



# **FORUM SYSTEMS HANDS-ON TRAINING**

## **LAB 1. INTRODUCTION TO SOAP WEB SERVICES**



# FORUM SYSTEMS

A Crosscheck Networks Company

## Legal Marks

No portion of this document may be reproduced or copied in any form, or by any means – graphic, electronic, or mechanical, including photocopying, taping, recording, or information retrieval system – without expressed permission from Forum Systems, Inc.

FORUMOST™ Firmware, Forum Systems XMLSec™ WebAdmin, Forum Systems XML Security Appliance™, Forum Sentry™, Forum Presidio™, Forum XWall™, Forum Sentry™ Web Services Gateway, Forum Presidio™ OpenPGP Gateway, Forum FIA Gateway™, Forum XWall Type-PCI™, Forum XWall® Web Services Firewall and Forum XRay™ are trademarks and registered trademarks of Forum Systems, Inc.

All other products are trademarks or registered trademarks of their respective companies.

Copyright © 2002-2014 Forum Systems, Inc. – All Rights Reserved.

Published: September 2014

Forum Systems Hands-on Training – Lab 1. Introduction to SOAP Web Services  
D-ASF-SE-010029

## Contents

Introduction.....	4
<i>Skill Level</i> .....	4
<i>Prerequisites</i> .....	4
<i>Lab Overview</i> .....	4
A Simple SOAP Web Service .....	6
<i>Configuring .NET Web Matrix</i> .....	6
SOAPSonar.....	7
<i>Testing the SOAP Service</i> .....	7
WSDL and SOAP Basics .....	7
<i>WSDL File</i> .....	8
<i>SOAP Message</i> .....	8
<i>Why Use SOAP?</i> .....	9
Additional Testing and More Reading.....	10
<i>BACK IT UP!</i> .....	10
<i>Additional Tests and Discussion Topics</i> .....	10
<i>Additional Information</i> .....	10
About Forum Systems.....	13

## Introduction

Lab 1.Introduction to SOAP Web services – building a sample SOAP web service

### Skill Level

This lab is beginner skill level. Little to no prior experience with Forum Sentry or SOAPSonar is required.

### Prerequisites

This lab requires the Forum Sentry Training Image, with a licensed copy of SOAPSonar Enterprise Edition.

Refer to the “Forum Sentry Online Training.pdf” document in the “Introduction To Forum Sentry Training” course for information on the Forum Sentry Training Image and licensing SOAPSonar Enterprise Edition.

### Lab Overview

One of the primary functions of the Forum Sentry API Gateway is to secure Web services. This lab will focus on building and testing a sample SOAP Web service that will be used in later labs in this series.

For a hands-on understanding of XML and SOAP, in this lab we will build a SOAP web service with simple operations such as *Multiply*, *Divide*, *Echo* and *Concat*. We will then test this service with the SOAPSonar tool, also installed on the Training Image.

Additional labs in this FS Sentry Training series will utilize this SOAP web service.

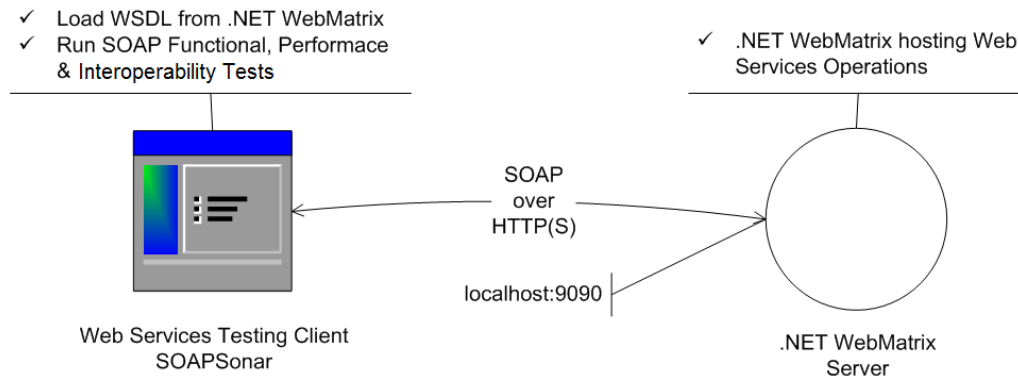
Lab 1 uses the following products that are installed on the FS Sentry Training Image.

1. Microsoft ASP.NET WebMatrix – this is a lightweight local application server that includes an IDE for building web services.
2. SOAPSonar Enterprise Edition – this is an enterprise class web services testing tool that supports testing of SOAP, XML, REST, JSON and other message flows. This tool will be used throughout the FS Sentry Training series.

Figure 1 below shows a typical web service deployment with a consumer-producer interaction model.

The producer is the ASP.NET Web Matrix server that supplies a web service with four operations (*Multiply*, *Divide*, *Echo*, *Concat*) that applications can invoke, typically remotely over HTTP(S). In addition, it produces the WSDL file that defines the web service interface.

This file provides all the necessary information for the consumer, SOAPSonar, to send SOAP requests to the target web service. SOAPSonar consumes and interprets the WSDL-based API published by the producer and invokes the web service.



**Figure 1: Web Services Consumer-Producer Setup.**

This lab will provide instructions for configuring a SOAP Web service with the .NET WebMatrix server and testing this service with SOAPSonar. Topics will include:

1. Running a SOAP Web service with .NET WebMatrix.
2. Introduction to SOAPSonar.
3. Testing the SOAP service with SOAPSonar.

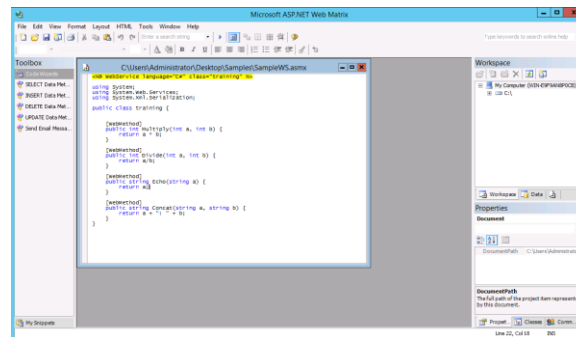
# A Simple SOAP Web Service

In this step we will configure .NET Web Matrix to run a simple SOAP Web service.

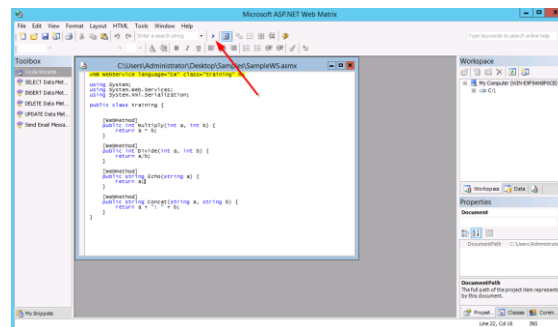
## Configuring .NET Web Matrix

Follow the steps below to configure .NET Web Matrix.

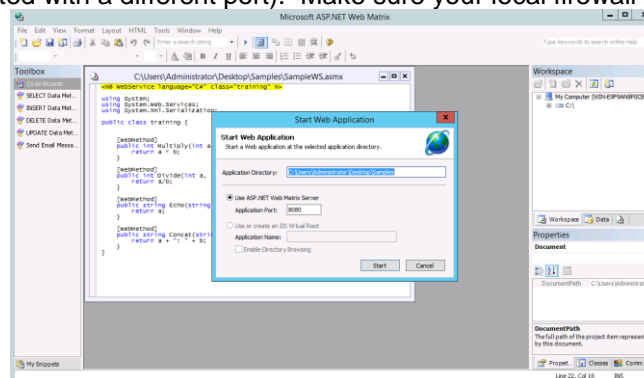
1. Double-click the “ASP .NET Web Matrix” icon on the desktop or launch from the Start Menu.
2. You will be prompted with a new file screen. Hit Cancel.
3. From Web Matrix IDE, select *File*→*Open Files*. Go to the *Samples* folder on the Desktop and select *SampleWS.asmx*. You will see the following C# code:



4. Click the *Start* button in the IDE as shown in the Figure below:



5. When prompted, click *Start* to launch the web application on port 8080 (change to port 8080 if you are prompted with a different port). Make sure your local firewall is turned off.



6. A web browser with a list of operations will appear. You can click the operation names and start experimenting with your first web services.

## SOAPSonar

SOAPSonar is an enterprise class Web services testing tool that is capable of testing many types of services, including: SOAP, XML, REST, JSON, HTML, EDI, FTP, HTML, etc.

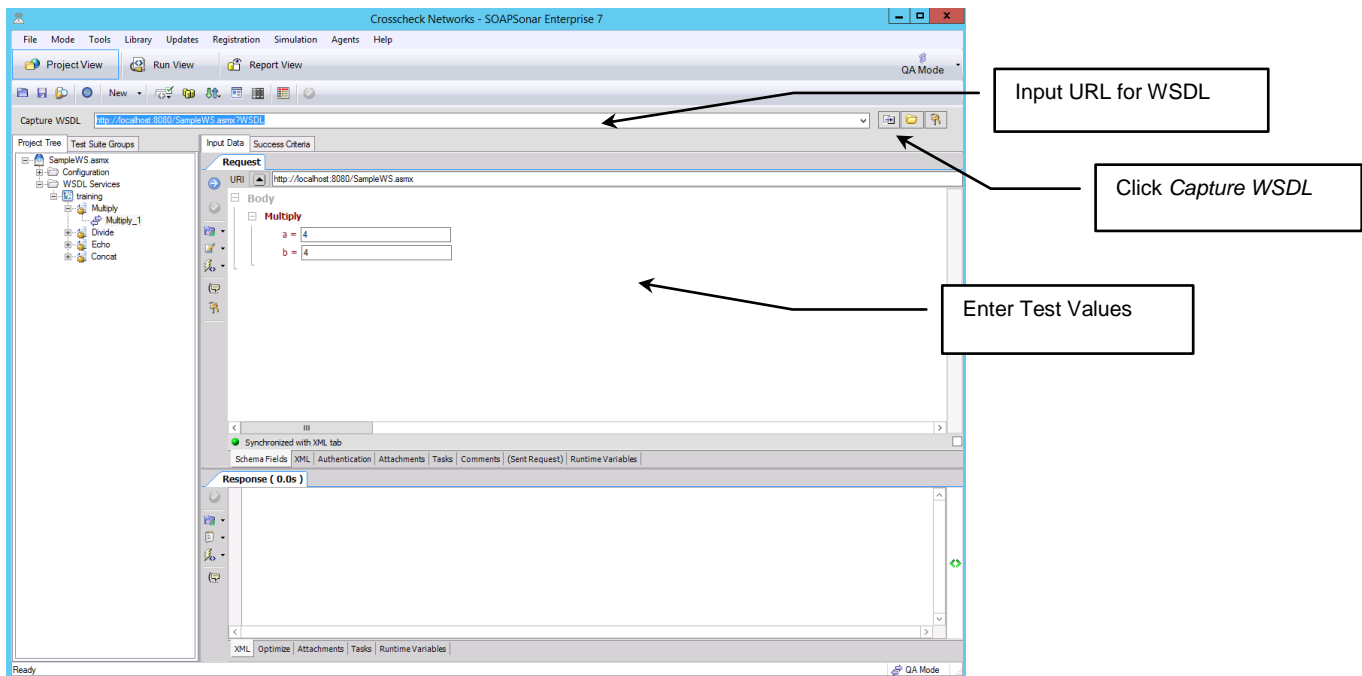
This tool will be used extensively throughout the Sentry Training Labs.



### Testing the SOAP Service

In this step we will configure the SOAPSonar tool to test the SOAP service.

**Follow the steps below to begin testing the SampleWS SOAP service with SOAPSonar.**

1. Start SOAPSonar by clicking on the SOAPSonar icon on the Desktop.
2. Load the WSDL published at the .NET WebMatrix Endpoint <http://localhost:8080/SampleWS.asmx?WSDL> into SOAPSonar as shown in the figure below. Under "Project Tree" you'll need to expand out to the Multiply\_1 test case that was created automatically by SOAPSonar.



3. Click  to commit changes and  to execute the test. These icons are found in the right pane on the top left. You will see the SOAP response in the *Response* Panel.
4. Save the project by going to *File > Save Project As*.
5. This concludes Lab 1. With the WSDL loaded into the test client, SOAPSonar, you now have a simple consumer (SOAPSonar) to producer (WebMatrix) Framework setup to perform comprehensive SOA Testing.

## WSDL and SOAP Basics

The following is very basic information about WSDL and SOAP. More information on a WSDL document can be found at the end of this lab under Additional Reading.

## WSDL File

A WSDL file tells the client application (e.g. SOAPSonar) how to build the properly formatted SOAP request message for the service.

The WSDL also defines the format of a proper SOAP response message from the service.

The WSDL file also lists the service endpoints that the client application should send the SOAP request to.

Most services expose the WSDL file via URL using the ?WSDL notation. For instance, the service location for the W3Schools Temp Convert SOAP Service is:

<https://www.w3schools.com/xml/tempconvert.asmx>

The WSDL can be retrieved by adding ?WSDL to the end of the URL:

<https://www.w3schools.com/xml/tempconvert.asmx?WSDL>

## SOAP Message

W3Schools.com defines a SOAP message as an ordinary XML document containing the following elements:

- An **Envelope** element that identifies the XML document as a SOAP message
- A **Header** element that contains header information
- A **Body** element that contains call and response information
- A **Fault** element containing errors and status information

All the elements above are declared in the default namespace for the SOAP envelope:

- <http://www.w3.org/2001/12/soap-envelope>

and the default namespace for SOAP encoding and data types is:

- <http://www.w3.org/2001/12/soap-encoding>

Below is a skeleton SOAP Message:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
```



```
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Header>
...
</soap:Header>

<soap:Body>
...
  <soap:Fault>
    ...
  </soap:Fault>
</soap:Body>

</soap:Envelope>
```

## Why Use SOAP?

The following is taken from the W3schools.com tutorial for SOAP:

...  
It is important for application development to allow Internet communication between programs.

Today's applications communicate using Remote Procedure Calls (RPC) between objects like DCOM and CORBA, but HTTP was not designed for this. RPC represents a compatibility and security problem; firewalls and proxy servers will normally block this kind of traffic.

A better way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. SOAP was created to accomplish this.

SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.

...

Some of the advantages of using SOAP for service communication include:

- Better interoperability by using a standard language format.
- Easier to understand the expected behavior and communication patterns from a service.
- Enables a more flexible approach to reusability for services.
- Provides more robust security and validation controls for the messages.
- Allows a simpler design of client applications.

**END**

## Additional Testing and More Reading

### BACK IT UP!

It is recommended that you save SOAPSonar project after completing this lab.

To export the SOAPSonar configuration, use the File→Save As option. The SOAPSonar configuration is saved as an .SSP file. Saving the configuration file(s) after each lab is recommended. In some cases you may choose to use new SOAPSonar project files for new labs.

We recommend including the lab number in the name of the saved project files.

## Additional Tests and Discussion Topics

1. Explore the SOAPSonar product. There are multiple views and modes.
2. We recommend visiting W3Schools.com for more information on SOAP and WSDL.

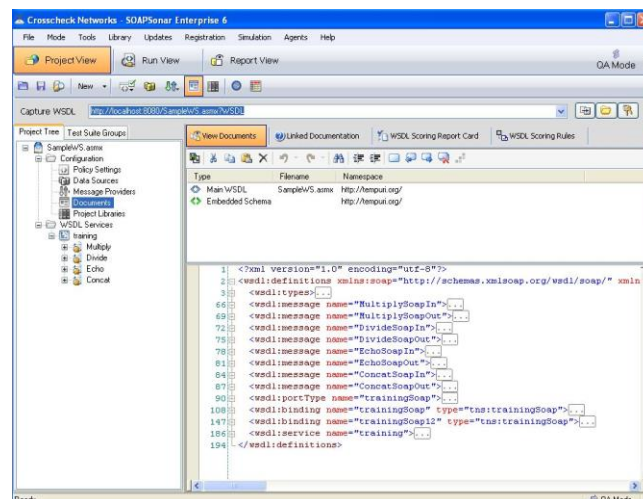
## Additional Information

### Understanding a WSDL File

WSDL is an XML-based language to *define* and *locate* a web service. A WSDL file describes the functionality of a web service and how it should be accessed.

The WSDL document is typically generated by the web services producer and provided to a web services consumer or client. Understanding a WSDL file is fundamental to building effective test suites. Typically, a QA professional is handed a WSDL file to determine the quality of a web service. They may or may not have access to component source code for White-Box testing. In SOA deployments with integration with external trading partners access to source code is typically never provided. In such cases, the WSDL is the only test material that a QA professional can utilize to build test suites. Therefore, understanding a WSDL file is critical in effectively testing a modern web services-based SOA deployment.

1. We begin by opening the WSDL file for the *SampleWS.asmx* web service. As shown in the figure below, select Configuration > Documents. The WSDL document is displayed fully expanded. Click Expand[+]/Collapse[-] Nodes to Collapse the WSDL document to the state shown below. We will now explore parts of the WSDL independently.



2. **Types:** We begin examining the WSDL file by expanding Line 3 `<wsdl:types>` as shown in the Figure below. The Types defines the data types used in the web service. In the Figure below, we focus on the *Multiply* element and the *MultiplyResponse* element. The *Multiply* type is a complex type that consists of 2 elements *a* and *b* both of type *int*. The *MultiplyResponse* consists of a *MultiplyResult* element of type *int*.

```
3 <wsdl:types>
4   <s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
5     <s:element name="Multiply">
6       <s:complexType>
7         <s:sequence>
8           <s:element minOccurs="1" maxOccurs="1" name="a" type="s:int"/>
9           <s:element minOccurs="1" maxOccurs="1" name="b" type="s:int"/>
10        </s:sequence>
11      </s:complexType>
12    </s:element>
13    <s:element name="MultiplyResponse">
14      <s:complexType>
15        <s:sequence>
16          <s:element minOccurs="1" maxOccurs="1" name="MultiplyResult" type="s:int"/>
17        </s:sequence>
18      </s:complexType>
19    </s:element>
```

3. **Message:** Next we examine the *Message* elements that define the messages for the web service. Expanding line 66, as shown below, displays the message *MultiplySoapIn* that refers to the type element *Multiply* in the `<wsdl:part ... >` on line 67. The *part* element ties the message *MultiplySoapIn* to the *Multiply* Message type, which is a complex type with two integer inputs as show in Step 2 above.

```
66 <wsdl:message name="MultiplySoapIn">
67   <wsdl:part name="parameters" element="tns:Multiply" />
68 </wsdl:message>
69 <wsdl:message name="MultiplySoapOut">
70   <wsdl:part name="parameters" element="tns:MultiplyResponse" />
71 </wsdl:message>
```

4. **portType:** The portType is an important part of the WSDL that defines all the operations available in the WSDL. In our example, we expand Line 90 and examine operation named *Multiply* that shows an input message *MultiplySoapIn* on Line 92 and an output message *MultiplySoapOut* as described in the Step 3. Each operation is the equivalent of a programming function. In our example, there are four functions that map to four operations. Each operation has a request-response message pair.

```
90 <wsdl:portType name="trainingSoap">
91   <wsdl:operation name="Multiply">
92     <wsdl:input message="tns:MultiplySoapIn" />
93     <wsdl:output message="tns:MultiplySoapOut" />
94   </wsdl:operation>
```

5. **Binding:** With the next two sections of the WSDL, *Binding* and *Service*, we are now making a transition from abstract data types, messages, and operations to the concrete physical representation of messages on the wire. The *Binding* defines whether an operations in the WSDL

file is “document oriented” or “remote procedure call (RPC) oriented,” as defined by the **style** attribute of the `<soap:operation>` element, Line 111. If an operation in the WSDL file is document-oriented, the input (request) and output (response) messages specified for that operation contain XML documents. RPC-oriented operations have input messages that contain the operations' input parameters and output messages that contain the operations' results instead of XML documents. The `<soap:body>` element on Line 113 that can either be “literal” or “encoded.” Typically, documents are literal and RPCs are encoded. For interoperability a document/literal binding is recommended.

```
108 <wsdl:binding name="trainingSoap" type="tns:trainingSoap">
109   <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
110   <wsdl:operation name="Multiply">
111     <soap:operation soapAction="http://tempuri.org/Multiply" style="document" />
112     <wsdl:input>
113       <soap:body use="literal" />
114     </wsdl:input>
115     <wsdl:output>
116       <soap:body use="literal" />
117     </wsdl:output>
118   </wsdl:operation>
```

6. **Service:** The final part of a WSDL file is the `<service>` element shown below by expanding line 186. The `<service>` element defines a physical location for a communication end-point. It uses the port type and binding (to be described later), and gives the Web address or URI for a particular provider of the described service. In our sample web service, the endpoint location is <http://localhost:8080/SampleWS.asmx>. This is the physical URI with IP address *localhost* and port *8080* for our web service running on the webMatrix Server installed in Lab 1.

```
186 <wsdl:service name="training">
187   <wsdl:port name="trainingSoap" binding="tns:trainingSoap">
188     <soap:address location="http://localhost:8080/SampleWS.asmx" />
189   </wsdl:port>
190   <wsdl:port name="trainingSoap12" binding="tns:trainingSoap12">
191     <soap12:address location="http://localhost:8080/SampleWS.asmx" />
192   </wsdl:port>
193 </wsdl:service>
```

## About Forum Systems

Forum Systems is the global leader in API and Cloud Security technology with industry-certified, patented, and proven products deployed in the most rigorous and demanding customer environments worldwide. Forum Systems has been an industry leader for over 12 years and has built the core architecture of its technology on the foundation of FIPS 140-2 and NDPP. Forum Systems security-first mindset enables trusted, network edge deployments of its technology for protecting critical enterprise transactions.

Our product technology is purpose-built and designed for mission-critical, enterprise-class scalable solutions where business solutions require the modern day security and identity enforcement protection, while enabling a scalable architecture and low-latency, high-volume throughput.

Forum Systems supports global enterprise customers across industries in commercial, government, and military sectors. Forum Systems technology provides the leading-edge of modern-day cyber-security innovation with integrated identity and SSO features that enable out-of-the box business solutions with point-and-click technology.

Forum's patented, FIPS 140-2 and NDPP certified hardware and virtual products make modern-day business communications secure by actively protecting and accelerating data exchange and API service access across networks and business boundaries. For more information, please visit [www.forumsys.com](http://www.forumsys.com).