD.xml
       </wsi-analyzerConfig:testAssertionsFile>
       <wsi-analyzerConfig:wsdlReference>
      <wsi-analyzerConfig:logFile>log.xml</wsi-analyzerConfig:logFile>
      <wsi-analyzerConfig:wsdlElement type="port" namespace="http://www.ws-
i.org/SampleApplications/SupplyChainManagement/2002-08/RetailerService.wsdl"
```

```
          parentElementName="RetailerService">LocalIBMRetailerPort</wsi-
analyzerConfig:wsdlElement>
          <wsi-analyzerConfig:wsdlURI>samples/RetailerService.wsdl</wsi-
analyzerConfig:wsdlURI>
        </wsi-analyzerConfig:wsdlReference>
    </wsi-analyzerConfig:configuration>
  </analyzer>

  <artifact type="discovery">
    <entry type="[discovery]" >
      <assertionResult id="BP3001" result="missingInput"/>
      <assertionResult id="BP3002" result="missingInput"/>
      <assertionResult id="BP3003" result="missingInput"/>
    </entry>
  </artifact>

<artifact type="description">
    <entry type="definitions" referenceID="file:samples/RetailerService.wsdl">
      <assertionResult id="BP2703" result="passed/">
    </entry>
    <entry type="definitions" referenceID="http://www.ws-
i.org/SampleApplications/SupplyChainManagement/2002-08/Retailer.wsdl">

       <assertionResult id="BP2703" result="passed"/>
    </entry>
    <entry type="definitions" referenceID="http://www.ws-
i.org/SampleApplications/SupplyChainManagement/2002-08/Configuration.wsdl">
      <assertionResult id="BP2703" result="passed"/>
    </entry>
    <entry type="binding" referenceID="http://www.ws-
i.org/SampleApplications/SupplyChainManagement/2002-
08/Retailer.wsdl:RetailerSoapBinding">
      <assertionResult id="BP2019" result="notApplicable"/>
      <assertionResult id="BP2012" result="notApplicable/">
      <assertionResult id="BP2020" result="passed"/>
      <assertionResult id="BP2021" result="passed"/>
      <assertionResult id="BP2022" result="passed"/>
      <assertionResult id="BP2404" result="passed"/>
      <assertionResult id="BP2013" result="passed"/>
      <assertionResult id="BP2017" result="passed"/>
      <assertionResult id="SSBP2402" result="passed"/>
    </entry>
    <entry type="portType" referenceID="http://www.ws-
i.org/SampleApplications/SupplyChainManagement/2002-
08/Retailer.wsdl:RetailerPortType">
      <assertionResult id="BP2010" result="passed"/>
    </entry>
    <entry type="operation" referenceID="getCatalog"
parentElementName="RetailerService">
      <assertionResult id="BP2208" result="passed/">
    </entry>
    <entry type="operation" referenceID="submitOrder"
parentElementName="RetailerService">
      <assertionResult id="BP2208" result="passed"/>
    </entry>
  </artifact>

<artifact type="message">
    <artifactReference timestamp="2004-11-09T16:06:03.605Z">
    <wsi-monConfig:comment>This configuration file is used to test the WS-I sample
applications.</wsi-monConfig:comment>
    </artifactReference>
    <entry type="requestMessage" referenceID="19">
  <wsi-log:messageEntry xsi:type="wsi-log:httpMessageEntry" ID="19"
conversationID="1" type="request" timestamp="2004-11-09T14:20:51.234Z">
    <wsi-log:messageContent>[…message content…]</wsi-log:messageContent>
    <wsi-log:senderHostAndPort>127.0.0.1:3666</wsi-log:senderHostAndPort>
    <wsi-log:receiverHostAndPort>localhost:6080</wsi-log:receiverHostAndPort>
    <wsi-log:httpHeaders>POST /Retailer/services/Retailer HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.0
Host: localhost:6080
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 3446

</wsi-log:httpHeaders>
  </wsi-log:messageEntry>
      <assertionResult id="BP1002" result="passed"/>
      <assertionResult id="BP1001" result="warning">
        <failureMessage xml:lang="en">Message is not sent using
HTTP/1.1.</failureMessage>
        <failureDetail xml:lang="en">POST /Retailer/services/Retailer HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.0
Host: localhost:6080
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: &quot;&quot;
Content-Length: 3446
        </failureDetail>
      </assertionResult>
      <assertionResult id="SSBP5100" result="passed"/>
    </entry>
      <!-- Other message entry results go here. -->
  </artifact>

  <artifact type="envelope">
    <artifactReference timestamp="2004-11-09T16:06:03.605Z">
    <wsi-monConfig:comment>This configuration file is used to test the WS-I sample
applications.</wsi-monConfig:comment>
    </artifactReference>
    <entry type="requestEnvelope" referenceID="19">
      <wsi-log:messageEntry xsi:type="wsi-log:httpMessageEntry" ID="19"
        conversationID="1" type="request" timestamp="2004-11-09T14:20:51.234Z">
        <wsi-log:messageContent>[…message content…]</wsi-log:messageContent>
      </wsi-log:messageEntry>
      <assertionResult id="BP1201" result="passed">
      <assertionResult id="BP1601" result="passed"/>
      <assertionResult id="BP1701" result="passed"/>
      <assertionResult id="BP1308" result="passed"/>
      <assertionResult id="BP4109" result="notApplicable"/>
      <assertionResult id="SSBP1601" result="passed"/>
      <assertionResult id="SSBP9704" result="passed"/>
    </entry>
    <!-- Other envelope entry results go here. -->
  </artifact>
  <summary result="passed">
  </summary>
</report>
```

Figure 8. Example of Profile Conformance Report.

How to interpret the ReferenceID attribute in the Entry element of a report

- **discovery**
  For an entry type of **bindingTemplate**, the referenceID value is the bindingTemplate key. If the entry type is **tModel**, then the referenceID is a tModel key.
- **description**
  For this artifact, the referenceID value will vary based on the entry type value for the **port** and **operation** entry types, the referenceID value is the value of the "name" attribute on the element for the entry instance. For the **binding**, **portType**, and **message** entry types, the referenceID value is the QName for the element associated with the entry instance. When the entry type is **definitions**, the referenceID value is the location of the WSDL document. For an entry type of **import**, the referenceID value is the value from the namespace attribute on the <import> element. For the

types entry type, the referenceID value is the location of the WSDL document with "-Types" appended to it.

- **message**
  The referenceID value is always the entry identifier for the message log entry.
- **envelope**
  The referenceID value is the entry identifier for the message log entry.

Failure Detail Message Content

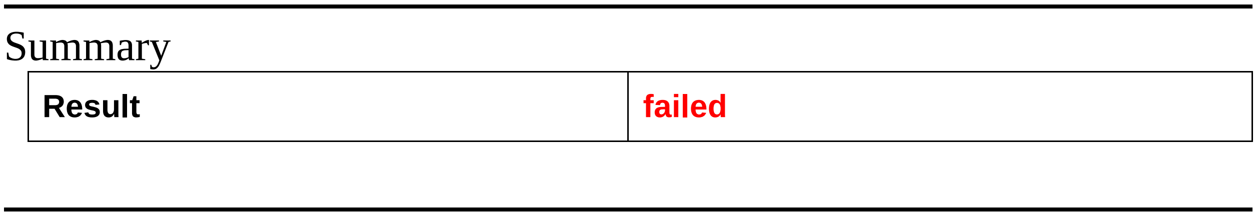
The test assertions in the Test Assertion Document may define the type of content for the <failureDetail> element. The actual content of the <failureDetail> element is implementation specific. Also, when a test assertion has one or more additional entry types and the entry instance is not available when the analyzer is running, then the <failureDetail> element will contain the following text: "Additional entry missing".

6.4 Conformance Report In HTML Format

The following samples are extracted from a sample conformance report after HTML rendering, as produced by the XSL transform.

Note: When using IE to view your report, you should use IE 5.5, or preferably IE 6.0. Simply opening your XML file with IE should be sufficient, as the xsl file (report.xsl) is automatically referred. The common.xsl file must also be accessible.

The general result of the analysis is provided at the beginning of the report (Figure 9):



Summary

Result	**failed**

Figure 9 – Summary line of a conformance report.

The above example illustrates an overall summary result of a conformance test. The possible values are:

- **Passed**: The result of processing the set of test assertions enabled in the Analyzer (see the Test Assertion Document) was positive. This means that each entry (of any of the three artifact types) passed all the relevant test assertions. Another way to state this is: for each test assertion that was enabled in the Analyzer, all the relevant or applicable artifact entries did pass, or generated at most a warning.

- **Failed**: The result of processing the set of test assertions enabled in the Analyzer (see the Test Assertion Document) was negative. This means at least one entry failed one of the test assertions.

(see the previous section)

Links to each sub-section of the report are then provided in an artifact index (Figure 10):

Artifacts

discovery
description
message
envelope

Figure 10 – General Artifact Index.

The above is an index on each of the four sections of the report that relate to each of the artifact types. In case no entry has been provided for an artifact, the referred section will be empty. The following message will appear instead:
This artifact was not processed by the analyzer

Artifact: message

Artifact Reference:

Timestamp
2004-11-09T16:06:03.605Z

Assertion Result Summary:

Assertion ID	Passed	Failed	Prerequisite Failed	Warning	Not Applicable	Missing Input
BP1001	2	0	0	0	0	
BP1002	1	0	0	1	0	
BP1004	1	0	0	0	0	
BP1006	1	0	0	0	0	
BP1010	0	1	0	0	0	
BP1101	0	0	0	0	1	
BP1103	0	0	0	0	1	
BP1116	1	0	0	0	0	

BP4103	0	0	0	0	2	
BP4104	2	0	0	0	0	
BP4105	0	0	0	0	2	
BP4106	0	0	0	0	1	
BP4107	0	0	0	0	1	
SSBP1003	2	0	0	0	0	
SSBP5100	2	0	0	0	0	
SSBP5101	2	0	0	0	0	

Figure 11 – Test Assertion Summary Report.

The example in Figure 11 shows a test assertion summary for the "message" artifact. For each of the test assertions that were enabled, there is a line in the summary. Each column shows one possible outcome of the test assertion, and the number of message entries that generated such outcome (an entry can only generate one outcome). See the previous section for the exact meaning of each outcome. There is a color code for the test assertions:

- **Green**: At least one entry has passed the test assertion (column "Passed"), and no entry generated failures or warnings. Such a test assertion can be considered verified on the set of artifacts in input of the analyzer. (note there may be nonApplicable entries)
- **Orange**: Although no failure was generated, at least one entry generated a warning (column "Warning"). This indicates that a recommended profile feature was not observed on the entry.
- **Red**: At least one entry failed the test assertion (column "Failed"). This means a profile violation, regardless of how well other entries fared for this test assertion.
- **Blue**: All entries of the type of artifact targeted by this test assertion, were not qualified (column "not Applicable"), i.e. not individually relevant to this assertion. It means that the set of entries provided as input were not appropriate to test this profile feature. Unless it is clear that the Web Service under test will never exercise such a feature, a more comprehensive set of entries should be provided.

The test assertion summary table is followed by an index of all entries that have been verified.

The artifact section of the report then contains the details of these entries, what were the test assertions applied to them, and the detailed outcome. The following example in Figure 12 shows one such message entry, and the results for three test assertions:

Entry: 1

Reference ID	Type
1	requestMessage

Message Entry:

Conversation ID	2
Sender Host and Port	127.0.0.1:1806
Receiver Host and Port	localhost:6080
HTTP Headers	POST /LoggingFacility/services/LoggingFacility HTTP/1.0 Content-Type: text/xml; charset=utf-8 Accept: application/soap+xml, application/dime, multipart/related, text/* User-Agent: Axis/1.0 Host: localhost:6080 Cache-Control: no-cache Pragma: no-cache SOAPAction: "" Content-Length: 632

Message	<?xml version="1.0" encoding="UTF-8"?>
	<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
	 <soapenv:Body>
	  <logEventRequestElement xmlns="http://www.ws-i.org/SampleApplications/SupplyChainManagement/2002-08/LoggingFacility.xsd">
	   <DemoUserID>AUser-1-4</DemoUserID>
	   <ServiceID>Retailer.submitOrder</ERROR>
	   <EventID>UC1-5</EventID>
	   <EventDescription>Order placed by ABCD999999999EFG for 605008, 605004, 605003</EventDescription>
	  </logEventRequestElement>
	 </soapenv:Body>
	</soapenv:Envelope>

Assertion: BP1004

Result	**passed**

Assertion: BP1002

Result	**failed**
Failure Message	The message is not sent using HTTP/1.1 or HTTP/1.0.
Failure Detail	POST /WSITest/servlet/rpcrouter HTTP/0.9 Host: volodin:9080 Content-Type: text/xml; charset=utf-8 Content-Length: 626 SOAPAction: ""

	`Element Location:`
	`  lineNumber=56`

Assertion: BP1001

Result	**prereqFailed**
Prereq Failed List	**BP1002**

Figure 12 – Part of a Report: Detail of an Entry Analysis.

The message entry is number 1 in the log file (Reference ID), and is of type “requestMessage”. Details of the entry are provided, as they appear in the log file. A list of test assertion reports is then provided for this entry. In case of failure, the cause of failure is reported. If the user wants to have more details on the test assertion that was exercised (e.g. BP1002), then s/he will access the Test Assertion Document, which provides the details for each test assertion (both the XML and HTML versions are available in the tool package).

7 Frequently Asked Questions (FAQ)

Question: Can testing tools certify that a Web Service is conforming to the Profile?

Answer: The tools can only verify the conformance of Web Service *artifacts* that are produced during a testing session. Some artifacts belong to the definition of the Web Service (WSDL); some others result from the observable behavior of the Web Service at run-time. It is rather difficult to test all possible behaviors that a Web Service can exhibit, mostly because exercising these behaviors is application-dependent and requires an application-level understanding of the Web Service. For these reasons, the Testing WS-I working group has not attempted to provide certification criteria. Indeed, using certification criteria that are too general or incomplete will not guarantee interoperability for every use case, and therefore a certification stamp would have little meaning. Instead, the tools are intended to observe and verify the messages produced during an interaction, possibly in a real deployment environment (because the tools are non-intrusive). The tools can also be used at development time, to verify that Web Service definitions are profile-conforming. The testing tools are then an *indicator* of conformance of a Web Service to the Profiles selected, based on the artifacts produced. In turn, this is an indicator of interoperability with other business partners who also have tested as conforming to the Profiles.

Question: Can testing tools verify all the requirements of a Profile?

Answer: No. A few requirements of the various WS-I profiles cannot be easily tested, and have been left out for V1.0 of the tools. Such requirements fall into one of these categories:

- The profile requirement refers to an external specification document that is too complex to test, for an outcome that has been prioritized as low, given current resources. An example is the requirement on cookies which, when used, must conform to RFC2965.
- The requirement is not possible to test using the current test harness. For example, requirements about the HTTP code value when a request has been redirected.
- The requirement is about interpretative behavior of a Web Service consumer or instance, which exceeds the capability of the test harness, and would require more intrusive technology, or more knowledge of the WS application and semantics.

This is another reason why the tools should be defined more as an indicator of conformance, rather than as certification tools. However, by addressing requirements that concern the run-time interaction between a Web Service and another party, the tools provide a powerful indicator of the ability of this Web Service to interoperate with any external party known to also comply with the Profile.

Question: How can we be sure that all the operations of a Web Service have been covered in the testing?

Answer: Because the test framework – in the current version – does not include a Test Driver, a complete coverage of all the operations will rely on the client program involved in the testing of the Web Service, which is either ad-hoc, or is a real application in deployment over which the test operator does not have much control. A Test Driver will require advanced parameterization so that it can exercise the testing of all the (request-response and one-way) operations of a Web

Service, for any Web Service. Even so, such a driver may not be able to trigger an exhaustive set of behaviors. Finally, not all ports in a Web Service may be required to be conforming.

Question: What are some practical situations where the testing tools show value to Web Services users or vendors?
Answer: An industry may define industry-specific Web Services – e.g. purchase order submission, request for product information - and specific usage scenarios. This industry may require that the Web Service, when used according to these expected interaction scenarios, exhibits a profile-conforming behavior, as verified by WS-I testing tools. In order to achieve this, this industry will likely define a specific test driver for its Web Services. By doing so, this industry has effectively defined an industry-specific test harness and certification criterion for interoperability, based on the WS-I Profile. If such a Web Service passes the tests, a vendor in this industry can claim that it is interoperable with any user application, provided that the user also complies with the WS-I Profile, and exercises the expected usage scenarios.
Another scenario shows value for interoperability trouble shooting: a client application may fail to interoperate with a Web Service, although both claim to be conforming to the WS-I Profile. Because the testing tools can monitor messages from both interacting parties, the tools can be used to diagnose a failure to interoperate, and to identify the cause: either the client application or the Web Service may exhibit non-conforming behavior during this particular interaction. This will help determine responsibilities.

Question: Is it acceptable for a Web Service to have some operations conforming to the Profile, and some other not conforming?
Answer: Yes. The Profile requirements are not defined at Service level, but at a lower level, typically at WSDL port level. Some requirements are about operations, or bindings. A Web Service may have some of its ports using the profile-conforming SOAP binding, some other port using a non-conforming SOAP binding, and some ports using a non-SOAP binding. Yet, some business users may only be interested in interoperating with these ports that are Profile-conforming. Therefore the testing tools will be able to assess the conformance of a Web Service at port level. This will simply require exercising this port only, during a monitoring session. (As well as targeting this port only when testing the WSDL file.)

Question: Will the WS-I Test Framework also support functional testing of the Web Service?
Answer: This is outside of the scope of conformance testing to the WS-I Profile. Such testing would involve knowledge of the application semantics that is specific to each Web Service. The Monitor developed by the Test working group could however be reused – for example by the Sample Application working group – to provide the message capture necessary to such testing.

Question: Are there some restrictions in using the testing tools?
Answer: This version of the Monitor will not handle secure connections. In particular, the current version of the Monitor will not handle SSL. This does not preclude one from using SSL with the WS-I Profile, but SSL traffic cannot be captured in the current version of Monitor. To add SSL capability to the Monitor, the tool would have to be coded to handle SSL handshakes as well as to hold its own server certificate. SSL is specifically designed to thwart man-in-the-middle attacks, which the current design of the monitor requires.

Acknowledgements

For the Java version of the tools: