Testing Work Group

Project:

WS-I Monitor Tool Functional Specification

[MonitorSpecification.doc]

Doc Type:

Technical Design Specification

Editor:

Scott Seely – Microsoft

David Lauzon – IBM

Contributors:

Peter Brittenham – IBM

Jacques Durand – Fujitsu

Lucien Kleijkers – Microsoft

Keith Stobie – Microsoft

Last Edit Date:

June 13, 2005

Document Status:

Version 1.1 Final

Notice

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

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

License Information

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

Feedback

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

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

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

Table of Contents:

1. Introduction

The Web Services Interoperability Organization (WS-I) will develop a set of interoperability tests and supporting testing resources for verifying Web service implementations for compliance WS-I implementation guidelines and the related industry specifications. The

results will help companies ensure that their Web services are compliant with interoperability guidelines.

One part of the testing infrastructure is a message capture and logging tool. This tool captures messages and stores them for later analysis. The tool itself will have to capture messages traveling over different protocols and transports. The first version of this tool will focus on being able to accurately capture HTTP based SOAP messages. Also, while many interception techniques are available, this implementation uses a man in the middle approach to intercept and record messages.

Section 2 of this document, as well as Appendix 2 (Schemas) are normative. Any monitoring tool claiming to be WS-I conformant, 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 Monitor specifications for versions beyond 1.1, 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/2004/07/log/	Message log
wsi-monConfig	http://www.ws-i.org/testing/2004/07/monitorConfig/	Monitor configuration file
wsi-common	http://www.ws-i.org/testing/2003/03/common/	Common element definitions

		•	
		•	
		•	
		•	
		•	
		•	

		•	
		•	
		•	
		•	
		•	
		•	

2. Monitor Tool Design

This section described the main design features for the Monitor Tool. It is focused on Version 1.1 of WS-I tools.

The Monitor has two distinct sets of functionality:

1. It is responsible for sending messages on to some other endpoint that is capable of accepting the traffic while preserving the integrity of communication between the two endpoints.
2. It is responsible for recording the messages that flow through it to a log file.

One can think of these two pieces as an interceptor and a logger. For this version of the Monitor, the interceptor and logger functionality will exist in the same application. The working group recognizes that we may later desire to separate the interceptor and the logger into two, standalone entities. This design discusses how one would go about structuring an application today that should be able to be broken into separate pieces in future versions.

Three designs of Interceptor have been investigated. They are described in Appendix A1. For the initial version of the Monitor, only one has been selected: the "man-in-the-middle" (MITM) design, described in Appendix A1.1. In the MITM model, the interceptor captures messages by situating itself between the sender and receiver of the message. The interception technique will vary depending on the transport. For the versions 1.0 and 1.1, we are only addressing HTTP riding on top of TCP/IP. The interceptor is responsible for capturing the message and any related data. The interceptor then takes that data and MUST forward it to the ultimate destination. It also sends the data on to the message logger. To send the data to the message logger, the interceptor should format the data into an XML stream or an object capable of being easily transformed into XML. The data captured is dependent on the transport protocol in use. With HTTP, the interceptor must capture the HTTP headers as well as the type of message (HTTP request or response).

Figure 1 shows the generic architecture this design (based on MITM) recommends. The interceptor sits between the message sender and receiver. When a message goes through the interceptor, the message is recorded in the logger and then passed on, unchanged, to the receiver. This architecture can be implemented in a number of ways. Each interceptor has advantages and disadvantages over other designs.

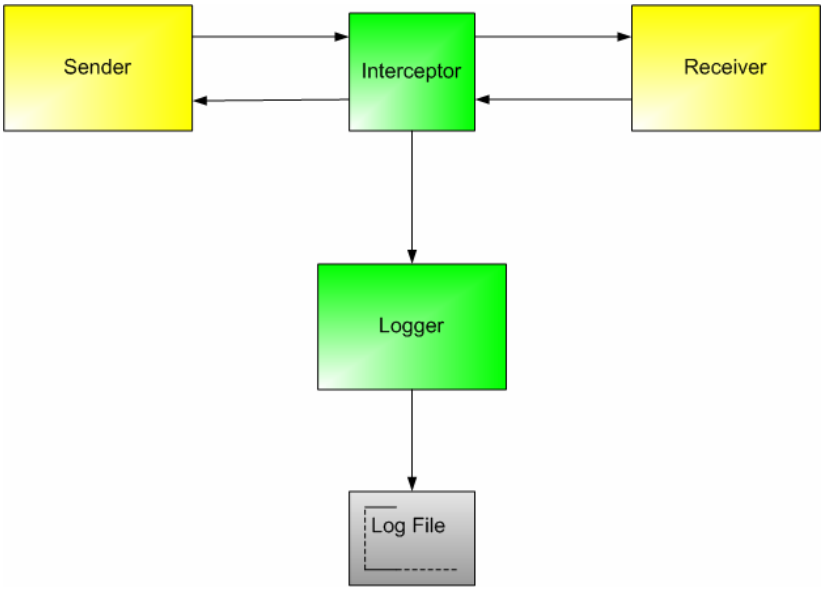

Figure 1. General architecture

The interceptor and logger have been logically (but not physically) separated for a number of reasons.

1. The interceptor will need to be transport protocol specific, resulting in many transport-specific Monitor configurations reusing the same Logger.
2. In future designs, the Logger can/should be a central repository of messages coming in from multiple endpoints during a test
3. Memory constrained devices such as PDAs and cell phones will not have the capacity to store message logs. Interceptors running on these devices ought to have the capacity to resend messages to a remote logger.

The separation approach should give the testing tools a reasonably long lifetime in terms of usability. The end goal is to have a set of tools that evolves while the profiles come out. The above architecture should allow such a design.

2.1 Monitor Requirements

This section contains a summary of the requirements for the monitor tool based on MITM design. The MITM design is implicitly assumed in the rest of this specification. All implementations of the monitor tool MUST adhere to these requirements. The MITM approach is defined in A1.1.

2.1.1 Listen on multiple ports

The monitor MUST create TCP/IP-based listeners for logging purposes. The tool MUST be able to support at least 5 concurrent listeners. The tool MAY support more than 5 concurrent

listeners. If the configuration file indicates a number of listeners greater than what the tool supports, the tool MUST output an error indicating how many ports are allowed.

2.1.2 Accept multiple connections per port

The monitor will be able to accept multiple connections per listening port. The tool MUST be able to accept 10 concurrent connections per port. The tool MAY support more than 10 concurrent connections per port.

2.1.3 Port to endpoint mapping

The monitor MUST enforce a one to one mapping between ports that it listens on and endpoints that it transmits data received on those ports to. For example, the Monitor can be setup in such a way that all data received on port 8080 will be forwarded to www.tempuri.org, port 80 and no where else. This mapping will be described through a configuration file. When the monitor receives a message, it MUST update the HTTP Host header to reference the host the monitor will forward the message to. This is necessary to make the HTTP Headers collection valid for the destination endpoint.

For example, if the monitor is on localhost and is forwarding the message to ws-i.org, here is how the HTTP Host header needs to change:

As received at monitor:

```
Host: localhost:8080
```

As updated and forwarded:

```
Host: ws-i.org:80
```

2.1.4 Data Exchange

When the monitor receives data on a given port, it MUST forward that data to the mapped endpoint. The monitor MUST exchange data between the client and the destination endpoint until the client or server terminates their TCP/IP connection to the monitor or until the duration of the logging / expiration time (as specified in configuration) expires. The monitor MUST transmit the data it receives without modification. Data received from a client will be transmitted to the mapped endpoint. Data received from the mapped endpoint during the HTTP conversation MUST be transmitted to the client that initiated the connection.

The data exchange MUST run for no longer than the number of seconds, logDuration + cleanupTimeoutSeconds, indicated in the monitor configuration file.

2.1.5 Logging

The monitor forwards all data it receives but logs a subset of all messages. We only log a subset in order to reduce the complexity of the analyzer correlation algorithms. The monitor will ignore certain control verbs because the verbs have nothing to do with SOAP over HTTP.

The logging filter is based on the control verb for each request. When any of the following control verbs appear in a request message, the request and any resulting response MUST NOT be recorded:

- OPTIONS: This verb is not logged because an HTTP server currently has no use of the body. Also, since the server (and not an HTTP application) responds to this verb, we can safely assume that it was never intended for a SOAP endpoint.

- HEAD: The server cannot return a message body for this type of request.
- DELETE: Removes the resource identified by the resource URI.
- TRACE: This message only reflects the originally sent message. This is not useful for the analyzer as the monitor already tracks the originally sent message.
- CONNECT: This is intended for use for a tunneling proxy.

All other control verbs MUST be recorded. This includes the known control verbs POST, GET and PUT as well as any not in the above list.

Each log entry MUST contain the following pieces of information:

1. The IP address and port of the client socket.
2. The IP address and port that the information was sent to. This must correspond to the endpoint in the configuration file configured for the monitor listener port.
3. All HTTP headers as sent to the forwarded endpoint.
4. The time that the message was received by the monitor.
5. Indication of whether the message was sent by the initiator (HTTP request) or mapped endpoint (HTTP response).
6. Unique identifier per HTTP connection request, shared amongst log entries created during the conversation, indicating which conversation the log entry belongs to.
7. Unique identifier per log entry. This can be a monotonically increasing value or a GUID. The identifier must be unique within the log file.

It is possible for the message log to contain a log entry with the following HTTP header/SOAP message combinations:

1. HTTP headers, no SOAP message
2. HTTP headers, SOAP message

Examples:

1. In a one-way SOAP message, the server returns a 200-OK with no content in the HTTP body.
2. Client sends HTTP Headers and SOAP message all at once.

Within a given conversation, the log MUST store all entries in time-received order. The time received is determined by the time the first byte is received from a given direction in the conversation. Conversations may be interleaved. That is, if two separate clients are connected to the monitor, the requests and responses between those clients and any receivers may be interleaved.

2.1.6 Use of Configuration File

The monitor tool MUST have a command line interface. When the analyzer tool is used on a command line, there MUST be at least one input option. This input option is used to reference the monitor configuration file, defined in 2.3. This is the format for the required command line interface:

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

Option	Definition

-config	This option contains a reference to the monitor configuration file. This can be a URL.

2.1.7 Output Equivalence

Any two implementations of the monitor that follow this specification must produce equivalent content for the logged HTTP Headers and SOAP messages for an equivalent combination of SOAP endpoints. Two log files are equivalent if they record the same wsi-log:HttpHeader and wsi-log:messageContent information for a given SOAP message. A set of endpoints are equivalent if they implement the same WSDL. Two log files are equivalent if they contain the same messages

2.1.8 HTTP Logging Behavior

When an HTTP message is captured, the message contents MUST be split in the following way:

- The portion of the message containing the HTTP headers MUST be placed into the httpHeaders log file element, described in 2.4.
- The remainder of the message MUST be placed into the messageContent log file element, described in 2.4. The message must be stored as an XML encoded string. Any unprintable characters and any special XML characters must be XML encoded within the document.

The behavior for the monitor when receiving any data that is not an HTTP Header is to encode that data so that it can be represented as a string. The data MUST be stored as binary encoded (xsd:base64Binary or xsd:hexBinary).

For HTTP messages, the Monitor MUST read the HTTP Content-Length header. The monitor MUST NOT record the message until Content-Length bytes have been received OR the connection has been closed. The connection can be closed by the message sender. The connection can be closed by the Monitor when the logDuration + cleanupTimeoutSeconds number of seconds defined in the configuration file has elapsed.

When logging message, the Request MUST be logged before the corresponding Response.

2.2 Monitor Functional Description

The following list is a functional overview of the monitor tool:

- Intercept messages between Web service endpoints using a man in the middle configuration.
- Record intercepted messages to a log file.
- Format intercepted messages with a specific XML schema, explained in 2.4 and detailed in A3.1.

The monitor SHOULD be designed so that the message capture and message logging components could be separated at a later date. The message capture mechanism SHOULD handle mapping the received data into a format capable of being logged. Options include formatting the received message data into XML as either an XML Element or XML Serializable piece of binary data.

2.2.1 Message Capture Process

The monitor application uses a man in the middle interceptor. Typical message flow is depicted in Figure 2.

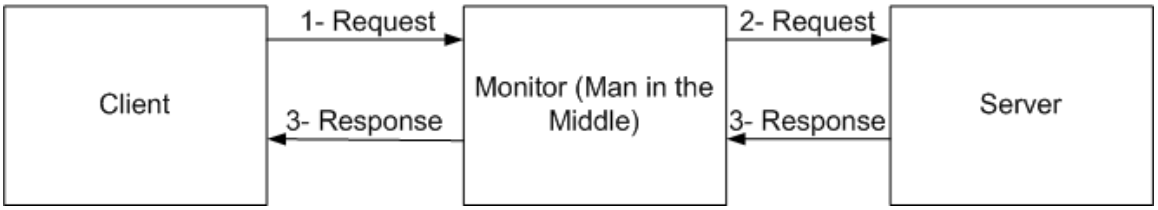

Figure 2. Message Capture Process

When a monitor starts up, it follows these steps:

1. Open the configuration file specified on the command line.
2. Open up one SOAP listener for each entry in the configuration file.
3. Listen for connections on each SOAP listener.
4. Stay running until the monitor is shut down.

Each SOAP listener MUST exhibit the following behavior:

1. Waits for a client to establish a connection with the open port.
2. Upon the client establishing a connection to the monitor, the monitor stores received bytes into a buffer.
3. As the client sends bytes, the monitor sends these bytes to the server corresponding to the port as mapped in the configuration file.
4. After the client is done sending any data, the following information is captured and logged:
    a. Client IP address and port
    b. Server IP address and port
    c. Message is marked as request
    d. Time the first bytes of the message were received is recorded.
    e. Protocol Headers are separated from SOAP message

    The data, already deciphered for the protocol being listened to, is sent to the message log.
5. Waits for response from server.
6. As the server sends bytes, the monitor streams these bytes to the client that initiated the connection.
7. After the server is done sending any data the following information is captured and logged:
    a. Client IP address and port
    b. Server IP address and port
    c. Message is marked as response

d. Time the first bytes of the message were received is recorded.

e. Protocol headers are separated from SOAP message.

The data, already deciphered for the protocol being listened to, is sent to the message log.

Under typical conditions, the monitor will stop accepting new connections after a specified number of seconds, as specified in the configuration file. Once this period has passed, the monitor MUST stop running after all open connections finish their conversations. To prevent an infinitely long wait period, the configuration file contains another value, cleanupTimeoutSeconds, that specifies how long the monitor MUST wait before forcibly terminating any remaining connections.

2.2.2 Monitor Logging Process

All captured messages MUST be added to the message log. The location of the log is read from the configuration file. At startup, the Monitor will check to see if the log exists. If the log does not exist, the Monitor must create a new file with the name of the file. If the file does exist, the Monitor will delete the file and create a new one if the value of wsi-monConfig:logFile/@replace is true. If the file does exist and wsi-monConfig:logFile/@replace is false, the Monitor will output the error message:

```
Error: File [file name from config] already exists. replace is set to false.
Monitor cannot start.
```

If it is not possible to create a file at the specified location, the Monitor MUST output an error message stating “The Monitor could not start because of issues with the log file: [file name from config].” An implementation MAY output the reason why (out of space, security, file is read-only, etc.). When the log is initially created, the Monitor MUST write out the root log element and the Monitor element. No other information SHOULD go into the root element.

As messages are received, they MUST be added to the end of the log.

2.3 Monitor Configuration File

The monitor MUST be configurable by a configuration file. The configuration file location will be passed to the monitor at startup via the command line. The command line syntax is as follows:

```
-config [config file name]
```

For a monitor whose executable is named MONITOR.EXE and a configuration file named monitorConfig.xml, the command line would read

```
MONITOR –config monitorConfig.xml
```

The configuration file schema associated with the namespace
http://www.ws-i.org/testing/2004/07/monitorConfig/
is defined in section A3.2. The wsi-common file schema associated with name space
http://www.ws-i.org/testing/2003/03/common/
can be found in the “A 2.4 Common Element Schema” appendix of “WS-I Analyzer Tool Functional Specification” [Analyzer Tool Funtional Specification.doc]

A typical configuration file will look like this:

```
<?xml version="1.0" encoding="utf-8" ?>
   <!--
        Copyright © 2003-2004 by The Web Services-Interoperability Organization (WS-I) and
        Certain of its Members. All Rights Reserved.
        ...
    -->
<wsi-monConfig:configuration
  xmlns:wsi-monConfig="http://www.ws-i.org/testing/2004/07/monitorConfig/" >
   <wsi-monConfig:comment>This is a sample monitor config file
        </wsi-monConfig:comment>
   <wsi-monConfig:logFile replace="true" location="traceLog.xml">
        <wsi-monConfig:addStyleSheet href="..\..\common\xsl\log.xsl"
                                    type="text/xsl" alternate="false" />
        <wsi-monConfig:multipartRelatedMessage logRootPartOnly'true" />
   </wsi-monConfig:logFile>
   <wsi-monConfig:logDuration>900</wsi-monConfig:logDuration>
   <wsi-monConfig:cleanupTimeoutSeconds>30
        </wsi-monConfig:cleanupTimeoutSeconds>
   <wsi-monConfig:manInTheMiddle>
     <wsi-monConfig:redirect>
          <wsi-monConfig:comment>This redirects to local machine
            </wsi-monConfig:comment>
          <wsi-monConfig:listenPort>9090</wsi-monConfig:listenPort>
          <wsi-monConfig:schemeAndHostPort>http://localhost:80
            </wsi-monConfig:schemeAndHostPort>
          <wsi-monConfig:maxConnections>1000
            </wsi-monConfig:maxConnections>
          <wsi-monConfig:readTimeoutSeconds>30
            </wsi-monConfig:readTimeoutSeconds>
     </wsi-monConfig:redirect>
     <wsi-monConfig:redirect>
          <wsi-monConfig:comment>This redirects to a different machine
            </wsi-monConfig:comment>
          <wsi-monConfig:listenPort>8080</wsi-monConfig:listenPort>
          <wsi-monConfig:schemeAndHostPort>http://www.tempuri.org
            </wsi-monConfig:schemeAndHostPort>
          <wsi-monConfig:maxConnections>1000
            </wsi-monConfig:maxConnections>
          <wsi-monConfig:readTimeoutSeconds>30
            </wsi-monConfig:readTimeoutSeconds>
```

```
    </wsi-monConfig:redirect>
  </wsi-monConfig:manInTheMiddle>
</wsi-monConfig:configuration>
```

The above configuration file tells the monitor to do the following things:

1. Log all messages to a file named traceLog.xml with an xml StyleSheet.
2. For a multipart/related message, log only the root part of that message.
3. Run a man in the middle listener.
4. Open up listener connections on two ports: 9090 and 8080.
5. Forward any traffic received on port 9090 to localhost, port 80.
6. Forward any traffic received on port 8080 to www.tempuri.org, port 80.
7. Allow up to 1000 connections on ports 9090 and 8080.
8. Set the cleanupTimeoutSeconds value for a connection for ports 9090 and 8080 to 30 seconds.

The elements are defined as follows:

Element	**Description**
Configuration	Root element for the configuration file.
Comment	This element may appear at a number of levels. The element contains data that is meant to be read by the people trying to understand the configuration document.
logFile	Identifies the file that will contain the serialized XML and optionally an XML style sheet..
logFile/@replace	This attribute of logFile is a boolean value that tells how to behave when the logFile already exists. If replace is true, the logFile is deleted at startup and a new file is created in its place. If replace is false, the monitor will fail to start and output an error message.
logFile/@location	Identifies the name of the file that will contain the serialized XML.
addStyleSheet	Indicates if a XML [3] style sheet reference should be added to the output trace log. **Note:** If this element is not specified, then the following comment line will be inserted in the report file after the XML declaration statement: <!-- ?xml-stylesheet type="text/xsl" href="..\common\xsl\traceLog.xsl"? -->
addStyleSheet/@href	The location of the style sheet.

Element	Description
addStyleSheet/@type	The content type for the style sheet. The default for this attribute is “text/xsl”.
addStyleSheet/@title	Advisory information about the style sheet.
addStyleSheet/@media	Intended destination medium.
addStyleSheet/@charset	Character encoding for the style sheet.
addStyleSheet/@alternate	Indicates use of alternate style sheet.
multipartRelateMessage	Element containing options for logging multipart/related messages.
multipartRelateMessage@logRootPartOnly	Indicates whether all parts or just the root part of a multipart/related message will get logged. The default is to log only the root part.
logDuration	Identifies the number of seconds that the monitor MUST accept new client connections. After this many seconds, the monitor MUST NOT accept any new client connections. The stop time should be set immediately after all SOAP listeners have been started.
cleanupTimeoutSeconds	Once logDuration seconds passes, the monitor MUST begin to shutdown and accept no new connections. To allow any existing conversations to finish, the monitor will leave any active ports alive for the number of seconds set in cleanupTimeoutSeconds. Once that period passes, the ports MUST be shutdown.
manInTheMiddle	Containing element for any manInTheMiddle monitors.
redirect	complexType that defines where the monitor will listen for traffic and how it will forward that traffic.
listenPort	Tells the manInTheMiddle which port to listen on.

schemeAndHostPort	When traffic is received on listenPort, it is sent to the URL specified here. The URL is specified as an HTTP URL as specified in RFC1738 [1], section 3.3. An HTTP URL has the form,
	
	`http://<host>:<port>/<path>?<searchpart>`
	where port is optional and defaults to 80 when not specified. The URL that can be specified in schemeAndHostPort takes a subset of the above form:
	
	`http://<host>:<port>`
	where the port is still optional and defaults to

Element	Description
	80 when not specified.
maxConnections	Specifies the maximum number of connections that the port will queue up before it begins refusing connections.
readTimeoutSeconds	Specifies how long a listenPort should wait for a read operation before assuming that the connection timed out and releasing the connection. This timeout occurs when no data has been received by the client or server during this duration. If either end does send data, then neither connection is assumed to have timed out.

In versions 1.0 and 1.1, the monitor and analyzer are only being designed to handle HTTP traffic. For future versions of the Monitor, consideration should be given to adding information about the transport layer as well.

2.4 Monitor Log File

The monitor logs all information to a file. The log file schema associated with the namespace
http://www.ws-i.org/testing/2004/07/log/
is defined in section A3.1. The file MUST use XML and MUST be valid when compared against the schema in A3.1. The file contains information explaining what monitor produced the log. This entry MUST appear as the first child element in the log. Following the information about the monitor will be the log messages.

A log between two endpoints, both on localhost, that logs a simple conversation looks like this (Note for BOMs: 4294851584 = FF FE 3C 00 and 15711167 = EF BB BF)

```
<?xml version="1.0" encoding="UTF-8" ?>
 <log timestamp="2004-01-30T12:22:51.724"
     xmlns="http://www.ws-i.org/testing/2004/07/log/"
     xmlns:wsi-log="http://www.ws-i.org/testing/2004/07/log/"
     xmlns:wsi-monConfig="http://www.ws-i.org/testing/2004/07/monitorConfig/"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <monitor version="1.0" releaseDate="2003-03-20"><implementer name="IBM"
       location="" />
      <environment>
         <runtime name="Java(TM) 2 Runtime Environment, Standard Edition"
            version="1.4.1" />
         <operatingSystem name="Windows XP" version="5.1" />
         <xmlParser name="Apache Xerces" version="XML4J 4.2.2" />
       </environment>
      <wsi-monConfig:configuration>
         <wsi-monConfig:comment>Comment</wsi-monConfig:comment>
        <wsi-monConfig:logFile replace="true" location="URL">
           <wsi-monConfig:addStyleSheet href="null" type="null" />
```

```
        </wsi-monConfig:logFile>
        <wsi-monConfig:logDuration>600</wsi-monConfig:logDuration>
        <wsi-monConfig:cleanupTimeoutSeconds>3</wsi-
            monConfig:cleanupTimeoutSeconds>
        <wsi-monConfig:manInTheMiddle>
          <wsi-monConfig:redirect>
            <wsi-monConfig:comment>This redirects to local machine
              </wsi-monConfig:comment>
            <wsi-monConfig:listenPort>9090</wsi-monConfig:listenPort>
            <wsi-monConfig:schemeAndHostPort>http://localhost:80
              </wsi-monConfig:schemeAndHostPort>
            <wsi-monConfig:maxConnections>1000
              </wsi-monConfig:maxConnections>
            <wsi-monConfig:readTimeoutSeconds>30
              </wsi-monConfig:readTimeoutSeconds>
          </wsi-monConfig:redirect>
          <wsi-monConfig:redirect>
            <wsi-monConfig:comment>This redirects to a different machine
              </wsi-monConfig:comment>
            <wsi-monConfig:listenPort>8080
              </wsi-monConfig:listenPort>
            <wsi-monConfig:schemeAndHostPort>http://www.tempuri.org
              </wsi-monConfig:schemeAndHostPort>
            <wsi-monConfig:maxConnections>1000
              </wsi-monConfig:maxConnections>
            <wsi-monConfig:readTimeoutSeconds>30
              </wsi-monConfig:readTimeoutSeconds>
          </wsi-monConfig:redirect>
        </wsi-monConfig:manInTheMiddle>
      </wsi-monConfig:configuration>
    </monitor>

    <messageEntry xsi:type="wsi-log:httpMessageEntry" ID="1" conversationID="1"
        type="request" timestamp="2004-01-30T12:22:35.191Z">
      <messageContent>
        &lt;?xml version=&apos;1.0&apos; encoding=&apos;UTF-8&apos;?&gt;&#xd;
        &lt;SOAP-ENV:Envelope xmlns:SOAP-ENV=&quot;http://schemas.xmlsoap.org/
        soap/envelope/&quot;&gt;&#xd;
        &lt;SOAP-ENV:Body&gt;&#xd;
        &lt;GetProductDetails xmlns=&quot;http://www.ws-i.org/Sample
        Applications/SupplyChainManagement/2003-07/Catalog.xsd&quot;&gt;&#xd;
        &lt;ProductNumber&gt;605001&lt;/ProductNumber&gt;&#xd;
        &lt;/GetProductDetails&gt;&#xd; &lt;/SOAP-ENV:Body&gt;&#xd;
```

```
&lt;/SOAP-ENV:Envelope&gt;&#xd;
</messageContent>
<senderHostAndPort>localhost:9081</senderHostAndPort>
<receiverHostAndPort>localhost:9080</receiverHostAndPort>
<httpHeaders>POST Catalog/services/catalog HTTP/1.1
  Host: localhost:9080
  Content-Type: text/xml; charset=utf-8
  Content-Length: 1386
  SOAPAction: ""
</httpHeaders>
</messageEntry>

<messageEntry xsi:type="wsi-log:httpMessageEntry" ID="2" conversationID="1"
    type="response" timestamp="2004-01-30T12:22:35.261Z">
  <messageContentWithAttachments>
    <mimePart>
      <boundaryString>&#xd;&#xa;--MIME_boundary</boundaryString>
      <mimeHeaders>
        Content-Type: text/xml; charset=UTF-8
        Content-Transfer-Encoding: 8bit
        Content-ID: <soapbody.scm11=4d7a5fa2-14af-451c-961b-
        5c3abf786796@ws-i.org>
      </mimeHeaders>
      <mimeContent>
        &lt;SOAP-ENV:Envelope
        xmlns:SOAP-ENV=&quot;http://schemas.xmlsoap.org/soap/envelope/
        &quot;&gt;&#xd;
        &lt;SOAP-ENV:Body&gt;&#xd;
        &lt;ProductDetails xmlns=&quot;http://www.ws-i.org/
        SampleApplications/ SupplyChainManagement/2003-07/Catalog.xsd
        &quot;&gt;&#xd;
        &lt;Weight&gt;24.6&lt;/Weight&gt;&#xd;
        &lt;WeightUnit&gt;pounds&lt;/WeightUnit&gt;&#xd;
        &lt;Dimensions&gt;&#xd;
        &lt;Height&gt;26&lt;/Height&gt;&#xd;
        &lt;Width&gt;24&lt;/Width&gt;&#xd;
        &lt;Depth&gt;21&lt;/Depth&gt;&#xd;
        &lt;/Dimensions&gt;&#xd;
        &lt;DimensionUnit&gt;inches&lt;/DimensionUnit&gt;&#xd;
        &lt;/ProductDetails&gt;&#xd;
        &lt;/SOAP-ENV:Body&gt;&#xd;
        &lt;/SOAP-ENV:Envelope&gt;
      </mimeContent>
```

```
        </mimePart>
        <mimePart>
            <boundaryString>&#xd;&#xa;--MIME_boundary</boundaryString>
            <mimeHeaders>
                Content-Type: image/jpeg
                Content-Transfer-Encoding: binary
                Content-ID: <Pic=605001_big.jpeg@ws-i.org>
            </mimeHeaders>
            <mimeContent />
        </mimePart>
        <mimePart>
            <boundaryString>&#xd;&#xa;--MIME_boundary</boundaryString>
            <boundaryString>&#xd;&#xa;--MIME_boundary</boundaryString>
            <mimeHeaders>
                Content-Type: text/xml
                Content-Transfer-Encoding: binary
                Content-ID: <Specs=605001_specs.xml@ws-i.org>
            </mimeHeaders>
            <mimeContent />
        </mimePart>
    </messageContentWithAttachments>
    <senderHostAndPort>localhost:9080</senderHostAndPort>
    <receiverHostAndPort>localhost:9081</receiverHostAndPort>
    <httpHeaders>
        HTTP/1.1 200 OK
        Server: WebSphere Application Server/5.1
        MIME-Version: 1.0
        Content-Type: multipart/related; type="text/xml";
        start="<soapbody. scm11= 4d7a5fa2-14af-451c-961b-
        5c3abf786796@ws-i.org>"; boundary="MIME_boundary"
        Content-Language: en-US
        Connection: close
    </httpHeaders>
  </messageEntry>
</log>
```

What follows here is a description of the elements:

Element	Description	Attributes
log	Root element for the message log.	• timestamp: Identifies the time that the log was created. This value

Element	Description	Attributes
		MUST be less than or equal to the timestamp of the first messageEntry in the log.
monitor	Contains information that should allow someone reading a log to identify exactly which tool produced the log. This element also contains the XML in the configuration file used by the monitor.	• version: The version number for the implementation of the tool. • releaseDate: The date the tool was released.
implementer	Identifies the organization that created the monitor tool.	• name: Name of the organization that created the monitor tool. • location: URL pointing to the Web site where the monitor tool can be obtained.
environment	Describes the environment used to run the application and the language used to construct the application.	
runtime	Identifies the primary programming language used to create the monitor.	• name: Name of the runtime environment. Ex: Java, .NET, etc. • version: Identifies the version of the runtime in use. This could be a dotted version string (1.0.3705.288.
operatingSystem	Identifies the name and version of the operating system that the monitor is running on.	• name: Name of the operating system. • version: Identifies which version of the operating system is in use.
xmlParser	For some monitors, it is known that the XML Parser version will be important. This element is optional and should only be included if the XML Parser makes a difference.	• name: Recognized name of the XML parser. • version: Identifies which version of the XML Parser was used.
configuration	This is a straight import of the	

Element	Description	Attributes
	configuration file used when the logFile was created. This is included to help understand how the log itself produced its results.	
messageEntry	Used to identify a message. This represents a message moving in one direction.	• timestamp: Time that the message was received by the monitor tool. • conversationID: This identifier is used to group messages received between the time that the client connects to the monitor port and when that connection is closed. The string MUST change in between connections. • ID: This attribute is used to uniquely identify the message within the log. • type: Identifies the entry as an HTTP request or an HTTP response. Valid values are request and response.
messageContent, messageContentWithAttachments	Contains the HTTP Body message in the HTTP POST or HTTP response. Note that the actual element to log is determined by the Content-Type HTTP header field-value. If the field-value has a media-type of "multipart/related", then the <messageContentWithAttachments> element is logged. Otherwise, the <messageContent> element is always logged.	• BOM: This attribute is used to provide the Byte Order Mark (if any) that was originally in the HTTP payload.
mimePart	A root or non-root part of a multipart/related message.	
boundaryString	This identifies the boundary string delimiter including the	

Element	Description	Attributes
	preceding ascii characters that separate the part boundary.	
mimeHeaders	A standardized set of headers that describe the structure and content of a MIME part.	
mimeContent	Contains the MIME body or content of a part. . Note for all non-root parts, the content is encoded to base64 for logging purposes. This ensures that we are always able to generate a well-formed XML log file.	
senderHostAndPort	This identifies the host and TCP port that is sending data. In a Request message, this must match the value of the host element in the configuration file. Instead of including the scheme identifier, the value will simply be <host>:<port>. In this element, the port must be listed. When a port is not specified in an HTTP-based URL, the port value defaults to 80.	
receiverHostAndPort	This identifies the host and TCP port that is receiving data. In a Response message, this must match the value of the host element in the configuration file. Instead of including the scheme identifier, the value will simply be <host>:<port>. In this element, the port must be listed. When a port is not specified in an HTTP-based URL, the port value defaults to 80.	
httpHeaders	The raw text of any HTTP headers sent by one direction of an HTTP request or response.	

It is important to note that the monitor only logs the HTTP conversation, and makes as few assumptions as possible about the content of an HTTP body itself. For instance, the monitor does not check whether a messageContent element contains a SOAP message. If a message contains an HTTP Content-Length header, it will not record the message until the number of bytes specified in that header is received. This feature is included primarily to deal with HTTP messages that issue an HTTP 100 continue request separate from sending the

message body. (It is legal to send the HTTP headers independently of the message body with a HTTP Expect Header of 100-continue.)

However there are two cases where the monitor will do further processing. The first is related to a byte order mark [4]. A byte order mark, if present, will be removed from the actual body (<messageContent> or <messageContentWithAttachments>) and will be added as an attribute value. The second is related to non-root parts of a multipart/related message. The monitor provides a **logRootPartOnly** option. This option gives the added flexibility of being able to filter out the contents of the non-root parts (or the attachments) of a multipart/related message when logging information to a file. To support this feature, the multipart/related message contents will be further parsed into its MIME parts (<mimePart>). Note that a filtered non-root part will be represented as an empty MIME body (<mimeContent>) in the log. If not filtered, the content of a non-root part will be encoded to base64 to ensure that any binary content can be logged.

3. Security

This version of the Monitor will not address security. In particular, the monitor will not allow for SSL. This does not preclude one from using SSL with the Basic Profile. Instead, SSL traffic cannot be analyzed for compliance with the Basic Profile. To add SSL analysis capabilities to the Monitor, the Monitor would have to be coded to handle SSL handshakes as well as to hold its own server certificate. SSL is specifically designed to thwart man in the middle attacks, which the current design of the monitor requires.

If SSL capabilities were added to the monitor, the tests would only make sure that each SSL stack was able to interop with the monitor. Such testing would not uncover issues between the client and server SSL implementations. For this tool, we must assume that behavior when a Web service uses SSL is identical to the behavior it exhibits when using an open, unencrypted channel.

A1. Interceptor

In general, the interceptor positions itself in the message stream and captures data. This section presents three varieties of interceptor. For each interceptor, the strengths and weaknesses of each are discussed. Each interceptor is responsible for capturing the SOAP message. In addition to this, the interceptor must capture any applicable details associated with the protocol in question.

A1.1 Man In The Middle

One interceptor design is termed the “Man in the middle” (MITM). Figure 2 illustrates how a MITM has traditionally operated in the SOAP world. The MITM is typically capable of intercepting and forwarding messages on to multiple SOAP endpoints. This is accomplished by listening on a specific TCP/IP port for traffic. All traffic that arrives on this port is forwarded to a single IP address and port. For example, a MITM might open up port 8080. Any traffic that comes in on port 8080 would be recorded in the Logger and sent on to a specific IP address such as IP address: tempuri.org, port 80. Examples of MITM implementations include tcpTrace by Simon Fell (http://www.pocketsoap.com/tcpTrace), tcpTunnelGUI from the Apache SOAP Toolkit (http://xml.apache.org/soap), and MSSOAPT from the Microsoft SOAP Toolkit (http://msdn.microsoft.com/downloads/default.asp?URL=/downloads/sample.asp?url=/msdn-files/027/001/948/msdncompositedoc.xml). The MITM approach works well in traditional RPC message capture. When applying other SOAP messaging architectures such as routed messages, the MITM approach can suffer. A routed message may not necessarily return using the same path it left on. This approach also has

issues with encrypted messages. The MITM can capture the encrypted message but it may not always be able to decrypt the message. A MITM is a good interceptor for unencrypted, RPC style SOAP traffic. The primary benefit of the MITM is that the MITM is fairly easy to develop and requires only basic knowledge of how to read and write data over a TCP/IP socket.

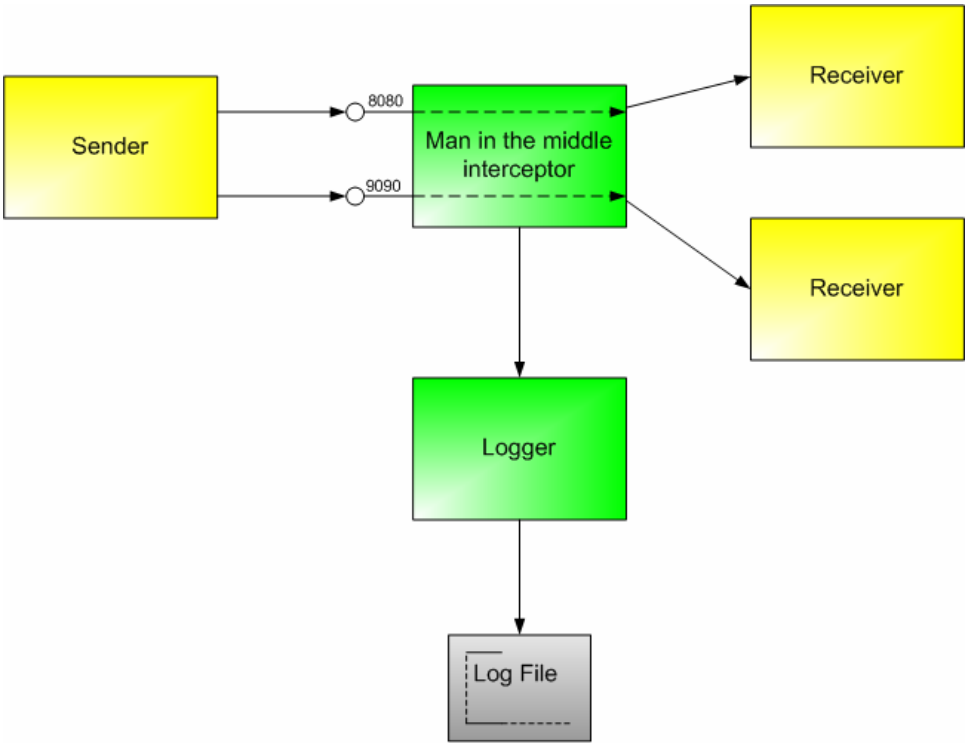

Figure 3. Man in the middle architecture

A1.2 Proxy Interceptor

Another option is a proxy interceptor. A proxy interceptor typically acts as a proxy for a particular protocol. HTTP traffic is handled using an HTTP proxy such as Microsoft's ISA Server. Simon Fell has created a SOAP, HTTP based proxy interceptor called proxyTrace (http://www.pocketsoap.com/tcpTrace/pt.aspx). Proxy interceptors have the benefit of only needing to listen on one port for all traffic. Configuration is much simpler than a MITM approach. The proxy also has many of the same drawbacks as well:

- Hard to handle encrypted streams
- Routed messages may be difficult to monitor on just one proxy
- Specific to a given protocol.

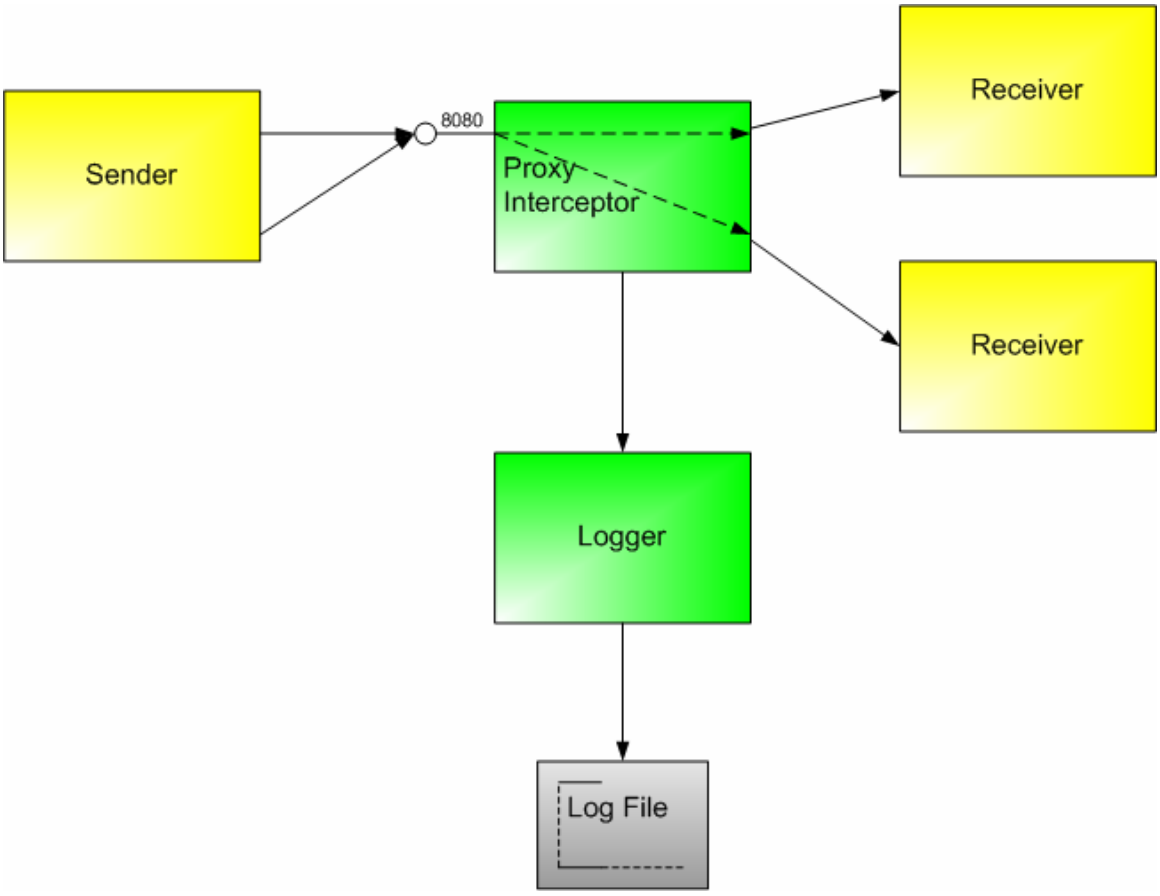

Figure 4. Proxy interceptor

A1.3 Same Machine Interceptor

The final option that will be discussed in this document is the idea of an interceptor that is able to capture messages as they flow through the system. We will call this a "same machine interceptor" or SMI. The SMI always sits on a message receiver. Depending on the protocol in use, an SMI may be able to capture messages created by the sender as well. For example, request and response should be able to be captured in an RPC-style SOAP interchange occurring over HTTP. The SMI works in one of two manners:

- Moves the Web service to a different listening location and then assumes the original listening location. For example, an HTTP-based Web service listening on port 80 would be moved to another available port such as 8080. The interceptor would then open up a socket listener on port 80 and forward all traffic to port 8080. The interceptor would record all traffic.
- Insinuates itself into the protocol stack. Example: on an ASP.NET based Web service, the interceptor would be implemented as an HTTP module. It could record any and all data coming into the Web service without changing any addressing on the server.

When creating a SMI, #2 represents the preferred approach. For example, HTTPS traffic can be read in its unencrypted form without any special knowledge about the server's certificate in the interceptor.

The SMI approach has a few potential drawbacks. There is no guarantee that a sender and receiver with similar instrumentation would have any orchestration in how messages are logged. As a result, messages may be logged more than once. Any analysis by the downstream analyzer would have to be able to handle duplicate messages.

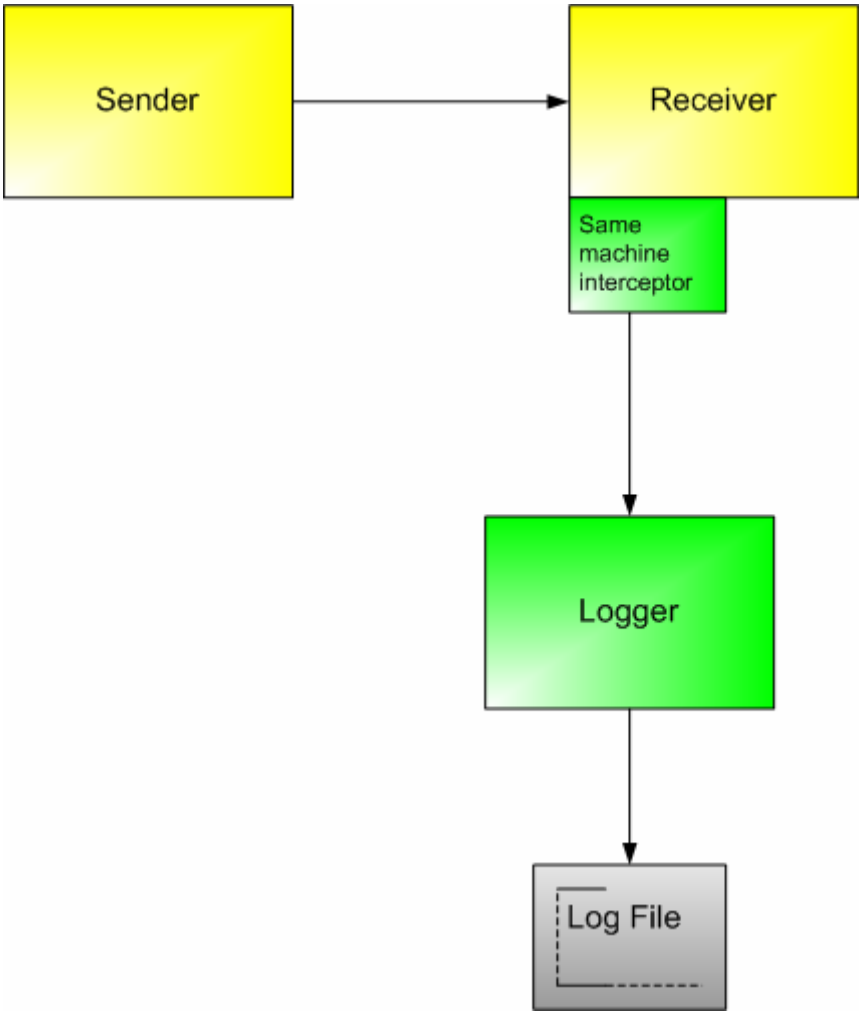

Figure 5. Same machine interceptor

A2. Schema

A2.1 Logging Schema

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

```
      "http://www.ws-i.org/testing/2004/07/monitorConfig/">
  <xs:import namespace=
      "http://www.ws-i.org/testing/2004/07/monitorConfig/" />
  <xs:element name="log" nillable="true" type="tns:log" />

  <xs:complexType name="log">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1"
        name="monitor" type="tns:monitor" />
      <xs:element minOccurs="0" maxOccurs="unbounded"
        name="messageEntry" type="tns:messageEntry" >
        <xs:unique name="ID_PK" >
           <xs:selector xpath="messageEntry" />
           <xs:field xpath="@ID" />
        </xs:unique>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="timestamp" type="xs:dateTime" />
  </xs:complexType>

  <xs:complexType name="monitor">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1"
        name="implementer" type="tns:implementation" />
      <xs:element minOccurs="1" maxOccurs="1"
        name="environment" type="tns:environment" />
      <xs:element minOccurs="1" maxOccurs="1"
        ref="wsi-monConfig:configuration" />
    </xs:sequence>
    <xs:attribute name="version" type="xs:string" />
    <xs:attribute name="releaseDate" type="xs:date" />
  </xs:complexType>

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

  <xs:complexType name="environment">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1"
        name="runtime" type="tns:nameVersionPair" />
```

```
    <xs:element minOccurs="1" maxOccurs="1"
      name="operatingSystem" type="tns:nameVersionPair" />
    <xs:element minOccurs="1" maxOccurs="1"
      name="xmlParser" type="tns:nameVersionPair" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="nameVersionPair">
  <xs:attribute name="name" type="xs:string" />
  <xs:attribute name="version" type="xs:string" />
</xs:complexType>

<xs:complexType name="messageEntry" abstract="true">
  <xs:choice>
    <xs:element minOccurs="0" maxOccurs="1"
      name="messageContent" type="tns:content" />
    <xs:element minOccurs="0" maxOccurs="1"
      name="messageContentWithAttachments"
      type="tns:messageContentWithAttachments" />
  </xs:choice>
  <xs:attribute name="timestamp" type="xs:dateTime" />
  <xs:attribute name="conversationID" type="xs:NMTOKEN" />
  <xs:attribute name="ID" type="xs:string" />
</xs:complexType>

<!--Byte Order Mark -->
<xs:attribute name="BOM" type="xs:unsignedInt" />

<xs:complexType name="content" mixed="true">
  <xs:simpleContent>
    <xs:extension base="xs:string">
     <xs:attribute ref="tns:BOM" use="optional" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="messageContentWithAttachments" mixed="false">
  <xs:sequence>
    <xs:element minOccurs="1" maxOccurs="unbounded"
        name="mimePart" type="tns:mimePart" />
  </xs:sequence>
  <xs:attribute ref="tns:BOM" use="optional" />
```

```
</xs:complexType>

<xs:complexType name="mimePart">
  <xs:sequence>
    <xs:element minOccurs="1" maxOccurs="1"
      name="boundaryString" type="xs:string" />
    <xs:element minOccurs="0" maxOccurs="1"
      name="mimeHeaders" type="xs:string" />
    <xs:element minOccurs="1" maxOccurs="1"
      name="mimeContent" type="xs:string" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="tcpLog" abstract="true">
  <xs:complexContent mixed="false">
    <xs:extension base="tns:messageEntry">
      <xs:sequence>
        <xs:element minOccurs="1" maxOccurs="1"
          name="senderHostAndPort" type="xs:string" />
        <xs:element minOccurs="1" maxOccurs="1"
          name="receiverHostAndPort" type="xs:string" />
      </xs:sequence>
      <xs:attribute name="type"
          type="tns:tcpMessageType" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:simpleType name="tcpMessageType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="request" />
    <xs:enumeration value="response" />
  </xs:restriction>
</xs:simpleType>

<xs:complexType name="httpMessageEntry">
  <xs:complexContent mixed="false">
    <xs:extension base="tns:tcpLog">
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="1"
          name="httpHeaders" type="xs:string" />
      </xs:sequence>
```

```
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:schema>
```

A2.2 Configuration File Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema
    xmlns:tns="http://www.ws-i.org/testing/2004/07/monitorConfig/"
    elementFormDefault="qualified"
    targetNamespace=
        "http://www.ws-i.org/testing/2004/07/monitorConfig/"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:wsi-common="http://www.ws-i.org/testing/2003/03/common/">
  <xs:import namespace="http://www.ws-i.org/testing/2003/03/common/" />
  <xs:element name="configuration" nillable="true"
    type="tns:configuration" />
  <xs:complexType name="logFile">
    <xs:sequence>
      <xs:element name="addStyleSheet" type="wsi-common:addStyleSheet"
                  minOccurs="0" maxOccurs="1"/>
      <xs:element name="multipartRelatedMessage"
                  type="tns:multipartRelatedMessage"
                  minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
    <xs:attribute name="location" type="xs:anyURI" use="required"/>
    <xs:attribute name="replace" type="xs:boolean" use="optional"
                  default="false"/>
  </xs:complexType>
  <xs:complexType name="configuration">
    <xs:sequence>
      <xs:element name="comment" minOccurs="0" maxOccurs="1"
         type="tns:comment" />
      <xs:element minOccurs="1" maxOccurs="1" name="logFile"
        type="tns:logFile" />
      <xs:element minOccurs="1" maxOccurs="1" name="logDuration"
        type="xs:int" />
      <xs:element minOccurs="1" maxOccurs="1"
        name="cleanupTimeoutSeconds"
        type="xs:int" />
      <xs:element minOccurs="0" maxOccurs="1" name="manInTheMiddle"
        type="tns:ArrayOfRedirect" />
```

```
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name=" multipartRelatedMessage">
    <xs:attribute name="logRootPartOnly" type="xs:boolean" use="optional"
      default=”true”/>
  </xs:complexType>
 <xs:complexType name="ArrayOfRedirect">
    <xs:sequence>
      <xs:element name="comment" minOccurs="0" maxOccurs="1"
         type="tns:comment" />
      <xs:element minOccurs="1" maxOccurs="unbounded" name="redirect"
        nillable="true" type="tns:redirect" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="redirect">
    <xs:sequence>
      <xs:element name="comment" minOccurs="0" maxOccurs="1"
         type="tns:comment" />
      <xs:element minOccurs="1" maxOccurs="1"
        name="listenPort" type="xs:int" />
      <xs:element minOccurs="1" maxOccurs="1"
        name="schemeAndHostPort" type="xs:anyURI" />
      <xs:element minOccurs="1" maxOccurs="1"
        name="maxConnections" type="xs:int" />
      <xs:element minOccurs="1" maxOccurs="1"
        name="readTimeoutSeconds" type="xs:int" />
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="comment">
    <xs:restriction base="xs:string" />
  </xs:simpleType>
</xs:schema>
```

A3. References

[1] Uniform Resource Locators (URL), RFC 1738. http://www.ietf.org/rfc/rfc1738.txt

[2] WS-I Analyzer Tool Functional Specification, http://ws-i.org/Documents.aspx

[Analyzer Tool Funtional Specification.doc]

[3] Associating Style Sheets with XML documents, Version 1.0
http://www.w3.org/1999/06/REC-xml-stylesheet-19990629

[4] Extensible Markup Language (XML) 1.0 (Second Edition) [Appendix F]
W3C Recommendation 6 October 2000
http://www.w3.org/TR/REC-xml