Testing Work Group

Project:

WS-I Analyzer Tool Functional Specification
[Analyzer Tool Funtional Specification.doc]

Document Type:

Technical Design Specification

Editors:

Peter Brittenham, IBM
David Lauzon, IBM

Contributors:

Jim Clune, Parasoft
Jacques Durand, Fujitsu
Lucien Kleijkers, Microsoft
Krishna Sankar, Cisco Systems Inc.
Scott Seely, Microsoft
Keith Stobie, Microsoft
Graham Turrell, IBM

Last Edit Date:

3/24/2005 4:39 PM

Document Status:

Version 1.1

This is the Board Approval Draft release of this specification. The contents of this document may change before the final draft.

Table of Contents:

1 Introduction

The WS-I Test Tools Working Group is responsible for developing the supporting documentation and processes for WS-I Test Tool development, and the Test Materials used to test Web service implementations for conformance with a WS-I profile [4][5][6][7]. Since the profiles will vary in content, the testing tools and supporting materials should be designed so that they can be easily enhanced to support new profiles while still supporting the existing profiles.

The Test Tools Architecture consists of a message monitor and an analyzer. The monitor is used to log the messages that were sent to and from a Web service. The analyzer is used to validate that the Web service interactions contained in the message log conform to a WS-I profile. The analyzer is responsible for validating all of its input artifacts. This includes the message log file from the monitor, the WSDL-based Web service description, and UDDI entries.

This document contains the design for Version 1.1 of the analyzer tool, which will be used for conformance testing of WS-I profiles. This specification is normative. Any Analyzer tool claiming to be WS-I conforming, should implement the design features of this specification according to the level of requirement (mandatory, recommended, optional), or should implement the design features of subsequent specifications for Analyzer versions beyond 1.0, when applicable.

1.1 Notational Conventions

The following notational conventions are used in this document:

1. The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119.

2. The following namespace prefixes are used throughout this document:

Prefix	Namespace URI	Definition
wsi-log	http://www.ws-i.org/ testing/2003/03/log/	Message log
wsi-assertions	http://www.ws-i.org/ testing/2004/07/assertions/	Profile test assertions
wsi-report	http://www.ws-i.org/ testing/2004/07/report/	Profile conformance report
wsi-analyzerCconfig	http://www.ws-i.org/ testing/2004/07/analyzerConfig/	Analyzer configuration file
wsi-monConfig	http://www.ws-i.org/ testing/2003/03/monitorConfig/	Monitor configuration file
wsi-common	http://www.ws-i.org/ testing/2003/03/common/	Common element definitions

2 Analyzer Tool Design

The purpose of the Analyzer tool is to validate the messages that were sent to and from a Web service. The analyzer is also responsible for verifying the description of the Web service. This includes the WSDL document that describes the Web service, and the XML schema files that describe the data types used in the WSDL service definition.

The analyzer tool has a defined set of input files, all of which are used to verify conformance to a WS-I profile definition.

- Analyzer configuration file
- Test assertion definition file
- Message log file
- WSDL for the Web service

The analyzer configuration file and test assertion definition file are described in greater detail in the subsequent sections of this document (sections 2.4 and 2.5).

The message log file contains the list of messages that were captured by the monitor tool. The content and format of this file is defined in [1].

The WSDL document that describes a Web service could be specified as a URL, or as an entry in a UDDI registry. If a UDDI entry is specified, then it MUST conform to the format defined in the *Using WSDL in a UDDI Registry* best practices document [3].

The output from the analyzer will be a report, which will indicate whether the input to the analyzer conforms to a specific WS-I profile. All deviations from the profile specification MUST be listed in the conformance report.

2.1 General Term Definitions

The following terms are used when describing the materials used as input to the analyzer:

Artifact: General term used to designate the material used as input to the analyzer. For the basic profiles, there are four types of artifacts:

- message: the transport and message protocol[1] (e.g. SOAP/HTTP message)
- envelope: the serialization of the soap:Envelope element and its content
- description: any material within WSDL files
- discovery: any material represented in UDDI, not including WSDL items

Entry type: An entry type is a sub-type of an artifact type. It defines the type of artifact data that should be analyzed.

Artifact	Entry Types
message	requestMessage, responseMessage, anyMessage
envelope	requestEnvelope, responseEnvelope, anyEnvelope
description	port, binding, portType, operation, message, import, types, definitions
discovery	bindingTemplate, tModel

[1] Note in Basic Profile 1.0, the message artifact also included the message contents.

Entry: An entry is an instance of an entry type. The entry MUST be uniquely identified within the scope of the material used for a test analysis.

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

2.2 Analyzer Requirements

This section contains a summary of the requirements for the analyzer tool. All implementations of the analyzer tool MUST adhere to these requirements.

- Be able to activate or generate an analyzer instance based on a specified profile definition.

  This type of approach would delegate the responsibility for understanding each profile to the analyzer implementation. This isn't as good as having one versatile analyzer implementation that could configure itself based on a profile definition, but it is the most realistic approach.

  As an example, in Java this could be done by having an Analyzer interface or abstract class that is implemented to provide support for a profile. A factory mechanism could be used to return the analyzer implementation based on the profile definition that was specified.

  ```
  Analyzer analyzer = AnalyzerFactory.newAnalyzer(ProfileDefinition profile);
  ```

- The input parameters for the analyzer tool MAY vary based on the profile definition that it supports.

  Future profile definitions may require additional input options beyond those that were required by the basic profiles. An analyzer tool MUST be designed so that it is easy to add new parameters to an implementation.

- The analyzer MUST provide a set of artifact validators.

  A validator is a function that verifies conformance of an artifact entry to a particular profile. Entity validators focus on the artifacts that are independent of the information in the message log files. Examples of these validators are the UDDI and WSDL validators.

  The message validators are used to validate test assertions that are applicable to the messages that are sent to and from a Web service. Examples of these validators are the messageTransport validator which validates the transport data and message protocol, and the envelope validator that validates the soap:Envelope element and its content.

  The message validators need to correlate request and response messages. Some test assertions will require analysis of the request and response message together. This requirement will imply that the message validators must be able to process request/response message pairs, instead of just single messages.

The analyzer MUST be designed so that each of the entity validators and all of the message validators can be invoked separately. For example, the WSDL validator should be able to validate any WSDL document independent of running the UDDI validation and the message validation functions. This could be accomplished by specifying input options that limit the number of validation functions that are used.

Also, an implementation MAY choose to design these functions so that they can be used outside of the analyzer tool. For example, the message validators could be designed to process a single message that is outside the context of a log file. This would allow for interactive processing of messages, in addition to batch processing where a set of messages is read from a message log file.

- An analyzer implementation MUST support the entire set of test assertions defined in a test assertion document.

  This means that an analyzer implementation MUST have a set of validators that will validate the complete set of test assertions for a specified profile.

- An analyzer implementation MUST separate data processing functions from view functions.

  The primary functions of the analyzer are the validators which process the test assertions. These functions should not be tied directly to any GUI. This would allow the base implementation to use a command line interface, and it would also allow the analyzer functions to be enhanced to use different types of GUI interfaces.

- The analyzer tool MUST produce the conformance report as it is specified in the analyzer functional specification.

  For example, given a set of inputs (profile definition, WSDL, message log file, etc.), all analyzer implementations should produce the same conformance report as output. The only differences should be the content that may contain implementation specific information.

2.3 Analyzer Functional Description

The analyzer tool has two sets of functions. One is to validate the discovery and description artifacts for a Web service to ensure that they conform to a WS-I profile definition. The second function is to analyze artifacts related to the operation of a Web service to ensure that they conform to the profile definition. The second set of artifacts is the messages that are captured by the monitor in a message log file. The first function MUST be processed before the second function is started.

The following list is a functional overview of the analyzer tool for the basic profiles:

- Process the input parameters to locate configuration file.
- Read the configuration file and process the options contained in this file.
- Read the test assertion definition file, which contains the test assertions for the profile.
- Process the discovery and description validation functions.

  - o This includes reading the WSDL document and associated XML schema files to validate that the format and content of the document conforms to the profile definition.

- Run the message validation functions. These test assertions are processed for each message in a message log file.
  - o For the basic profiles, this will include validating the transport and message protocol, as well as, the message content.

- The analyzer must generate a conformance report. The contents of this report can be produced as each test assertion is processed or it can be generated when all of the validation functions have been processed.

The analyzer tool SHOULD be designed so that it can be easily enhanced to support new WS-I profiles. One way that this can be done is by implementing a core set of components, which are augmented by plugging in additional functions, which are used to process specific parts of a profile. The core function will include the basic required functions, such as processing the input parameters and producing the final conformance report.

2.3.1 Implementing the Analyzer Tool as a Web Service

In addition to implementing the analyzer as a tool which can be executed on a specific platform, the analyzer MAY be implemented as a Web service. The input to the analysis Web service would be either a set of files or a URL list, which could be used to access the files. The output from the analyzer service would be the conformance report.

2.4 Analyzer Configuration File

The analyzer tool MUST have a command line interface. When the analyzer tool is used on a command line, there SHOULD be at least one input option. This input option is used to reference the analyzer configuration file. The analyzer MAY be implemented to allow the use of a default configuration filename, so that the configuration file need not be specified on the command line. This is the format for the required command line interface:

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

	Option	Definition
1	-config	This option contains a reference to the analyzer configuration file. This can be a URL.

2.4.1 Analyzer Configuration File Format

The analyzer configuration file MUST contain the list of options for this tool. This file MAY also contain implementation specific configuration parameters.

The following two figures contain examples of analyzer configuration files. The first example references a Web service description in a WSDL document, and the second example references the service description in a UDDI entry.

```
<?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 >
  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 1. Example Analyzer Configuration File Using Direct WSDL 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>
   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 2. Example Analyzer Configuration File Using 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/2004/07/analyzerConfig/">
  <wsi-analyzerConfig:description>
    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: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 3. Example of Analyzer Configuration File Using UDDI Reference

Note: If the analyzer tool is deployed as a Web service, then this file could be used as one of the input messages for the service. The output would be a copy of the console log and the conformance report file.

The following table defines each of the elements that can be used in the conformance report file. A complete XML schema definition is in Section A 2.1 on page 47.

Element	Description	Attributes
<configuration>	The root element for the configuration file.	• name The name associated with the option settings in the configuration file.
<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". When running the analyzer tool on the command line, the verbose output is displayed on the console.	[None]
<assertionResults>	• This element is used to indicate the type of assertion results that should be listed in the conformance report.	• type The type of assertion results to include in the conformance report. The values for the type attribute have the following meaning: o all List the results from all test assertions. o notPassed List all of the assertion test results except the ones that have a result of passed. o onlyFailed List only the test assertion results which have a result

		of failed. o notInfo List the results from all non-informational test assertions.[2] 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. Note: The file extension for the report file SHOULD be ".xml" when XSLT will be used as the primary method to process this	• replace This attribute indicates whether the report file should be replaced if it already exists. The valid values for this attribute are "true" or

[2] Note that informational test assertions are used to identify extensibility points which are not in the scope of a profile and may or may not cause interoperability problems.

	file. If a unique file extension is needed for this file, then ".wsirpt" SHOULD be used.	"false". If the report file already exists and this attribute is set to "false", then the analyzer MUST terminate with an error message. The default value for this attribute is "false". • location The location where the report file should be created. This attribute is optional and the default report filename is "report.xml".
<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: If this element does not appear in the configuration file, then all of the test assertions that operate on the log entries will not be processed.	• correlationType Defines the type of correlation that should be used to determine which Web service a message is associated with a message in the log file. The default value is "operation". The valid values for the correlationType attribute are:

		o operation Correlation requires a match on the endpoint, namespace, a operation signature. o namespace The correlation process will use both the endpoint and namespace to match a message to a Web service. o endpoint A message is correlated to a Web service based only on the endpoint definition.
<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 MUST NOT be processed.	[None]
<wsdlElement>	This element contains the reference to the WSDL element that should be analyzed. This element MUST contain a reference to the type of element referenced by the type attribute	• type The type of WSDL element that is referenced by the name attribute. The following values can be specified on the type attribute. Each value corresponds to a WSDL element. o port o binding o portType o operation o message • 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 MUST NOT be specified with the <wsdlReference> element. Note: If this element does not appear in the configuration file, then the UDDI test assertions will not be processed. If neither this element nor the <wsdlReference> element appear in the configuration file, then the 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	[None]

	the UDDI bindingTemplate or tModel associated with the uddiKey element.	

2.4.2 Optional Analyzer Tool Command Line Syntax

An implementation of the analyzer tool MAY provide command line options in addition to those defined in Section 2.4, but the additional options are not required. For a user who is just starting to use this tool, command line options could be easier to use.

The additional command line options MUST 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.

```
analyzer –config <file-location>
         -verbose true | false
         -assertionResults all | notPassed | onlyFailed | notInfo
         -messageEntry true | false
         -assertionDescription true | false
         -failureMessage true | false
         -failureDetail true | false
         -reportFile <file-location>
         -replace true | false
         -addStyleSheet <href> [<type> [<title> [<media>
                        [<charset> [<alternate>]]]]]
         -testAssertionFile <file-location>
         -logFile <file-location>
         -correlationType <correlationType>
         -wsdlElement <name> <type> <namespace> [<parentName>]
          [-wsdlURI <file-location>
          [-serviceLocation <service-endpoint>] |
          -uddiKeyType <keyType> -uddiKey <keyValue>
          -inquiryURL <inquiryURL>]
```

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

	Option	Definition	Defaults
1	-config, -c	This option contains a reference to the analyzer configuration file.	
2	-verbose, -v	Display diagnostic messages on the console.	false
3	-assertionResults, -a	This option indicates the type of assertion results that should appear in the conformance report. The valid values for this option are: all, notPassed, onlyFailed, and notInfo.	notInfo
4	-messageEntry, -M	Include log entries in the report file.	true
5	-assertionDescription, -A	Include the assertion description in the report file.	false

6	-failureMessage, -F	Include the error messages that are pre-defined for each test assertion in the report file.	false
7	-failureDetail, -D	Include error detail messages in the report file.	true
8	-reportFile, -r	The name of the output conformance report file.	report.xml
9	-replace, -R	Indicates whether the report file should be replaced if it already exists.	false
10	-testAssertionFile, -t	The profile option identifies the profile definition that is used to validate the messages in the log file.	
11	-logFile, -l	The location of the message log file that was created by the message monitor tool. This option may contain a URL.	
12	-correlationType, -C	The type of message correlation function that should be used. The valid values are the same as those for the correlationType attribute on the <logFile> element in the analyzer configuration file.	operation
13	-wsdlElement, -w	The name, type, and namespace for the WSDL element to analyze.	
14	-parentElement, -p	If the WSDL element type specified with the –wsdlElement option is port or operation, then the parent element name is required. For port elements this would be the name of the service element, and for operation elements this would be the name of the portType element.	
15	-wsdlURI, -W	The Web service description for the service that is being analyzed. This option MUST be ignored if the –uddiKey option is specified.	
16	-serviceLocation, -S	If the WSDL service description does not contain a port element, then this option is used to specify the service endpoint.	
17	-uddiKeyType, -K	A reference to a type of UDDI entry. The valid values for key type are bindingKey or tModelKey.	
18	-uddiKey, -k	The value for the UDDI key.	
19	-inquiryURL, -i	The URL which is used to send	

		an inquiry to a UDDI registry.	

2.4.3 Using the <wsdlElement> Element

The <wsdlElement> element contains both a name and type attribute. The name attribute contains the name of the WSDL element that should be analyzed, and the type attribute indicates the type of WSDL element that should be analyzed. Since the element definitions in a WSDL document are referential, the element that is specified indicates the point at which the WSDL related analysis will begin.

For example, if the <wsdlElement> element references a <wsdl:port> element, then the analyzer will process the <wsdl:port>, the <wsdl:binding> referenced by the <wsdl:port>, the <wsdl:portType> referenced by the <wsdl:binding>, and the <wsdl:operation> and <wsdl:message> elements referenced by the <wsdl:portType>. If the <wsdlElement> element references a <wsdl:portType> element, then the only elements that will be analyzed are the <wsdl:portType> and the <wsdl;operation> and <wsdl:message> elements that it references.

The following table lists the WSDL elements that can be analyzed and the other WSDL elements that will be analyzed when that type of element is specified.

	WSDL Element	Other WSDL Elements Analyzed
1	port	binding, portType, operation, message
2	binding	portType, operation, message
3	portType	operation, message
4	operation	message
5	message	[none]

2.4.4 Using the <uddiReference> Element

The <uddiReference> element can be used to reference either a UDDI bindingTemplate or tModel. A bindingTemplate may contain a link to one or more tModels. When the <uddiReference> element contains a reference to a bindingTemplate, at least one of the tModels referenced by the bindingTemplate must be associated with a WSDL service description.

When processing the <uddiReference> element, if the <uddiKey> element references a bindingTemplate then both the bindingTemplate and the tModel with the reference to the WSDL service description MUST be processed. If the <uddiKey> element contains a reference to a tModel, then only the tModel MUST be processed. Since a tModel contains a reference to a <wsdl:binding>, the WSDL binding and the elements that it references (i.e. portType, operation and message elements) will also be processed by the analyzer.

2.4.5 Interpreting Combinations of Configuration Options

The test assertion document defines four primary artifacts: envelope, message, description, and discovery. The envelope and message artifacts correlate to the <logFile> element, while the description and discovery artifacts correlate to the <wsdlReference> and <uddiReference> elements, respectively. The following rules MUST be used when processing these configuration elements:

- 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 envelope, message 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 envelope, message, 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 MUST terminate after processing the configuration options.
- If a <uddiReference> element contains a <wsdlElement> element, then the type attribute value MUST be "binding". If it is not, then the analyzer MUST terminate after processing the configuration options.
- If a <serviceLocation> element is specified within a <wsdlReference> element, the <wsdlElement> element MUST contain a reference to either a <wsdl:port> or <wsdl:binding>. If it does not, then the analyzer MUST 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 MUST contain a reference to a <wsdl:binding>. If it does not, then the analyzer MUST 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" SHOULD 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 MUST terminate after detecting this condition.

2.5 Profile Test Assertions Document

The WS-I Basic Profile working group defines the profile definition file [4]. The Testing Tools working group translates the requirements into test assertions. A profile test assertions document [2] MUST contain all of the information that is required to validate conformance of artifacts to the corresponding WS-I basic profile definition and that is required by the analyzer architecture. A test assertion is an encoding of requirements defined in the profile document. They can represent part of a requirement, a single requirement, or more than one requirement. The test assertion document MUST be used by the analyzer to control its processing.

2.5.1 Term Definitions

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

Primary Entry Type: The primary entry type for a test assertion is the entry type that is singled out by the context, as being the main object of the test assertion. An instance of the primary entry type for this test assertion is called a primary entry for this test assertion. The test assertion result (i.e. its conformance statement) will be about primary entries (even though 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 instances of the primary entry type in the input material to the Analyzer. A test assertion definition MUST define a unique primary entry type.

Secondary Entry Type: A secondary entry type is any entry that is required to process a test assertion, but is not the primary target of the test assertion. A test assertion MAY define one or more secondary entry types.

Qualified Instance: Test material artifact entry that matches the context of a test assertion.

Context: Intuitively, the context of a test assertion defines which test material artifacts are relevant to – or qualified 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 to qualify and process the test assertion. 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.

Assertion Description: The assertion description for a test assertion is the actual profile requirement to which a qualified entry is expected to conform.

Pre-requisites: A test assertion may refer to prerequisite 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.

Referenced Profile Requirements: A test assertion (TA) normally has references to one or more profile requirements, labeled as Rxxxx in a profile definition. Each profile requirement that is referenced by a TA falls into one of three 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.)

- Partial-target profile requirement: This is equivalent to the "target" profile requirement above. However this type indicates that the assertion does not fully cover all possibilities of the profile requirement. That is, the assertion tests some subset of the requirement and, as such, cannot fully verify that some entry material strictly conforms to the target requirement.

- Collateral profile requirement: This type of profile requirement is NOT verified by the TA, but instead, can be assumed by someone implementing the TA. In other words, the verification procedure for this TA can be designed assuming that the material in input will satisfy the collateral requirements (if it is a MUST) or may satisfy (if it is a MAY or SHOULD). The difference with "pre-requisites" as defined in this specification is that collateral requirements may concern other material than the primary entry for the TA. For example, verifying that a UDDI tModel has the expected category, will assume that another tModel be present to represent this category (another profile requirement), which will condition the verification. If this collateral requirement is not satisfied (e.g. if an associated test assertion has failed in the report), then the outcome of the assuming TA should not be relied upon, or in case of failure, the collateral requirement should be investigated as a possible cause. It may also be that the collateral requirements are not verified by any TA, but are just mentioned for a better understanding of the target requirement. This is often the case when the collateral requirement is a MAY or a SHOULD, in which case it means that the verification of the target requirement MUST handle such possibilities.

Referenced Extensibility Points: A test assertion (TA) may also reference an identified extensibility point, labeled as Exxxx in a profile definition. These extensibility points are not in the scope of the profile but still may or may not cause interoperability problems. The assertion related to an extensibility point is referred to as "informational" and can have a result of either passed or notApplicable. An entry for an informational assertion is said to pass the test assertion if some entry material actually contains the extensibility point.

2.5.2 Interpretation of a Test Assertion

The following terms are used when describing how to interpret a test assertion for a set of entries:

- A primary entry for a test assertion is said to qualify for the test assertion (or be applicable) if the primary entry (and its related secondary entries, if any) satisfies the context of the test assertion, and also satisfies the pre-requisites for the test assertion. The test assertion will only be verified (its assertion condition evaluated) on qualified entries.

- A primary entry for a test assertion is said to pass the test assertion if the primary entry is qualified for the test assertion, and primary entry (and its related secondary entries, if any) also satisfies the condition of this test assertion.

- A primary entry for a test assertion is said to fail the test assertion if the primary entry is qualified for this test assertion, and the primary entry (and its related secondary entries, if any) does NOT satisfy the condition of this test assertion.

2.5.3 Test Assertion Results

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

- passed
  A test assertion completed its check without detecting any errors.
- failed
  A test assertion detected an error.
- warning
  A test assertion failed, but the type attribute for the test assertion indicated that it was “recommended”, not “required”. This type of failure will not affect the overall conformance result.
- notApplicable
  The test assertion was not processed because it did not match the assertion context.
- prereqFailed
  The test assertion was not processed because a prerequisite test assertion failed.
- missingInput
  The test assertion was not processed based on the input parameters that were specified (i.e. the input required to process the test assertion was missing), or the entry does not exist in the specified artifacts (e.g. a WSDL document does not contain a <types> element).

2.5.4 Interpretation of Prerequisite Test Assertions

A test assertion may contain a reference to one or more prerequisite test assertions.

- If any of the prerequisite test assertions has a result of failed, then the current test assertion MUST NOT be processed and MUST be assigned a result of prereqFailed.

- If any of the prerequisite test assertions has a result of missingInput, then the current test assertion MUST NOT be processed and MUST be assigned a result of missingInput.

- If any of the prerequisite test assertions is informational and has a result of passed, then the current test assertion MUST NOT be processed and MUST be assigned a result of notApplicable. Note that a passed result of an informational assertion indicates that some entry material actually contains the extensibility point and as such is beyond the scope of the current profile.

2.5.5 Interpretation of Additional Entry Types in a Test Assertion

A test assertion will always contain a primary entry type, but in some cases additional entry instances will be needed to process a test assertion. The additional entry types are specified in the <additionalEntryType> element. When processing a test assertion, if the additional entry types were not provided as input to the analyzer, then the test assertion MUST NOT be processed and it MUST be given a result of notApplicable.

2.5.6 How to Handle the State for Different Input Artifacts

There are certain situations where the input artifacts for the analyzer are in a non-typical state, such as an empty log file or incomplete WSDL description. The following table contains a list of input artifact states and a definition of how the analyzer should handle these states.

	Input Artifact State	Method to Handle this State
1	Log file with no <messageEntry> elements. This could happen when the monitor is started and stopped without receiving any messages.	The test assertions with an envelope or a message as the primary target should have a result of missingInput.
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 should terminate with an error message which indicates that the WSDL document did not contain the target WSDL element.
3	The analyzer configuration file contains a reference to a UDDI entry, but the UDDI entry does not exist.	The analyzer should 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 is needed. 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 should 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 should 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	The test assertions which contain an entry type that matches the elements that aren't processed MUST have a result of missingInput.

	element, then the binding element will not be processed.	
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 an entry type of "types".	The test assertions which contain an entry type which matches the missing elements MUST have a result of missingInput.

2.5.7 Test Assertions Document Format

The format of the test assertions document follows that used by the profile definition. Since the profile definition document requirements are grouped by artifacts, the test assertion document is also grouped by artifacts. Each artifact section contains a list of specification references and a list of test assertions.

The following figure contains a portion of the profile test assertion document for the Basic Profile.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--?xml-stylesheet type="text/xsl" href="assertions.xsl"?-->
<profileAssertions name="Basic Profile Test Assertions" version="1.0"
   xmlns="http://www.ws-i.org/testing/2004/07/assertions/" >

  <description>
   This document contains the test assertions for the
   WS-I Basic Profile definition.  These test assertions are
   used by the analyzer testing tool to determine if
   a Web service is conformant to the Basic Profile.
  </description>

  <profileList>
    <profile id="BP1" name="Basic Profile" version="1.0" revision="10"
       location="Use of this WS-I Material is governed by the WS-I Test License at
http://ws-i.org/licenses/test_license.html"/>
  </profileList>

  <artifact type="discovery">
    <specificationReferenceList>
      <specification name="The UDDI Version 2.04 API Published Specification, Dated 19
July 2002"
       location="http://www.uddi.org/pubs/ProgrammersAPI-V2.04-Published-
20020719.pdf"/>
      <specification name="UDDI Version 2.03 Data Structure Reference, Published
Specification, Dated 19 July 2002"
       location="http://www.uddi.org/pubs/DataStructure-V2.03-Published-
20020719.pdf"/>
```

```
    <specification name="Version 2.0 UDDI XML Schema 2001"
     location="http://www.uddi.org/schema/uddi_v2.xsd"/>
   </specificationReferenceList>
   <description>
     The Basic Profile requires support for UDDI V2.0.
   </description>

   <testAssertion id="WSI5001" entryType="tModel" type="required" enabled="false">
     <context>For each web service definition that is published in a UDDI
registry.</context>
     <assertionDescription>A UDDI tModel element which represents a conformant Web
service
     type must use WSDL as the description language.  The uddi:overviewDoc element
     must contain a uddi:overviewURL element which contains a reference to
     a conformant WSDL binding.  The uddi:overviewURL in a uddi:tModel must use the
convention defined in the
UDDI best practices document to resolve to a wsdl:binding.</assertionDescription>
     <failureMessage>The UDDI tModel does not reference a WSDL based Web service
     definition.</failureMessage>
     <failureDetailDescription>tModel key</failureDetailDescription>
     <additionalEntryTypeList>
       <messageInput>none</messageInput>
       <wsdlInput>none</wsdlInput>
     </additionalEntryTypeList>
     <prereqList/>
     <referenceList>
       <reference profileID="BP1">R3002</reference>
     </referenceList>
     <comments></comments>
   </testAssertion>
   <!-- Other test assertions for the discovery artifact would go here -->
  </artifact>
  <!-- Other artifacts would appear here  -->
</profileAssertions>
```

Figure 4. Example Profile Test Assertion Document

The elements in this document are defined in the following table. A complete XML schema definition is in Section A 2.2 on page 49.

Element	Description	Attributes
<profileAssertions>	The root element for the profile test assertions document.	• name The name associated with the profile test assertion document. • version The version number for the test assertion document.

<description>	A text description of the parent element.	[None]
<profileList>	Contains a list of references to profile definitions. Each profile definition in the list must have a unique identifier.	[None]
<profile>	This element contains a reference to the profile that contains the requirements that were used to create the test assertions.	• id A unique identifier that can be used to reference the profile definition from other parts of this document. • name The name of the profile definition. • version The version of the profile definition. • revision The revision of the profile version. • location The source of the profile definition document.
<artifact>	This element is used to reference an artifact that is defined in the profile definition document. For a basic profile, the type attribute may have one of the following values: • envelope The primary target for the test assertions is the serialization of the soap:Envelope element and its content. • message The primary target for the test assertions is the transport and message protocol. • description The primary target for the test assertions is a WSDL-based service description. • discovery	• type The type of artifact.

	The test assertions for this artifact will focus on the entries in a UDDI registry.	
<specficationReferenceList>	Contains a list of specifications for this artifact which are referenced by the profile definition.	[None]
<specification>	A reference to a single specification.	• name The name of the specification. • location The location of the specification related document.
<testAssertion>	A test assertion defines one item that needs to be validated for a specification. Note: The valid values for the type attribute are listed below by artifact type. message These entry types correspond to the type of message in a log file. • requestMessage • responseMessage • anyMessage Either a request or a response message. envelope • requestEnvelope • responseEnvelope • anyEnvelope An envelope within either a request or a response message.. description All of the entry types correspond to WSDL elements. • definitions • import • types • message • operation • portType • binding • port	• id The unique identifier for this test assertion. • entryType The primary entry type which is based on the artifact that contains this element. • type The type of assertion. The valid values are "required", "informational" or "recommended". • enabled A boolean value which indicates if the test assertion should be processed by the analyzer. The attribute should be set to false when a new test assertion is added and it hasn't been implemented by the analyzer yet. The default value for this attribute is "false".

	discovery The entry types for this artifact correspond to the type of UDDI entries. • bindingTemplate • tModel	
<context>	Defines the conditions that must exist before executing a test assertion.	[None]
<assertionDescription>	A text description of the functions that must be performed by a test assertion.	[None]
<failureMessage>	The exact message that can be displayed when a test assertion fails.	[None]
<failureDetailDescription>	This element contains a description of the failure details that SHOULD be included in the conformance report. The failure detail messages MAY contain implementation specific information, such as XML parser error messages.	[None]
<detailDescription>	This element contains a description of the identified extensibility point. This is used solely for informational assertions.	
<additionalEntryTypeList>	This element contains a list of additional entry type input information, or the secondary entry types. The child elements within this element identify the data that is required by a test assertion in addition to the primary entry type. Any or all of the three child elements can be specified. This element is optional.	[None]
<messageInput>	Identifies the type of message log file entry that should be processed by a test assertion as a secondary entry. One of the following values can be specified. • none No log input is required by the test assertion.	[None]

	• anyMessage • requestMessage • responseMessage • anyEnvelope • requestEnvelope • responseEnvelope	
<wsdlInput>	Defines the component within the WSDL document which is the secondary entry type for the test assertion. One of the following values can be specified. Except for "none", these values correspond to an element in a WSDL document. • none No WSDL input is required by this test assertion. • definitions • import • types • message • operation • portType • binding • port	[None]
<prereqList>	This element contains a list of references to prerequisite test assertions that must be processed on the same or related entries before the current test assertion is processed.	[None]
<testAssertionID>	The identifier for a prerequisite test assertion.	[None]
<referenceList>	Contains a list of references to requirements listed in the profile definition document. The profile definition document MUST be referenced by a <profile> element in the <profileList>.	[None]
<reference>	A reference to a single requirement in a profile definition document. The role attribute may have one of the following values: • target The requirement is	• profileID A reference to an identifier that was specified on a <profile> element within the <profileList> element.

	verified by the test assertion. • partial-target The requirement is partially verified by the test assertion. • collateral The requirement is NOT verified by the test assertion, but can be assumed by a test assertion implementation.	• role The role defines the purpose of the requirement reference. The default value for this attribute is "target".
<comments>	Free-format text comments that provide additional details about a test assertion.	[None]

2.6 Message Log File

The message log file is defined in the Message Monitor functional specification document [1]. The message log file contains one entry for each request or response message that is sent between a client application and a Web service.

2.7 Profile Conformance Report

The analyzer tool produces one output file. This file contains the conformance report for a Web service. This report contains the conformance test results for the test assertions that have been processed. The conformance report also details the conformance level for each test assertion that was processed, and may list detailed information for any errors that were encountered. The report also contains a summary of the test assertions results. This summary will indicate if the artifacts related to Web service passed or failed the profile conformance test.

The format of the conformance report has been designed so that the contents can be written out as the analyzer tool does its processing. For example, after each entry is processed by an individual test assertion, the result for the test assertion can be written out to the report file.

2.7.1 Conformance Report Format

The following figure contains an example of a profile conformance report file.

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="2004-10-25T16:56:56.905Z"
   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="0.91" releaseDate="2004-10-25">
    <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="1.4.2"/>
      <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-analyzerConfig: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="missingInput">
      </assertionResult>
      <assertionResult id="WSI3003" result="missingInput">
      </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="2004-10-25T16:06:03.605Z">
    <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="2004-10-25T14:20:51.234Z">
   <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 5. Example Profile Conformance Report

The following table defines each of the elements that can be used in the conformance report file. A complete XML schema definition is in Section A 2.2 on page 49.

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 MUST contain 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 SHOULD contain an indication of the analyzer version. This can be done using a date or a version number. Here are two examples of how this can be done: http://hostname/2004/10/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.	
<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 MUST match 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 in the monitor configuration section of the log file if it is present.	• timestamp The timestamp from the message log file or the date and time for the WSDL file.
<wsi-monConfig:comment>	The comment element that is the first child of the configuration element in the message log file.	
<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: • requestEnvelope • responseEnvelope • anyEnvelope • 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 must include 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. The <wsi-log:messageEntry> element is defined in the Message Monitor Document [1].	
<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.	• id Test assertion identifier. This value must match the value that is listed in the profile definition document. • result

	• failed The test assertion detected an error. A description of the error MUST be put 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. • prereqFailed The test assertion was not processed because a prerequisite test assertion failed. • missingInput The test assertion was not processed based on the input parameters that were specified, or an element of an artifact is missing.	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 must be 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 MUST contain at least one <failureDetail> elements as a sub-element. The < failureDetail > element MUST indicate the source of the error, and contain instructions on how to correct the error.	[None]

2.7.2 Processing Test Assertions

This section provides additional information on how to process test assertions.

- All test assertions are associated with an artifact definition. Since each artifact correlates to an input option defined in the analyzer configuration file, the artifact's test assertions are only processed against the entries obtained from the input option.
- An artifact defines a class of entry types. For example, the message artifact has request and response entry types. All of the test assertions for an artifact are directly associated with the primary entry type defined in the test assertion. When processing an entry instance, its entry type must match the primary entry type for a test assertion before a test assertion can be processed. If the entry type of an entry instance does not match the primary entry type for a test assertion, then the test assertion MUST NOT be processed.
- All of the test assertions for an artifact MUST have a result of missingInput when the configuration option associated with the artifact is not specified, or when the input artifact does not contain any data to analyze. For example, all of the test

assertions for the message artifact will have a result of missingInput if the log file option was not specified, or if the log file does not contain any entries.

- Test assertions with a result of missingInput MUST be listed in the conformance report within an <entry> element. If the result for all test assertions within an artifact is missingInput, then the <entry> element should have only a type attribute with a value that is the artifact type name prefixed with "[" and suffixed with "]". When an artifact has a mixture of results, then the <entry> elements that contain the missingInput results should have both a type and referenceID attribute. The referenceID attribute should contain the entry type name prefixed with "[" and suffixed with "]".
- A test assertion that requires more than one input, but with one not available MUST have a result of notApplicable.
- The test assertions for the description artifact that have a primary entry type of definitions, import or types MUST be processed once for each WSDL document. For example, if the WSDL document referenced by the analyzer configuration file contains two import elements, these types of test assertions must be processed three times (once for initial WSDL document and once for each document referenced by an import element).

2.7.3 Entry Reference Identifier

The referenceID attribute on the <entry> element MUST contain a reference to the entry instance that was analyzed when the test assertion results for the entry is NOT missingInput. The value of the referenceID attribute will vary depending upon the type of artifact that is being processed.

- discovery
  For an entry type of bindingTemplate, the referenceID value MUST be the bindingTemplate key. If the entry type is tModel, then the referenceID MUST be 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 MUST be the value of the name attribute on the element for the entry instance. For the binding, portType, and message entry types, the referenceID value MUST be the QName for the element associated with the entry instance. When the entry type is definitions, the referenceID value must be the location of the WSDL document. For an entry type of import, the referenceID value MUST be the value from the namespace attribute on the <import> element. For the types entry type, the referenceID value MUST be the location of the WSDL document with "-Types" appended to it.
- message and envelope
  The referenceID value MUST always be the entry identifier for the message log entry.

2.7.4 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 should contain the following text: "Additional entry missing".

2.7.5 Test Assertion Summary Results

The following terms are used when describing the report summary of a test assertion over a set of artifacts:

- The result summary of processing a Test Assertion (TA) over a set of artifacts is notApplicable if either there was no entry usable as a primary entry for this TA, or no primary entry was qualified for this TA.
- The result summary of processing a Test Assertion (TA) over a set of artifacts is passed if there were qualified primary entries for this TA, and they all passed the TA.
- The result summary of processing a Test Assertion (TA) over a set of artifacts is failed if there were qualified primary entries for this TA, and at least one failed the TA.

2.7.6 Report Summary Result

The following terms are used when describing the general report summary of a set of test assertions over a set of artifacts:

- The summary result for processing a set of test assertions over a set of artifacts is passed if, for each test assertion, the summary result was either passed, warning or notApplicable.

- The summary result for processing a set of test assertions over a set of artifacts is failed if, for at least one test assertion, the summary result was failed.

2.7.7 Special Considerations when Building a Conformance Report

The conformance report was designed to contain all of the analysis information that is produced by the analyzer tool. The errors that are typically listed in the conformance report are identified when processing test assertions. But there are other errors that may be encountered by the analyzer tool, which will cause it to terminate. These errors MUST also be listed in the conformance report.

The following guidelines should be used to ensure that the conformance report produced by the analyzer tool contains a well-formed XML document and additional information that explains why the analysis process was terminated.

- All input options must be validated before any test assertions are processed. This includes verifying that all of the input files are accessible. If an input option is not valid or if one or more input files are not accessible, then a conformance report MUST be created that contains the following elements: <report>, <analyzer>, <analyzerFailure>. The <analyzerFailure> element MUST contain at least one <failureDetail> element, which identifies the error and includes information on how to correct the error.

- The conformance report was designed so that it can be written out as test assertions are being processed. If an error is detected while the test assertions are being processed, then the conformance report must be completed so that it is a well-formed XML document. This includes terminating any XML elements that have already been written out to the report file, adding a <analyzerFailure> element that identifies the source of the error, and closing the <report> element.

2.7.8 Comparing Conformance Reports from Different Tool Implementations

When comparing conformance reports that are generated by different implementations of the analyzer tool for the same set of artifact instances, the <analyzer> and <failureDetail> elements should be ignored. These elements contain information that is specific to the implementation of the tool.

2.8 Analyzer Validation Process

The analyzer tool MUST process all of the enabled test assertions, which are defined in the Profile Test Assertion Document. Before processing any test assertions, the analyzer MUST verify that all enabled test assertions can be run. If all of the enabled test assertions can not be run, then a report is generated with an <analyzerFailure> element which describes why the analysis could not processed.

The analyzer tool will have the following primary functions:

- Process Control
  This function will control all processing within the analyzer. It will parse the input configuration options and then pass control to the other functions.

- UDDI Entry Validation
  This function will be used only when the input options indicate that the Web service is defined in a UDDI registry. This function MUST validate the UDDI entries for the Web service, to ensure that it conforms to the specification in the profile.

- Web Service Description Validation
  Before processing any of the message log entries, the Web service description must be validated. This function MUST process both the WSDL service description and any XML schema documents that contain data type definitions.

- Validation Functions for each Message Log Entry
  A message log file may contain messages for more than one Web service. This function must correlate the messages with the specified Web service definition, and only process the messages which are associated with the Web service. The message correlation process is described in detail in the next section.

  Each of the message log entries are processed one at a time. For each entry that matches the Web service definition, first the transport data and message protocol is validated and then the SOAP:envelope element and its content is validated.

- Build the Conformance Report
  The only output from the analyzer tool is a conformance report. As the test assertions are being processed, the conformance report is built based on the output from the validation functions.

The following figure provides an overview of these functions.

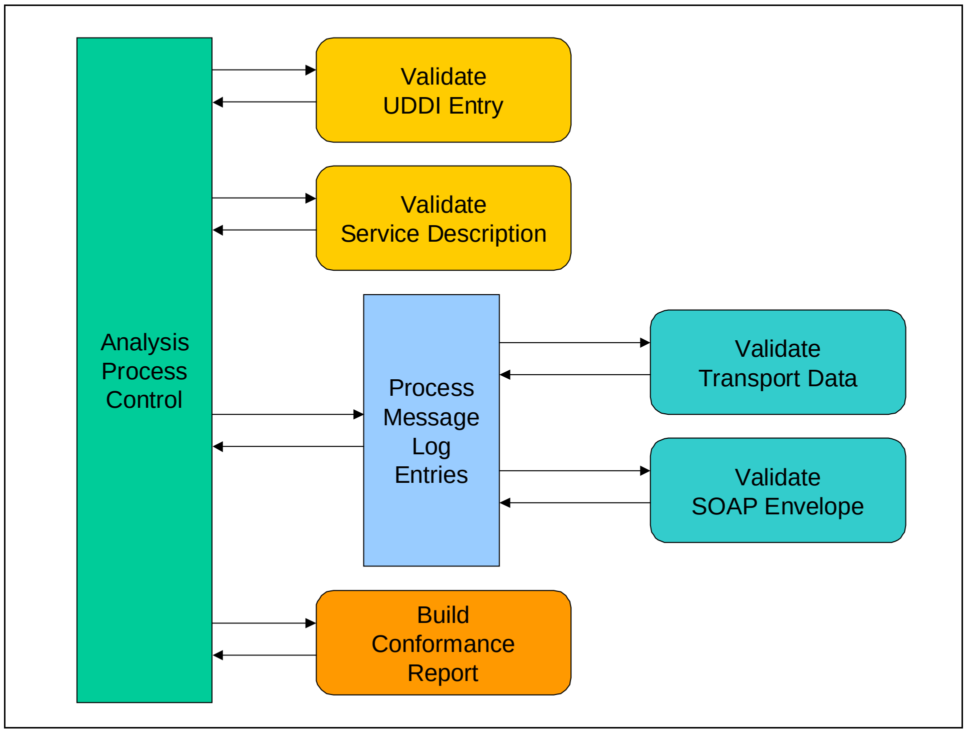

Figure 6. Analyzer Validation Process

2.8.1 Processing WSDL Import Statements

All WSDL import elements MUST be processed before processing any test assertions. The import related test assertions should be processed before the other types of test assertions, which mean that they SHOULD appear at the beginning of the description section of the test assertion document.

2.8.2 Processing Messages from the Log File

When processing messages in from the log file, the following process MUST be used to determine the message encoding:

- When the analyzer processes a message entry from the log file, it MUST first look at the BOM attribute on the <messageContent> element to determine the message encoding.
- If the BOM attribute is not present, then the XML declaration SHOULD be used to determine the encoding.
- If there is no BOM attribute or XML declaration, then the message SHOULD be UTF-8 encoded.
- The charset value in the Content-Type header will be validated against the BOM attribute value and XML declaration encoding value by one of the test assertions.
- Before parsing the message content (e.g. SOAP envelope), the XML declaration MUST be removed if leaving this line in would cause a parser error.

2.8.3 Message Correlation Process

Since a single message log file can contain messages for more than one Web service, this section defines the method used by the analyzer to correlate a message with a Web service. All of the messages in the log file MUST be run through the correlation process before processing any of the message-related test assertions. If a message does not correlate to the specified Web service, then it MUST NOT be processed. This means that the conformance report will not contain an assertion result for any test assertion that did not pass the correlation process.

The analyzer is designed to provide a conformance report for a single Web service. To support messages from multiple web services in one log file, the analyzer MUST be able to correlate a log entry with a Web service.

To do this, the analyzer uses input options which clearly identify the Web service that is being analyzed. In addition, the message correlation process requires the specification of an option, which indicates how extensive the correlation process should be. The values for the type of message correlation process are:

- endpoint – correlate messages based on endpoints only
- namespace – correlate messages using the endpoint and the namespace in the message
- operation – process all message correlation functions (endpoint, namespace, and operation)

Since a Web service definition is not needed for all of the log entry related test assertions, the analyzer should process all of the test assertions that don't need it and then do the correlation assertions. This would help ensure a fairly consistent and valid correlation process.

Here are the fragments of information in the log file that can be used to help correlate a message with a web service definition.

1. The host and port for the Web service endpoint. This information is in either the <senderHostAndPort> or the <receiverHostAndPort> depending upon the type of message. If the message is a request message, then it will be in the <receiverHostAndPort> element. If the message is a response to a request, then the host and port will be in the <senderHostAndPort> element.
2. URL path for the web service endpoint (in the HTTP POST header for a request message)
3. Namespaces from the SOAP message content
4. Operation information (derived from message content)

Based on the information listed above, the following steps MUST be used to correlate a Web service definition with a log entry. The primary issue with this process is that steps 3 and 4 require information from the SOAP envelope which could be invalid.

1. For each request message, match the host and port information for the Web service to the information in the <senderHostAndPort> or <receiverHostAndPort>.
    a. If the message doesn't match, then the request message and its associated response MUST NOT be processed.
    b. If the message does match, then go to the next step.

2. For a request message, use the host, port, and URL path (from the POST header) to associate the log entry with an endpoint in a WSDL document or UDDI entry.
    a. If the endpoint does not match, then the request message and its associated response MUST NOT be processed.
    b. If there is a match and the correlation type option is "endpoint", then process both the request and response message.
    c. Otherwise go to the next step. (Note: Since more than one service could be available on one endpoint, there is no way to determine if a Web service is correlated to a message based on analyzing just the endpoint.)

3. Using the namespaces declared in the request message within the SOAP body element, determine if the service definition declares it. When the specified WSDL binding contains document-literal operation definitions, the namespace is the targetNamespace of the schema that defines the element which is the child of the soap:Body. For rpc-literal operations, the namespace of the child element within the soap:Body is the value of the namespace attribute of the soapbind:body element within the wsdl:input element. All operations in the specified binding should be searched for a namespace match.
    a. If there is no match, then the request message and its associated response MUST NOT be processed.
    b. If there is a match and the correlation type option is “namespace”, then process both the request and response message.
    c. Otherwise go to the next step. (Note: The correlation process could be stopped here, but there is no way to ensure that a namespace is used only once.)

4. Determine if the request operation name in the specified WSDL binding definition matches the request message content. For document-literal operations, the QName for the child element within the soap:Body is used to search for a WSDL message part definition that has the same name. For rpc-literal operations, the name for the child element within the soap:Body is used to search for an operation definition with the same name. If an operation is found, then the namespace from the child element must match the value of the namespace attribute on the soapbind:body element within the wsdl:input element for that operation definition.
    a. If there is no match, then the request message and its associated response MUST NOT be processed.
    b. If there is a match, then process the remaining test assertions for this message.

A 1 References

[1] Monitor Tool Functional Specification
http://www.ws-i.org/Testing/Specs/MonitorFunctionalSpecification_1.02.pdf

[2] Testing Tool Test Assertion Document (TAD) Version 1.0,
http://www.ws-i.org/Testing/Tools/2004/01/BasicProfileTestAssertions.xml

[3] Using WSDL in a UDDI Registry Best Practices,
http://www.uddi.org/bestpractices.html

[4] Basic Profile Version 1.0, Final Material,
http://www.ws-i.org/Profiles/BasicProfile-1.0-2004-04-16.html

[5] Basic Profile Version 1.1, WS-I Approval Draft,
http://ws-i.org/Profiles/BasicProfile-1.1-2004-07-21.html

[6] Simple Soap Binding Profile Version 1.0, WS-I Approval Draft,
http://ws-i.org/Profiles/SimpleSoapBindingProfile-1.0-2004-07-21.html

[7] Attachments Profile Version 1.0, WS-I Approval Draft,
http://ws-i.org/Profiles/AttachmentsProfile-1.0-2004-07-21.html

A 2 Schemas

This section contains the XML schema definitions for the XML documents used and created by the analyzer test tool.

A 2.1 Analyzer Configuration File Schema

This section contains the XML schema definition for the analyzer configuration file.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://www.ws-i.org/testing/2004/04/analyzerConfig/"
   xmlns:tns="http://www.ws-i.org/testing/2004/04/analyzerConfig/"
   xmlns:uddi="urn:uddi-org:api_v2"
   xmlns:wsi-common="http://www.ws-i.org/testing/2003/03/common/"
   elementFormDefault="qualified">

  <xs:import namespace="http://www.ws-i.org/testing/2003/03/common/" />

  <xs:element name="configuration" type="tns:configuration" nillable="true"/>
  <xs:complexType name="configuration">
    <xs:sequence>
          <xs:element name="description" type="wsi-common:description"
minOccurs="0" maxOccurs="1"/>
      <xs:element name="verbose" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
      <xs:element name="assertionResults" type="tns:assertionResults"/>
      <xs:element name="reportFile" type="tns:reportFile"/>
      <xs:element name="testAssertionsFile" type="xs:anyURI"/>
      <xs:element name="logFile" type="tns:logFile" minOccurs="0" maxOccurs="1"/>
      <xs:choice minOccurs="0" maxOccurs="1">
        <xs:element name="wsdlReference" type="tns:wsdlReference"/>
        <xs:element name="uddiReference" type="tns:uddiReference"/>
      </xs:choice>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="optional"/>
  </xs:complexType>

  <xs:complexType name="assertionResults">
    <xs:attribute name="type" use="optional" default="all">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="all"/>
          <xs:enumeration value="onlyFailed"/>
          <xs:enumeration value="notPassed"/>
          <xs:enumeration value="notInfo"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="messageEntry" type="xs:boolean" use="optional"
default="true"/>
    <xs:attribute name="assertionDescription" type="xs:boolean" use="optional"
           default="false"/>
   <xs:attribute name="failureMessage" type="xs:boolean" use="optional"
          default="true"/>
   <xs:attribute name="failureDetail" type="xs:boolean" use="optional"
          default="true"/>
  </xs:complexType>

  <xs:complexType name="reportFile">
   <xs:sequence>
    <xs:element name="addStyleSheet" type="wsi-common:addStyleSheet"
minOccurs="0" maxOccurs="1"/>
   </xs:sequence>
   <xs:attribute name="location" type="xs:anyURI" use="optional"
default="report.xml"/>
   <xs:attribute name="replace" type="xs:boolean" use="optional" default="false"/>
  </xs:complexType>

  <xs:complexType name="logFile">
   <xs:simpleContent>
    <xs:extension base="xs:anyURI">
     <xs:attribute name="correlationType" use="optional" default="operation">
      <xs:simpleType>
       <xs:restriction base="xs:string">
        <xs:enumeration value="operation"/>
        <xs:enumeration value="namespace"/>
        <xs:enumeration value="endpoint"/>
       </xs:restriction>
      </xs:simpleType>
     </xs:attribute>
    </xs:extension>
   </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="wsdlReference">
   <xs:sequence>
    <xs:element name="wsdlElement" type="tns:wsdlElementReference"/>
    <xs:element name="wsdlURI" type="xs:anyURI"/>
    <xs:element name="serviceLocation" type="xs:anyURI" minOccurs="0"
maxOccurs="1"/>
   </xs:sequence>
  </xs:complexType>

  <xs:complexType name="wsdlElementReference">
   <xs:simpleContent>
    <xs:extension base="xs:NCName">
     <xs:attribute name="type" type="tns:wsdlElementType" use="required"/>
     <xs:attribute name="namespace" type="xs:anyURI" use="required"/>
     <xs:attribute name="parentElementName" type="xs:NCName" use="optional"/>
    </xs:extension>
   </xs:simpleContent>
  </xs:complexType>

  <xs:simpleType name="wsdlElementType">
```

```
    <xs:restriction base="xs:string">
      <xs:enumeration value="port"/>
      <xs:enumeration value="binding"/>
      <xs:enumeration value="portType"/>
      <xs:enumeration value="operation"/>
      <xs:enumeration value="message"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="uddiReference">
    <xs:sequence>
      <xs:element name="uddiKey" type="tns:uddiKey"/>
      <xs:element name="inquiryURL" type="xs:anyURI"/>
      <xs:element name="wsdlElement" type="tns:wsdlElementReference"
minOccurs="0" maxOccurs="1"/>
      <xs:element name="serviceLocation" type="xs:anyURI" minOccurs="0"
maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="uddiKey">
    <xs:simpleContent>
      <xs:extension base="xs:NMTOKEN">
        <xs:attribute name="type" use="required">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="bindingKey"/>
              <xs:enumeration value="tModelKey"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:schema>
```

A 2.2 Profile Test Assertions Document Schema

This section contains the XML schema definition for the profile test assertion document.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://www.ws-i.org/testing/2004/07/assertions/"
   xmlns:tns="http://www.ws-i.org/testing/2004/07/assertions/"
   xmlns:wsi-common="http://www.ws-i.org/testing/2003/03/common/"
   elementFormDefault="qualified">

  <xs:import namespace="http://www.ws-i.org/testing/2003/03/common/"
schemaLocation="common.xsd"/>

  <xs:element name="profileAssertions" type="tns:profileAssertions"/>
  <xs:complexType name="profileAssertions">
   <xs:sequence>
     <xs:element name="description" type="wsi-common:description"/>
     <xs:element name="profileList" type="tns:profileList"/>
     <xs:element name="editors" type="tns:editorsType" minOccurs="0"/>
     <xs:element name="contributorText" type="xs:string" minOccurs="0"/>
     <xs:element name="artifact" type="tns:artifact" minOccurs="1"
maxOccurs="unbounded"/>
   </xs:sequence>
   <xs:attribute name="name" type="xs:string" use="required"/>
   <xs:attribute name="version" type="xs:string" use="required"/>
   <xs:attribute name="date" type="xs:date" use="optional"/>
   <xs:attribute name="status" type="xs:string" use="optional"/>
  </xs:complexType>

<xs:complexType name="editorsType">
     <xs:sequence>
         <xs:element name="person" type="tns:personType" minOccurs="0"
maxOccurs="unbounded"/>
     </xs:sequence>
  </xs:complexType>
  <xs:complexType name="personType">
     <xs:simpleContent>
     <xs:extension base="xs:string">
       <xs:attribute name="affiliation" type="xs:string" use="optional"/>
       <xs:attribute name="href" type="xs:anyURI" use="optional"/>
     </xs:extension>
   </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="profileList">
   <xs:sequence>
     <xs:element name="profile" type="tns:profile" minOccurs="1"
maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:complexType>

  <xs:complexType name="profile">
   <xs:attribute name="id" type="xs:ID" use="required"/>
   <xs:attribute name="name" type="xs:string" use="required"/>
   <xs:attribute name="version" type="xs:string" use="required"/>
   <xs:attribute name="revision" type="xs:string" use="required"/>
   <xs:attribute name="location" type="xs:anyURI" use="required"/>
  </xs:complexType>

  <xs:complexType name="artifact">
   <xs:sequence>
     <xs:element name="specificationReferenceList"
type="tns:specificationReferenceList"/>
     <xs:element name="description" type="wsi-common:description"/>
     <xs:element name="testAssertion" type="tns:testAssertion" minOccurs="1"
maxOccurs="unbounded"/>
```

```
    </xs:sequence>
    <xs:attribute name="type" type="tns:artifactTypes" use="required"/>
  </xs:complexType>

  <xs:complexType name="specificationReferenceList">
    <xs:sequence>
      <xs:element name="specification" type="tns:specification" minOccurs="1"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="specification">
    <xs:attribute name="name" type="xs:string" use="required"/>
    <xs:attribute name="location" type="xs:anyURI" use="required"/>
  </xs:complexType>

  <xs:element name="abstractDescription" type="xs:string" abstract="true"/>
<xs:element name="failureDetailDescription" type="xs:string"
    substitutionGroup="tns:abstractDescription"/>
<xs:element name="detailDescription" type="xs:string"
    substitutionGroup="tns:abstractDescription"/>

  <xs:complexType name="testAssertion">
    <xs:sequence>
      <xs:element name="context" type="xs:string"/>
      <xs:element name="assertionDescription" type="xs:string"/>
      <xs:element name="failureMessage" type="xs:string" minOccurs="0"
        maxOccurs="1"/>
      <xs:element ref="tns:abstractDescription"/>
      <xs:element name="additionalEntryTypeList"
         type="tns:additionalEntryTypeList"/>
      <xs:element name="prereqList" type="tns:prereqList"/>
      <xs:element name="referenceList" type="tns:referenceList"/>
      <xs:element name="comments" type="xs:string"/>
      <xs:element name="additionalInfo" type="tns:additionalInfo" minOccurs="0"
        maxOccurs="1"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:ID" use="required"/>
    <xs:attribute name="entryType" type="tns:entryTypes" use="required"/>
    <xs:attribute name="type" type="tns:testAssertionTypes" use="required"/>
    <xs:attribute name="enabled" type="xs:boolean" use="optional" default="true"/>
  </xs:complexType>

  <xs:complexType name="additionalEntryTypeList">
    <xs:sequence>
      <xs:element name="messageInput" type="tns:messageInput" minOccurs="0"
maxOccurs="1"/>
      <xs:element name="wsdlInput" type="tns:wsdlInput" minOccurs="0"
maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="prereqList">
    <xs:sequence>
      <xs:element name="testAssertionID" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="referenceList">
    <xs:sequence>
      <xs:element name="reference" type="tns:reference" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="reference">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="profileID" type="xs:IDREF" use="required"/>
        <xs:attribute name="role" type="tns:referenceRole" use="optional"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:simpleType name="artifactTypes">
    <xs:restriction base="xs:string">
      <xs:enumeration value="message"/>
      <xs:enumeration value="description"/>
      <xs:enumeration value="discovery"/>
      <xs:enumeration value="envelope"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="testAssertionTypes">
    <xs:restriction base="xs:string">
      <xs:enumeration value="required"/>
      <xs:enumeration value="recommended"/>
      <xs:enumeration value="informational"/>
      <xs:enumeration value="notTestable"/>
      <xs:enumeration value="driverTestable"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="messageEntryTypes">
    <xs:restriction base="xs:string">
      <xs:enumeration value="requestMessage"/>
      <xs:enumeration value="responseMessage"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="wsdlEntryTypes">
    <xs:restriction base="xs:string">
      <xs:enumeration value="definitions"/>
      <xs:enumeration value="import"/>
      <xs:enumeration value="types"/>
      <xs:enumeration value="message"/>
      <xs:enumeration value="operation"/>
      <xs:enumeration value="portType"/>
      <xs:enumeration value="binding"/>
      <xs:enumeration value="port"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="uddiEntryTypes">
    <xs:restriction base="xs:string">
      <xs:enumeration value="bindingTemplate"/>
      <xs:enumeration value="tModel"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="additionalEntryTypes">
    <xs:restriction base="xs:string">
      <xs:enumeration value="anyMessage"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="envelopeEntryTypes">
    <xs:restriction base="xs:string">
      <xs:enumeration value="requestEnvelope"/>
      <xs:enumeration value="responseEnvelope"/>
      <xs:enumeration value="anyEnvelope"/>
    </xs:restriction>
  </xs:simpleType>


  <xs:simpleType name="entryTypes">
    <xs:union memberTypes="tns:messageEntryTypes tns:wsdlEntryTypes
tns:uddiEntryTypes tns:additionalEntryTypes tns:envelopeEntryTypes"/>
  </xs:simpleType>

  <xs:simpleType name="baseInput">
    <xs:restriction base="xs:string">
      <xs:enumeration value="none"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="messageInput">
    <xs:union memberTypes="tns:messageEntryTypes tns:additionalEntryTypes
tns:envelopeEntryTypes tns:baseInput"/>
  </xs:simpleType>

  <xs:simpleType name="wsdlInput">
    <xs:union memberTypes="tns:wsdlEntryTypes tns:baseInput"/>
  </xs:simpleType>
  <xs:simpleType name="referenceRole">
    <xs:restriction base="xs:string">
      <xs:enumeration value="target"/>
      <xs:enumeration value="partial-target"/>
      <xs:enumeration value="collateral"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="additionalInfo">
    <xs:anyAttribute namespace="##any" processContents="lax"/>
  </xs:complexType>
</xs:schema>
```

A 2.3 Conformance Report Schema

This section contains the XML schema definition for the profile conformance report.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://www.ws-i.org/testing/2004/07/report/"
   xmlns:tns="http://www.ws-i.org/testing/2004/07/report/"
   xmlns:wsi-analyzerConfig="http://www.ws-i.org/testing/2004/07/analyzerConfig/"
   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:wsi-assertions="http://www.ws-i.org/testing/2004/07/assertions/"
   elementFormDefault="qualified">

  <xs:import namespace="http://www.ws-i.org/testing/2004/07/analyzerConfig/" />

  <xs:import namespace="http://www.ws-i.org/testing/2003/03/log/" />

  <xs:import namespace="http://www.ws-i.org/testing/2003/03/monitorConfig/" />

  <xs:import namespace="http://www.ws-i.org/testing/2004/07/assertions/" />

  <xs:element name="report" type="tns:report" nillable="true"/>
  <xs:complexType name="report">
    <xs:sequence>
      <xs:element name="analyzer" type="tns:analyzer"/>
      <xs:element name="artifact" type="tns:artifact" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="summary" type="tns:summary" minOccurs="0"
maxOccurs="1"/>
      <xs:element name="testCoverage" type="tns:testCoverage" minOccurs="0"
maxOccurs="1"/>
      <xs:element name="analyzerFailure" type="tns:analyzerFailure" minOccurs="0"
maxOccurs="1"/>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="optional"/>
    <xs:attribute name="timestamp" type="xs:dateTime" use="required"/>
  </xs:complexType>
  <xs:complexType name="analyzer">
    <xs:sequence>
      <xs:element name="implementer" type="tns:implementer"/>
      <xs:element name="environment" type="tns:environment"/>
      <xs:element ref="wsi-analyzerConfig:configuration"/>
    </xs:sequence>
    <xs:attribute name="version" type="xs:string" use="required"/>
    <xs:attribute name="releaseDate" type="xs:date" use="required"/>
  </xs:complexType>

  <xs:complexType name="implementer">
    <xs:attribute name="name" type="xs:string"/>
    <xs:attribute name="location" type="xs:anyURI"/>
  </xs:complexType>

  <xs:complexType name="environment">
    <xs:sequence>
      <xs:element name="runtime" type="tns:environmentElement"/>
      <xs:element name="operatingSystem" type="tns:environmentElement"/>
      <xs:element name="xmlParser" type="tns:environmentElement"/>
   </xs:sequence>
  </xs:complexType>

  <xs:complexType name="environmentElement">
    <xs:attribute name="name" type="xs:string" use="required"/>
    <xs:attribute name="version" type="xs:string" use="required"/>
  </xs:complexType>

  <xs:complexType name="artifact">
    <xs:sequence>
      <xs:element name="artifactReference" type="tns:artifactReference" minOccurs="0"
maxOccurs="1"/>
      <xs:element name="entry" type="tns:entry" minOccurs="1"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="type" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="message"/>
          <xs:enumeration value="description"/>
          <xs:enumeration value="discovery"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>

  <xs:complexType name="artifactReference">
    <xs:sequence>
      <xs:element name="comment" type="xs:string" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
    <xs:attribute name="timestamp" type="xs:dateTime" use="required"/>
  </xs:complexType>

  <xs:complexType name="entry">
    <xs:complexContent>
      <xs:extension base="tns:baseEntry">
        <xs:sequence>
          <xs:element name="messageEntry" type="wsi-log:messageEntry"
minOccurs="0" maxOccurs="1"/>
          <xs:element name="assertionResult" type="tns:assertionResult" minOccurs="1"
maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
   </xs:complexType>

  <xs:complexType name="baseEntry">
    <xs:attribute name="type" type="wsi-assertions:entryTypes" use="optional"/>
    <xs:attribute name="referenceID" type="xs:string" use="optional"/>
  </xs:complexType>

  <xs:complexType name="assertionResult">
    <xs:sequence>
      <xs:element name="additionalEntryList" type="tns:additionalEntryList"
minOccurs="0" maxOccurs="1"/>
      <xs:element name="prereqFailedList" type="tns:prereqFailedList" minOccurs="0"
maxOccurs="1"/>
      <xs:element name="assertionDescription" type="tns:message" minOccurs="0"
maxOccurs="1"/>
      <xs:element name="failureMessage" type="tns:message" minOccurs="0"
maxOccurs="1"/>
      <xs:element name="failureDetail" type="tns:failureDetail" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required"/>
    <xs:attribute name="result" type="tns:allResults" use="required"/>
  </xs:complexType>

  <xs:complexType name="prereqFailedList">
    <xs:sequence>
      <xs:element name="testAssertionID" type="xs:string" minOccurs="1"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="additionalEntryList">
    <xs:sequence>
      <xs:element name="additionalEntry" type="tns:additionalEntry" minOccurs="1"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="additionalEntry">
```

```
    <xs:complexContent>
      <xs:extension base="tns:baseEntry"/>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="summary">
    <xs:attribute name="result" type="tns:summaryResults" use="required"/>
  </xs:complexType>

  <xs:complexType name="testCoverage">
    <xs:sequence>
      <xs:element name="service" type="tns:service" minOccurs="1"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="service">
    <xs:sequence>
      <xs:element name="operation" type="tns:operation" minOccurs="1"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="name" type="xs:NCName" use="required"/>
    <xs:attribute name="location" type="xs:anyURI" use="required"/>
  </xs:complexType>

  <xs:complexType name="operation">
    <xs:attribute name="messages" type="xs:integer" use="required"/>
    <xs:attribute name="name" type="xs:NCName" use="required"/>
    <xs:attribute name="input" type="xs:NCName" use="required"/>
    <xs:attribute name="output" type="xs:NCName" use="required"/>
    <xs:attribute name="fault" type="xs:NCName" use="optional"/>
  </xs:complexType>

  <xs:complexType name="analyzerFailure">
    <xs:sequence>
      <xs:element name="failureDetail" type="tns:message"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="message">
    <xs:simpleContent>
      <xs:extension base="xs:string"/>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="failureDetail">
    <xs:complexContent>
      <xs:extension base="tns:message">
        <xs:attribute name="referenceType" type="tns:referenceType" use="optional"/>
        <xs:attribute name="referenceID" type="xs:string" use="optional"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:simpleType name="referenceType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="operation"/>
      <xs:enumeration value="message"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="summaryResults">
    <xs:restriction base="xs:string">
      <xs:enumeration value="passed"/>
      <xs:enumeration value="failed"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="otherResults">
    <xs:restriction base="xs:string">
      <xs:enumeration value="warning"/>
      <xs:enumeration value="notApplicable"/>
      <xs:enumeration value="missingInput"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="allResults">
    <xs:union memberTypes="tns:summaryResults tns:otherResults"/>
  </xs:simpleType>
</xs:schema>
```

A 2.4 Common Element Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://www.ws-i.org/testing/2003/03/common/"
   xmlns:wsi-common="http://www.ws-i.org/testing/2003/03/common/">

  <xs:complexType name="description">
    <xs:simpleContent>
      <xs:extension base="xs:string">
      <!--
        <xs:attribute ref="xml:lang"/>
      -->
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:element name="addStyleSheet" type="wsi-common:addStyleSheet" />
  <xs:complexType name="addStyleSheet">
    <xs:attribute name="href" type="xs:anyURI" use="required"/>
    <xs:attribute name="type" type="xs:string" use="optional" default="text/xsl"/>
    <xs:attribute name="title" type="xs:string" use="optional"/>
    <xs:attribute name="media" type="xs:string" use="optional"/>
    <xs:attribute name="charset" type="xs:string" use="optional"/>
    <xs:attribute name="alternate" type="xs:boolean" use="optional"/>
  </xs:complexType>

</xs:schema>
```

A 3 Existing Tools and Source Code

This section identifies existing tools and source code that can be used to build the analyzer tool.

	Tool or Package Name	Description	Language	Source Code Available	License
1	Eclipse	Eclipse contains a set of XML and Web service tools in an IDE format.	Java	Yes	Common Public License
	http://www.eclipse.org				
2	WSDL4J	Open source implementation of JWSDL standard. Can be used to parse WSDL documents.	Java	Yes	IBM Public License
	http://www.ibm.com/developerWorks/opensource/wsdl4j				
3	UDDI4J	Java client API for interacting with a UDDI registry.	Java	Yes	IBM Public License
	http://www.uddi4j.org				
4	Java XML Pack	Current version of JAXM, JAXP, JAXR, and JAX-RPC.	Java	No	Sun License
	http://java.sun.com/xml/downloads/javaxmlpack.html				
5	Xerces	Java-based XML parser with fully conforming XML Schema processor.	Java	Yes	Apache
	http://xml.apache.org/xerces2-j/index.html				