Testing Tools V0.91 User Guide

Document Type:
Technical User Guide

Editor:
Brian Macker, Computer Associates

Last Edit Date:
4/1/2003 11:56 AM

Document Status:
Version 0.91

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.

Notice

The material contained herein is not a license, either expressly or impliedly, to any intellectual property owned or controlled by any of the authors or developers of this material or WS-I. The material contained herein is provided on an "AS IS" basis and to the maximum extent permitted by applicable law, this material is provided AS IS AND WITH ALL FAULTS, and the authors and developers of this material and WS-I hereby disclaim all other warranties and conditions, either express, implied or statutory, including, but not limited to, any (if any) implied warranties, duties or conditions of merchantability, of fitness for a particular purpose, of accuracy or completeness of responses, of results, of workmanlike effort, of lack of viruses, and of lack of negligence. ALSO, THERE IS NO WARRANTY OR CONDITION OF TITLE, QUIET ENJOYMENT, QUIET POSSESSION, CORRESPONDENCE TO DESCRIPTION OR NON-INFRINGEMENT WITH REGARD TO THIS MATERIAL.

IN NO EVENT WILL ANY AUTHOR OR DEVELOPER OF THIS MATERIAL OR WS-I BE LIABLE TO ANY OTHER PARTY FOR THE COST OF PROCURING SUBSTITUTE GOODS OR SERVICES, LOST PROFITS, LOSS OF USE, LOSS OF DATA, OR ANY INCIDENTAL, CONSEQUENTIAL, DIRECT, INDIRECT, OR SPECIAL DAMAGES WHETHER UNDER CONTRACT, TORT, WARRANTY, OR OTHERWISE, ARISING IN ANY WAY OUT OF THIS OR ANY OTHER AGREEMENT RELATING TO THIS MATERIAL, WHETHER OR NOT SUCH PARTY HAD ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

License Information

Use of this WS-I Material is governed by the WS-I Test License at http://www.ws-i.org/docs/license/test_license.htm. By downloading these files, you agree to the terms of this license.

Feedback

The Web Services-Interoperability Organization (WS-I) would like to receive input, suggestions and other feedback ("Feedback") on this work from a wide variety of industry participants to improve its quality over time.

By sending email, or otherwise communicating with WS-I, you (on behalf of yourself if you are an individual, and your company if you are providing Feedback on behalf of the company) will be deemed to have granted to WS-I, the members of WS-I, and other parties that have access to your Feedback, a non-exclusive, non-transferable, worldwide, perpetual, irrevocable, royalty-free license to use, disclose, copy, license, modify, sublicense or otherwise distribute and exploit in any manner whatsoever the Feedback you provide regarding the work. You acknowledge that you have no expectation of confidentiality with respect to any Feedback you provide. You represent and warrant that you have rights to provide this Feedback, and if you are providing Feedback on behalf of a company, you represent and warrant that you have the rights to provide Feedback on behalf of your company. You also acknowledge that WS-I is not required to review, discuss, use, consider or in any way incorporate your Feedback into future versions of its work. If WS-I does incorporate some or all of your Feedback in a future version of the work, it may, but is not obligated to include your name (or, if you are identified as acting on behalf of your company, the name of your company) on a list of contributors to the work. If the foregoing is not

acceptable to you and any company on whose behalf you are acting, please do not provide any Feedback.

Feedback on this document should be directed to wsi-test-comments@ws-i.org.

Acknowledgement from the Editor: My 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

1.1 General Testing Process and Architecture

The Web Services Interoperability Organization (WS-I) has developed testing tools that evaluate Web services conformance to the Basic Profile. 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.

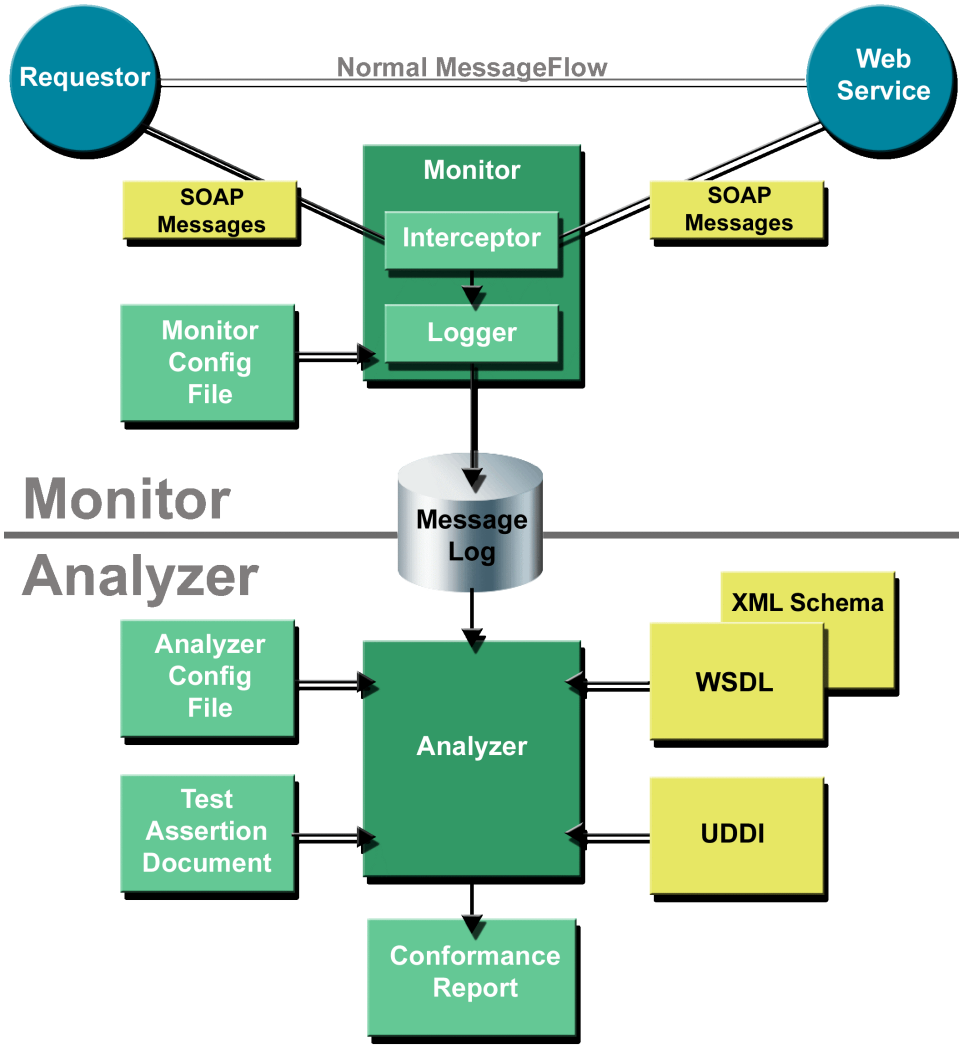

Figure 1 - Testing Tools Architecture.

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 the Basic Profile. 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.

1.2 Q & A on What to Expect from the Testing Tools

Question: Can the 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 Basic Profile, 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 Basic Profile.

Question: Can the testing tools verify all the requirements of the Basic Profile?
Answer: No. Some requirements of the WS-I basic profile cannot be easily tested, at least not by using a non-intrusive testing framework like the one described here. 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 Basic 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 Basic 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 Basic 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 Basic 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 Basic 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 Basic 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.

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:

- **C#** - The Microsoft .NET Framework release 1.1 "redistributable final beta" 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, the root directory of the directory structure is on the C drive in a directory called wsi-test-tools.

3 Using the Monitor Tool

The Monitor is implemented with a “man in the middle” approach so that it can intercept all the SOAP messages between the requestor and the service. The monitor configuration file controls the operation of the monitor and defines the parameters to ensure the SOAP messages are properly routed.

3.1 Configuration

The configuration file is an XML document that provides the key parameters for the monitoring tool.

Suppose that an attempt is being made to monitor two web service providers being accessed by some requestor application. In order to insert the monitor between the requestor and web service some method of redirection is required. In this example the choice is made to modify the requestor to access the monitor instead. Further suppose that the original web services were located at http://www.coldrooster.com port 80 and http://www.tempuri.com port 80 and that these will be mapped to Monitor ports 9090 and 8080 respectively. Then the requestor application will need to be modified to redirect its requests to the proper Monitor ports. Once this is done the following configuration file (Figure 2) will accomplish the redirection:

```
<?xml version="1.0" encoding="utf-8" ?>
<wsi-monConfig:configuration
   xmlns:wsi-common="http://www.ws-i.org/testing/2003/03/common/"
   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-common:addStyleSheet href="../common/xsl/messageLog.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 9090</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 Monitor Config File.

The schema for the configuration file can be found in the specification or relative to your working directory at wsi-test-tools\common\schemas.
The sample configuration file above instructs the monitor to do the following things:

1. Log all messages to a file named **c:\traceLog.xml**. Replacing any existing file.
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 **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.
9. Set the duration of the testing session to **900** seconds.

The definitions of the elements in the configuration file are as follows:

Element	Description	Attributes
Configuration	Root element for the configuration file.	[None]
Comment	Provides descriptive information about the monitor configuration document and does not affect execution. May be used at the configuration and redirect levels.	[None]
logFile	Identifies the file that will recieve the serialized XML output and optionally an XML style sheet. **Note:** The valid values for the replace attribute are: • true – the file is deleted at startup and a new file is created • false – (default) the monitor will fail to start and issue an error message	• **replace** Controls behavior if the logFile already exists. • **location** Identifies the name of the file that will contain the serialized XML.

Element	Description	Attributes
wsi-common: addStyleSheet	Indicates if a XML [3] style sheet reference should be added to the output trace log. **Note:** If this element is not specified, 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\traceLog.xsl"? -->	• **href** The location of the style sheet.. • **type** The content type for the style sheet. The default for this attribute is "text/xsl". • **title** Advisory information about the style sheet. • **media** Intended destination medium. • **charset** Character encoding for the style sheet. • **alternate** Indicates use of alternate style sheet.
logDuration	Identifies the number of seconds that the monitor will accept new client connections. After this many seconds, the monitor stops accepting any new client connections, and will write the log file..	[None]
cleanupTimeoutSeconds	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.	[None]
manInTheMiddle	Containing element for any manInTheMiddle port monitors.	[None]
redirect	The elements that define where the monitor will listen for traffic and how it will forward that traffic. Specify one or more as required.	[None]
listenPort	Tells the manInTheMiddle which port to listen on.	[None]
schemeAndHostPort	When traffic is received on listenPort, it is sent to the URL specified here. The URL can be specified as either: • An HTTP URL as specified in RFC1738 [2], section 3.3: *http://<host>:<port>/<path>?<searchpart>* • A subset form: *http://<host>:<port>* In both cases the port is optional and defaults to 80.	[None]
maxConnections	Specifies the maximum number of connections that the port will queue up before it begins refusing connections.	[None]
readTimeoutSeconds	Specifies 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	[None]

Element	Description	Attributes
	data, then neither connection is assumed to have timed out.	

In this release, the monitor and analyzer are only designed to handle HTTP traffic. The full monitor specification is available from www.ws-i.org.

3.2 Running the Monitor

3.2.1 Executing the C# Version of the Monitor

To run the message monitor (from the bin directory):

monitor [-config <configuration_file>]

Example:
cd <working-directory>\wsi-test-tools\cs\bin
monitor -config ..\samples\monitorConfig.xml

Note: If no configuration file is defined, the monitor will default to monitorConfig.xml file in the current directory. You must either wait for the logDuration time to expire or manually exit the monitor to properly close and complete the monitor log file. To manually exit, press <Ctrl-C>.

3.2.2 Executing the Java Version of the Monitor

To run the message monitor:

bin\Monitor -config <configFilename>

Example:
cd <working-directory>\wsi-test-tools\java
bin\Monitor -config samples\monitorConfig.xml

Note: You must either wait for the logDuration time to expire or manually exit the monitor session to close and complete the monitor log file. To manually exit, type “exit” at the command prompt and press Enter, or optionally press <Ctrl-C>

3.2.3 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, simply modify this accessPoint.

3.3 The Monitor Output

3.3.1 The Message Log File

The monitor logs all information to an XML file. The file starts with some header information explaining what monitor produced the log, and what the configuration options were. An example is given in Figure 3 below:

```
<?xml version="1.0" encoding="utf-8" ?>
<?xml-stylesheet type="text/xsl" href="..\lib\messageLog.xsl"?>
<wsi-log:log xmlns:wsi-log="http://www.ws-i.org/testing/2003/03/log/"
      xmlns:wsi-common="http://www.ws-i.org/testing/2003/03/common/"
   xmlns:wsi-monConfig=
      "http://www.ws-i.org/testing/2003/03/monitorConfig/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   timestamp="2002-10-01T13:07:30.9200652-07:00">
     <wsi-log:monitor version="1.0" releaseDate="2002-12-20">
          <wsi-log:implementer name="WS-I Organization" location="http://www.ws-
i.org/CSharp/2003/03/Monitor.zip" />
          <wsi-log:environment>
               <wsi-log:runtime name=".NET" version="1.0.3705.288" />
               <wsi-log:operatingSystem name="Win32NT"
          version="5.1.2600.0" />
               <wsi-log:xmlParser name=".NET" version="1.0.3705.288" />
          </wsi-log:environment>
     <wsi-monConfig:configuration>
       <wsi-monConfig:logFile replace="true"
          location="c:\traceLog.xml">
          <wsi-common:addStyleSheet
              href="../common/xsl/messageLog.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:listenPort
              >9090</wsi-monConfig:listenPort>
            <wsi-monConfig:schemeAndHostPort
              >http://localhost</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:listenPort
              >8080</wsi-monConfig:listenPort>
            <wsi-monConfig:schemeAndHostPort
              >http://localhost</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="2002-10-01T13:07:31.9200652-07:00"
        conversationID="e1432705-4984-4d9f-a00f-54abd8ec72e9"
    ID="1"
    type="request" >
        <wsi-log:messageContent>&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;FlightSearch
xmlns="http://tempuri.org/"&gt;&lt;Departure&gt;Point A&lt;/Departure&gt;&lt;
Destination&gt;Point B&lt;/Destination&gt;&lt;DepartureDate&gt;
2001-10-12T00:00:00.0000000-07:00&lt;/DepartureDate&gt;&lt;ReturnDate&gt;
2001-12-12T00:00:00.0000000-08:00&lt;/ReturnDate&gt;&lt;/FlightSearch&gt;&lt;
/soap:Body&gt;&lt;/soap:Envelope&gt;</wsi-log:messageContent>
        <wsi-log:senderHostAndPort
      >127.0.0.1:4870</wsi-log:senderHostAndPort>
        <wsi-log:receiverHostAndPort
      >localhost:80</wsi-log:receiverHostAndPort>
        <wsi-log:httpHeaders>POST /AsyncWebService/Service1.asmx HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client
Protocol 1.0.3705.288)
Content-Type: text/xml; charset=utf-8
SOAPAction: "http://tempuri.org/FlightSearch"
Content-Length: 489
Expect: 100-continue
Connection: Keep-Alive
Host: localhost
```

```
</wsi-log:httpHeaders>
    </wsi-log:messageEntry>
    <wsi-log:messageEntry xsi:type="wsi-log:httpMessageEntry"
    timestamp="2002-10-01T13:07:37.4269711-07:00"
        conversationID="e1432705-4984-4d9f-a00f-54abd8ec72e9"
    ID="2"
    type="response" >
        <wsi-log:messageContent>&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;
FlightSearchResponse
xmlns="http://tempuri.org/"&gt;&lt;FlightSearchResult&gt;VF77&lt;/
FlightSearchResult&gt;
&lt;Flight&gt;VF30&lt;/Flight&gt;&lt;Time&gt;9:00
AM&lt;/Time&gt;&lt;/FlightSearchResponse&gt;&lt;/soap:Body&gt;&lt;
/soap:Envelope&gt;</wsi-log:messageContent>
        <wsi-log:senderHostAndPort
      >localhost:80</wsi-log:senderHostAndPort>
        <wsi-log:receiverHostAndPort
      >127.0.0.1:4870</wsi-log:receiverHostAndPort>
        <wsi-log:httpHeaders>HTTP/1.1 200 OK
Server: Microsoft-IIS/5.1
Date: Tue, 01 Oct 2002 20:07:37 GMT
Cache-Control: private, max-age=0
Content-Type: text/xml; charset=utf-8
Content-Length: 405

</wsi-log:httpHeaders>
    </wsi-log:messageEntry>
</wsi-log:log>
```

Figure 3 – Example of Message Log file.

The schema for the log file can be found in the specification or relative to your working directory at wsi-test-tools\common\schemas.

What follows here is a description of the elements:

Element	Description	Attributes
log	Root element for the message log.	• **timestamp** Identifies the time that the log was created. This value is less than or equal to the timestamp of the first messageEntry in the log.

Element	Description	Attributes
monitor	Contains information that should allow someone reading a log to identify exactly which tool produced the log. This element also contains the XML in the configuration file used by the monitor.	• **version** The version number for the implementation of the tool. • **releaseDate** The date the tool was released.
implementer	Identifies the organization that created the monitor tool.	• **name** Name of the organization that created the monitor tool. • **location** URL pointing to the Web site where the monitor tool can be obtained.
environment	Describes the environment used to run the application and the language used to construct the application.	[None]
runtime	Identifies the primary programming language used to create the monitor.	• **name** Name of the runtime environment. Ex: Java, .NET, etc. • **version** Identifies the version of the runtime in use. This could be a dotted version string (1.0.3705.288.
operatingSystem	Identifies the name and version of the operating system that the monitor is running on.	• **name** Name of the operating system. • **version** Identifies which version of the operating system is in use.
xmlParser	For some monitors, it is known that the XML Parser version will be important. This element is optional and will only be included if the XML Parser makes a difference.	• **name** Recognized name of the XML parser. • **version** Identifies which version of the XML Parser was used.
configuration	This is a straight import of the configuration file used when the logFile was created. This is included to help understand how the log itself produced its results.	[None]
messageEntry	Used to identify a message. This represents a message moving in one direction.	• **timestamp** Time that the message was received by the monitor tool. • **conversationID** 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. • **ID** This attribute is used to uniquely

Element	Description	Attributes
		identify the message within the log. • **type** Identifies the entry as an HTTP request or an HTTP response. Valid values are request and response.
messageContent	Contains the HTTP Body message in the HTTP POST or HTTP response. Special characters like '<' and '>' will be converted to their entity reference equivalents. In this case &lt and &gt respectively.	[None]
senderHostAndPort	This identifies the host and TCP port that is sending data. In a Request message, this will match the value of the host element in the monitor configuration file. Instead of including the scheme identifier, the value will simply be <host>:<port>. The port is listed in this element. When a port is not specified in an HTTP-based URL, the port value defaults to 80.	[None]
receiverHostAndPort	This identifies the host and TCP port that is receiving data. In a Response message, this will match the value of the host element in the monitor configuration file. Instead of including the scheme identifier, the value will simply be <host>:<port>. The port is listed in this element.. When a port is not specified in an HTTP-based URL, the port value defaults to 80.	[None]
httpHeaders	The raw text of any HTTP headers sent by one direction of an HTTP request or response.	[None]

It is important to note that the monitor only logs the HTTP conversation, and makes no assumptions about the content of an HTTP body. The 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.)

4 Using the Analyzer Tool

4.1 Configuration

The analyzer configuration contains the list of options for this tool. This file may also contain implementation-specific configuration parameters. Three examples of configuration XML files are shown below:

The first example in Figure 4 directly references the WSDL document, e.g. a local file:

```
<?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/2003/03/analyzerConfig/"
   xmlns:wsi-common="http://www.ws-i.org/testing//2003/03/common/">
  <wsi-common: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-common: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-common: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 schema for the configuration file can be found in the specification or relative to your working directory at wsi-test-tools\common\schemas. The sample configuration file in the previous example instructs the analyzer to do the following things:

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 these, like the ones which failed (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, as it may be a large document.
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 there was already a report with same name, will replace it (replace="**true**").Write the log file to "tracelog.xml" (wsi-analyzerConfig:logFile value is **traceLog.xml**).
6. Correlate messages to Web services based on the endpoint (correlationType="**endpoint**").
7. The conformance target is a single port: (wsdlElement type="**port**" , parentElementName="**RetailerService**")
8. Take the WSDL definitions from "samples/retailer.wsdl" (wsdlElement namespace="http://.../Retailer.wsdl").

The second example in Figure 5 refers to the WSDL document, via the Service location:

```
<?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/2003/03/analyzerConfig/"
   xmlns:wsi-common="http://www.ws-i.org/testing//2003/03/common/">
  <wsi-common: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-common: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-common: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

An example of configuration file that points to a UDDI entry as the source of description material is shown in Figure 6 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/2003/03/analyzerConfig/"
    xmlns:wsi-common="http://www.ws-i.org/testing//2003/03/common/">
  <wsi-common: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-common: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-common: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:wsdlElement type="binding"
        namespace="http://.../Retailer.wsdl">
      RetailerSoapBinding
    </wsi-analyzerConfig:wsdlElement>
    <wsi-analyzerConfig:uddiKey type="tModelKey">…</wsi-analyzerConfig:uddiKey>
    <wsi-analyzerConfig:inquiryURL>
      http://tempuri.org/uddi/inquiryapi
    </wsi-analyzerConfig:inquiryURL>
  </wsi-analyzerConfig:uddiReference>
</wsi-analyzerConfig:configuration>
```

Figure 5 – Example of Analyzer Configuration File Using UDDI Reference.

The definitions of the elements in the analyzer configuration file are:

Element	Description	Attributes
configuration	The root element for the configuration file.	• **name** The name associated with the option settings in the configuration file.

wsi-common: description	A text description of the parent element.	[None]
verbose	Used to indicate whether diagnostic information should be displayed while the analyzer is running. The valid values for this element are "true" or "false". If this element is not specified, then the value is "false".	[None]
assertionResults	This element is used to indicate the type of assertion results that should be listed in the conformance report. **Note:** The values for the type attribute have the following meaning: • all List the results from all test assertions. • notPassed List all of the assertion test results except the ones that have a result of passed. • onlyFailed List only the test assertion results which have a result of failed.	• **type** The type of assertion results to include in the conformance report. The default value is "all". • **messageEntry** If "true", then include message entries in the report file. If "false", the log entries are not included in the report file. This attribute is "true" by default. • **assertionDescription** If "true", then include the assertion description for each test assertion in the report. If "false", then the assertion descriptions are not included the report. This attribute is "false" by default. • **failureMessage** If "true", then include the failure messages that are pre-defined for each test assertion in the report. This attribute is "false" by default. • **failureDetail** If "true", then include the failure detail messages in the report. This attribute is "true" by default.
reportFile	The name of the output conformance report file.	• **replace** This attribute indicates whether the report file should be replaced if it already exists. The valid values for this attribute are "true" or "false". If the report file already exists and this attribute is set to "false", then the analyzer will terminate with an error message. The default value for this attribute is "false". • **location** The location where the report file should be

		created.
wsi-common: addStyleSheet	Indicates if a style sheet reference should be added to the output conformance report. **Note:** If this element is not specified, 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"? -->	• **href** The location of the style sheet. • **type** The content type for the style sheet. The default for this attribute is "text/xsl". • **title** Advisory information about the style sheet. • **media** Intended destination medium. • **charset** Character encoding for the style sheet. • **alternate** Indicates use of alternate style sheet.
testAssertionFile	This element contains the location of the WS-I test assertion document, which is based on a profile definition.	[None]
logFile	The location of the messages that will be processed by the analyzer tool. **Note:** The valid values for the correlationType attribute are: • endpoint 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 The correlation process will use both the endpoint and 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.) • operation Correlation requires a match on the endpoint, namespace, 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.) **Note:** If this element does not appear in the	• **correlationType** Defines the kind of information used to match a message from the log file with the Web service that is being tested. The default value is "operation".

	configuration file, then all of the test assertions that operate on the log entries will not be processed.	
wsdlReference	This element contains a reference to the WSDL element and description document which should be analyzed. **Note:** If this element does not appear in the configuration file, then the WSDL related test assertions will not be processed.	[None]
wsdlElement	This element contains the reference to the WSDL element that should be analyzed. This element contains a reference to the type of element referenced by the type attribute **Note:** The following values can be specified on the type attribute. Each value corresponds to a WSDL element. • port • binding • portType • operation • message	• **type** The type of WSDL element that is referenced by the name attribute. • **parentElementName** The attribute is only required when the type attribute has a value of "port" or "operation". The parent element name is used to qualify the reference to a WSDL port definition within a service element, and the reference to a operation definition within a portType.
wsdlURI	This element contains the location of the WSDL document for the Web service.	[None]
serviceLocation	There are times when the service location is not defined in a WSDL document, but this information is required by the analyzer. When this situation occurs, the <wsdlElement> element should reference a WSDL binding and this element should contain the service endpoint. **Note:** 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 in the location attribute on the <soapbind:address> element.	[None]

uddiReference	This element can be used to reference a single UDDI bindingTemplate or tModel. This element is never specified with the <wsdlReference> element. **Note:** If this element does not appear in the configuration file, then both the UDDI and WSDL related test assertions will not be processed.	[None]
uddiKey	Contains either a bindingKey or tModelKey.	• **type** The type of UDDI key. The valid values are uddi:bindingKey or uddi:tModelKey.
inquiryURL	This element contains the inquiry URL that can be used to retrieve the UDDI bindingTemplate or tModel associated with the uddiKey element.	[None]

4.2 Running the Analyzer

4.2.1 Executing the C# Version of the Analyzer

To run the analyzer tool (from the bin directory):

analyzer [-config < configFilename >]

Example:
cd <working-directory>\wsi-test-tools\cs\bin
analyzer -config ..\samples\analyzerConfig.xml

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

4.2.2 Executing the Java Version of the Analyzer

To run the analyzer tool:

bin\Analyzer -config <configFilename>

Example:
cd <working-directory>\wsi-test-tools\java
bin\Analyzer -config samples\analyzerConfig.xml

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

4.2.3 Analyzer Tool Command Line Syntax

The command line options have the same meaning as the options defined in the configuration file. All command line options override the options that are specified in the configuration file. Note: the –config and –verbose options are supported by both tool packages (C# and Java). The Java analyzer supports additional options.

```
analyzer –config <file-location>
  -verbose <verbose-values>
```

The following table contains a definition of the command line options for the analyzer tool.

	Option	Definition	Required
1	-config	This option contains a reference to the analyzer configuration file.	Yes (by Java tool)
2	-verbose	Display diagnostic messages on the console.	No

4.3 Analyzer Input Material

4.3.1 Types of Input

The test target material that will be analyzed is of three different types:

- Description (WSDL data)
- Messages (HTTP message items from the XML message log file)
- Discovery (UDDI entries)

4.3.2 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 three types of artifacts, which correspond to the different inputs provided to the Analyzer:
    - messages: designate the entries in the *message log file*.
    - description: designate WSDL files or parts of these.
    - discovery: 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 two entry types are associated with the "messages" artifact
    - "port", "binding", "portType" are entry types that are associated with the "description" artifact
    - "bindingTemplate", "tModel" are entry types for the "discovery" 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).

4.3.3 Cases where incomplete input is provided

The test assertion document for the Basic Profile defines three primary artifacts: messages, description, and discovery. These three artifacts correlate to the <logFile>, <wsdlReference> and <uddiReference> elements, respectively. The following rules describe the expected combinations, and the behavior of the analyzer for these combinations:

- If only the <logFile> element is specified, then all of the 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 the description and discovery artifacts are processed.
- If only the <wsdlReference> element is specified, then the test assertions for just the description artifact are processed.
- If the <logFile> and <wsdlReference> elements are specified, then the test assertions for both the messages and 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 what is the behavior of the analyzer in such cases.

	Input Artifact State	**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 "notTestable".
2	A WSDL document is provided as input but it does not contain the WSDL element which is specified on the <wsdlElement> element.	The analyzer will terminate with an error 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.

4.4 The Test Assertions Document

4.4.1 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), stored in the **BasicProfileTestAssertion.xml** file. This document will be used by the Analyzer as input, and will determine the set of test procedures that will be activated.

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

```
<testAssertion id="WSI3003" entryType="tModel" type="required" enabled="true">
    <context>For a candidate uddi:tModel</context>
    <assertionDescription>The uddi:tModel is categorized using the uddi:types taxonomy, as
"wsdlSpec": the uddi:keyedReference element has an attribute keyValue equal to "wsdlSpec", and
keyName equal to "uddi-org:types" or "types"</assertionDescription>
    <failureMessage>The uddi:tModel is not categorized using the uddi:types taxonomy with a
categorization of "wsdlSpec".</failureMessage>
    <failureDetailDescription>{tModel key}{categoryBag}</failureDetailDescription>
    <additionalEntryTypeList>
      <messageInput>none</messageInput>
      <wsdlInput>none</wsdlInput>
      <uddiInput>none</uddiInput>
    </additionalEntryTypeList>
    <prereqList/>
    <referenceList>
      <reference profileID="BP1">R3003</reference>
    </referenceList>
    <comments/>
  </testAssertion>
```

Figure 7 – Example of Test Assertion.

4.4.2 Term Definitions

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

- **Test Assertion:** A test assertion is a translation of a WS-I profile requirement into a statement verifiable by the 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.

- **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".
- **Context**: Intuitively, the context of a test assertion defines which test 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 the entry (or related secondary entries) the relevant pre-requisite test assertions didn't pass. Pre-requisite test assertion Ids are listed in the XML element: "prereqList" in the TAD.

4.4.3 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 passed 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.
- **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 qualification criteria 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 relatin to WSDL.)

- **notTestable**
  The test assertion was not processed due to a lack of entries of the expected type.

4.5 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 BasicProfileTestAssertion.xml document). 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 users may want to consult, as a more readable document for conformance assessment.

4.5.1 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.

```
<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="2003-03-25T16:56:56.905Z"
   xmlns="http://www.ws-i.org/testing/2003/03/report/"
   xmlns:wsi-report="http://www.ws-i.org/testing/2003/03/report/"
   xmlns:wsi-log="http://www.ws-i.org/testing/2003/03/log/"
   xmlns:wsi-common="http://www.ws-i.org/testing/2003/03/common/"
   xmlns:wsi-analyzerConfig="http://www.ws-i.org/testing/2003/03/analyzerConfig/"
   xmlns:wsi-monConfig="http://www.ws-i.org/testing/2003/03/monitorConfig/"
   xmlns:wsi-assertions="http://www.ws-i.org/testing/2003/03/assertions/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <analyzer version="0.91" releaseDate="2003-03-25">
   <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="1.4.0_03-
b04"/>
     <operatingSystem name="Windows 2000" version="5.0"/>
     <xmlParser name="Apache Xerces" version="Xerces-J 2.2.1"/>
   </environment>
   <wsi-analyzerConfig:configuration>
     <wsi-analyzerConfig:verbose>false</wsi-analyzerConfig:verbose>
```

```
      <wsi-analyzerConfig:assertionResults type="all" messageEntry="true" failureMessage="true"
failureDetail="true"/>
        <wsi-analyzerConfig:reportFile replace="true" location="report.xml">
      <wsi-common:addStyleSheet href="../common/xsl/report.xsl" type="text/xsl"/>
        </wsi-analyzerConfig:reportFile>
        <wsi-analyzerConfig:testAssertionsFile>
          ../common/profiles/BasicProfileTestAssertions.xml
        </wsi-analyzerConfig:testAssertionsFile>
        <wsi-analyzerConfig:logFile>log.xml</wsi-analyzerConfig:logFile>
      <wsi-analyzerConfig:wsdlReference>
        <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 >
      <assertionResult id="WSI3002" result="notTestable">
      </assertionResult>
      <assertionResult id="WSI3003" result="notTestable">
      </assertionResult>
    </entry>
  </artifact>
  <artifact type="description">
    <entry type="definitions" referenceID="file:samples/RetailerService.wsdl">
      <assertionResult id="WSI2702" result="passed">
      </assertionResult>
    </entry>
    <entry type="definitions" referenceID="http://www.ws-
i.org/SampleApplications/SupplyChainManagement/2002-08/Retailer.wsdl">
      <assertionResult id="WSI2702" result="passed">
      </assertionResult>
    </entry>
    <entry type="definitions" referenceID="http://www.ws-
i.org/SampleApplications/SupplyChainManagement/2002-08/Configuration.wsdl">
      <assertionResult id="WSI2702" result="passed">
      </assertionResult>
    </entry>
    <entry type="binding" referenceID="http://www.ws-
i.org/SampleApplications/SupplyChainManagement/2002-08/Retailer.wsdl:RetailerSoapBinding">
      <assertionResult id="WSI2019" result="notApplicable">
      </assertionResult>
      <assertionResult id="WSI2012" result="notApplicable">
      </assertionResult>
      <assertionResult id="WSI2020" result="passed">
      </assertionResult>
      <assertionResult id="WSI2021" result="passed">
      </assertionResult>
      <assertionResult id="WSI2022" result="passed">
      </assertionResult>
      <assertionResult id="WSI2023" result="passed">
      </assertionResult>
```

```
      <assertionResult id="WSI2404" result="passed">
      </assertionResult>
      <assertionResult id="WSI2406" result="passed">
      </assertionResult>
      <assertionResult id="WSI2013" result="passed">
      </assertionResult>
      <assertionResult id="WSI2017" result="passed">
      </assertionResult>
    </entry>
    <entry type="portType" referenceID="http://www.ws-
i.org/SampleApplications/SupplyChainManagement/2002-08/Retailer.wsdl:RetailerPortType">
      <assertionResult id="WSI2010" result="passed">
      </assertionResult>
    </entry>
    <entry type="operation" referenceID="getCatalog" parentElementName="RetailerService">
      <assertionResult id="WSI2208" result="passed">
      </assertionResult>
    </entry>
    <entry type="operation" referenceID="submitOrder" parentElementName="RetailerService">
      <assertionResult id="WSI2208" result="passed">
      </assertionResult>
    </entry>
  </artifact>
  <artifact type="message">
    <artifactReference timestamp="2003-03-25T16:06:03.605Z">
    <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="2003-03-25T14:20:51.234Z">
    <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="WSI1004" result="passed">
      </assertionResult>
      <assertionResult id="WSI1601" result="passed">
      </assertionResult>
      <assertionResult id="WSI1201" result="passed">
      </assertionResult>
      <assertionResult id="WSI1701" result="passed">
      </assertionResult>
      <assertionResult id="WSI1202" result="passed">
```

```
      </assertionResult>
      <assertionResult id="WSI1306" result="notApplicable">
      </assertionResult>
      <assertionResult id="WSI1316" result="notApplicable">
      </assertionResult>
      <assertionResult id="WSI1307" result="passed">
      </assertionResult>
      <assertionResult id="WSI1308" result="passed">
      </assertionResult>
      <assertionResult id="WSI1318" result="passed">
      </assertionResult>
      <assertionResult id="WSI1309" result="passed">
      </assertionResult>
      <assertionResult id="WSI1002" result="passed">
      </assertionResult>
      <assertionResult id="WSI1001" 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="WSI1203" result="notApplicable">
      </assertionResult>
    </entry>

    <!-- Other message entry results go here. -->

  </artifact>
  <summary result="passed">
  </summary>
</report>
```

Figure 8. Example of Profile Conformance Report.

The following table defines each of the elements that can be used in the conformance report file.

Element	Description	Attributes
report	The root element for the profile conformance report file.	• **name** The name of the conformance report. • **timestamp**

		The date and time that the report was generated.
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.	• **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. • **releaseDate** The date the tool was built.
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	• **name** The name of the organization that implemented the tool. • **location** Web site where you can get more information on the implementation of the tool.
environment	The environment that was used to run the analyzer tool.	[None]
runtime	The runtime that was used by the analyzer.	• **name** Runtime name. • **version** Runtime version.
operatingSystem	The operating system where the analyzer tools was run.	• **name** Operating system name. • **version** Operating system version.
xmlParser	The XML parser that was used when running the analyzer.	• **name** XML parser name. • **version** XML parser version.
wsi-analzyerConfig: 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.	[None]
artifact	This element contains a reference to one of the artifacts that is listed in the test assertion document.	• **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.
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	• **timestamp** The timestamp from the message log file or the date and time for the WSDL file.

	in the monitor configuration section of the log file if it is present.	
wsi-monConfig:comment	The comment element that is the first child of the configuration element in the message log file.	[None]
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: requestMessage responseMessage anyMessage definitions import types message portType binding port bindingTemplate tModel	• **type** The type of entry for the test assertion. • **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).
wsi-log:messageEntry	This element contains a reference to the log entry which was the target of a test assertion.	[None]
assertionResult	This element contains the result for a single execution of a test assertion for an entry. **Note:** 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 qualification criteria or a prerequisite test assertion failed. notTestable The test assertion was not processed because missing parameters did not allow the testing of the assertion.	• **id** Test assertion identifier. This value matches the value that is listed in the profile definition document. • **result** This attribute contains the result from the execution of the test assertion.
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. **Note:** The values for the type attribute have the same meaning as those for the <entry> element.	• **type** The type of entry for the test assertion. • **referenceID** This attribute is optional. When it is specified, it contains an unique identifier for an instance of the type of entry (an example would

		be an identifier for an entry in a log file).
assertionDescription	The assertion description for the test assertion.	• **xml:lang** The language associated with the description.
failureMessage	The failure message that is defined for the test assertion.	• **xml:lang** The language associated with the message.
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.	• **xml:lang** The language associated with the message. • **referenceType** The type of entity that caused all or part of the test assertion failure. This attribute is optional. • **referenceID** The identifier for the entity that caused all or part of the failure. This attribute is optional.
summary	This element is the container for the conformance report summary. **Note:** 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".	• **result** The value for this attribute is either "passed" or "failed".
analyzerFailure	When an 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 contain instructions on how to correct the error.	[None]

4.5.2 Conformance Report In HTML Format

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

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

Summary

Result	failed

Figure 9 – Summary line of of a conformance report.

The above example illustrate 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

Figure 10 – General Artifact Index.

The above is an index on each of the three 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
2003-03-06T22:24:51.687Z

Assertion Result Summary:

Assertion ID	Passed	Failed	Warning	Not Applicable	Not Testable
WSI1001	7	1	8	2	
WSI1002	16	2	0	0	
WSI1004	8	1	0	0	
WSI1201	11	1	0	6	
WSI1202	8	1	0	9	
WSI1203	1	1	0	16	
WSI1302	1	0	1	7	
WSI1305	1	1	0	7	
WSI1306	2	0	0	16	
WSI1307	7	1	0	10	
WSI1308	7	1	0	10	
WSI1309	8	0	0	10	
WSI1316	2	0	0	16	
WSI1318	7	1	0	10	
WSI1601	12	1	0	5	
WSI1701	8	4	0	6	

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 was 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: WSI1004

Result	**passed**

Assertion: WSI1601

Result	**failed**
Failure Message	The soap:Envelope or soap:Body does not conform to XML 1.0.
Failure Detail Message	The element type "ServiceID" must be terminated by the matching end-tag "</ServiceID>".

Assertion: WSI1201

Result	**notApplicable**
Failure Detail Message	The following prerequisite test assertions failed: WSI1601.

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. WSI1601), 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).

Acknowledgements

For the Java version of the tools:

This product includes software developed by the Apache Software Foundation (http://www.apache.org/); (c) 1999 The Apache Software Foundation. All rights reserved. THE APACHE SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR

PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THE APACHE SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

