



Development of Web Services using
Rational Application Developer & WebSphere Application Server



Table of Contents

Objectives	4
Abbreviations.....	4
What are Web Services?	4
What are the Key Benefits of Web Services.....	5
Software Requirements.....	7
What is SOAP?	7
What is WSDL?	8
What is UDDI?	8
How Does it Work?	8
Creating Java Web Service	8
Creating Java Web Service Client.....	9
Web Services Security	10
Web Services Security Risk Exposures.....	10
Transport Level Security for Web Services	12
HTTP Basic Authentication	12
Enable Security on the Server (Global Security).....	13
HTTP Basic Authentication for WebService	13
HTTP Basic Authentication for WebService Client	14
HTTP/SSL (HTTPS) Authentication	15
Create Keystore, Truststore and Certificate for Web Service.....	15
Create SSL Configuration for the Web Services	16



Create Web Container Transport Chain to Use the New SSL Configuration	16
Configure SSL for the Client	16
Message Level Security for Web Services (WS- Security)	17
Authentication	18
Configure Web Services Client for Security Token	18
Configure Web Services for Security Token	19
Enable TCP Monitor to Review SOAP Headers	19
Confidentiality	20
Generate Key Stores and Certificates	20
Create Key Store and Certificate for Web Services	20
Create Key Store and Certificate for Client	21
Create the Certificate Files	21
Add the Public Certificates to the Opposite Key Store	22
Configure the Client to Specify the Confidentiality Part	22
Configure the Server to Specify the Confidentiality Part	23
Confidentiality for the Response Message	24
Integrity	25
Configure the Client to Specify the Integrity Part	25
Configure the Server to Specify the Integrity Part	26
Integrity for the Response Message	28
References	28

Objectives

The hotspot in the current IT world is optimized Web Services. Now the enterprises have started their vision towards next generation Web Services. This white paper gives information for the business managers and engineers in developing and deploying Web Services using Rational Application Developer (RAD) and WebSphere Application Server (WAS), respectively. This paper explains the process to

- create a Web Service and Web Server client
- build security to that web service
- deploy the web service in WebSphere application server

Abbreviations

RAD	Rational Application Developer
WAS	WebSphere Application Server
SOAP	Simple Object Access Protocol
WSDL	Web Services Description Language
UDDI	Universal Description, Discovery and Integration
TLS	Transport Layer Security
SSL	Secure Sockets Layer

What are Web Services?

Web Services are relatively a new technology that implements service-oriented architecture. The development of this technology, involves a major focus on making functional building blocks accessible over standard Internet protocols that are independent from platforms and programming languages.

Web Services are self-contained, modular applications that can be described, published, located and invoked over networks. Web Services encapsulate business functions, ranging from a simple request-reply to full business process interactions and can be new or wrapped around existing applications.

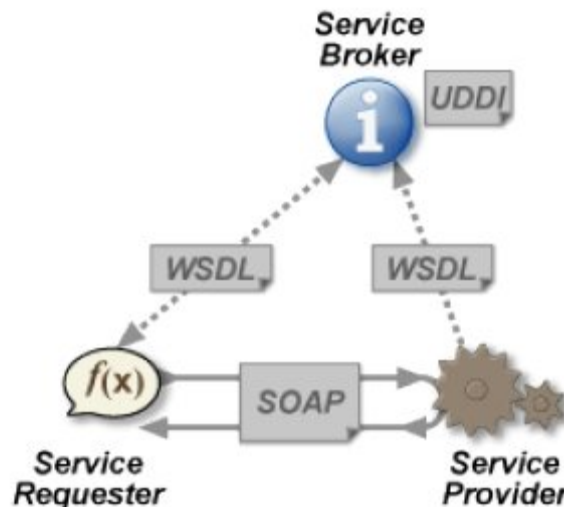
What are the Key Benefits of Web Services

Reusable Application Components

There are some components which different applications need it very often. So why not to develop a component and re-use it? These Re-usable Application Components are generally referred by the term, 'componentization'. Web Services offers developers the componentization feature for application components like currency conversion, weather reports, language translation, etc

Connecting Different Applications/Software

Web Services helps to solve the interoperability problem whereby you have a way to link data of different applications and exchange data between different applications and platforms.



Web Service Architecture

Web Services are self-contained

The client side requires no additional software. To start with, on the client-side a programming language with XML and HTTP client support is required and on the server side a HTTP server and SOAP server is required. It is possible to enable an existing application for Web Services without writing a single line of code.

Web Services are self-describing

The definition of the message format is transferred along with the message and requires no external metadata repositories or code generation tools.

Web Services can be published, located and invoked across the Web

The Web Services uses lightweight Internet standards such as HTTP and some additional standards—SOAP, WSDL and UDDI to leverage the existing infrastructure.

Web Services are modular

Simple Web Services can be aggregated to more complex ones, either using workflow techniques or by invoking lower-layer Web Services from a Web Service implementation. Web Services can be chained to perform higher-level business functions, which shorten the development time and enables best-of-breed implementations.

Web Services are language independent and interoperable

Web Services being language independent, any language can be used to implement Web Service clients and servers and the environment for implementation need not necessarily be the same, which can also be different. It is also not essential to change the existing code to be enabled with Web Service.

Web Services are inherently open standard

XML and HTTP are the major language and technology used in Web Services; hence you can use open source tools and technologies to build Web Services, thus making vendor independence and interoperability as realistic goals.

Web Services are loosely coupled

Traditionally, designing application required a lot of tight interconnections at both ends but Web Services require a simple level of coordination that allows more flexible reconfiguration for integration of various applications and services.

Web Services are dynamic

Web Services makes e-business dynamic by automating Web Service description and discovery using UDDI (Universal Description, Discovery and Integration) and WSDL (Web

Services Description Language). In addition, implementation and deployment of Web Services is easy and does not disturb the client using it.

Web Services provide programmatic access

The approach provides no graphical user interface; it operates at the code level. Service consumers have to know the interfaces to Web Services but do not have to know the implementation details of services.

Web Services enables to integrate already existing and new applications

Web Services provides an interface that enables to integrate stand-alone applications into the service oriented architecture.

Software Requirements

The basic software requirement is to install latest Rational Application Developer V7 pack to build Web Services is to install latest IBM and install WebSphere Application Serverv6.1 for deployment. Note that IBM releases a fix pack approximately every three months. It is highly recommended to install the latest version to avoid any encountering problems that have already been fixed. To ensure and update RAD with latest fixes, go to the IBM Installation Manager and select Update Packages.

- Web Services Platform
 - XML
 - HTTP
- Web Services Platform Elements
 - SOAP (Simple Object Access Protocol)
 - UDDI (Universal Description, Discovery and Integration)
 - WSDL (Web Services Description Language)

What is SOAP?

SOAP is

- a communication protocol for communication between applications
- a format for sending messages
- designed to communicate via Internet
- platform independent
- language independent

- based on XML
- simple and extensible
- to get you around firewalls
- developed as a W3C standard

What is WSDL?

WSDL is a language written in XML to describe and locate Web Services and is not yet a W3C standard.

What is UDDI?

UDDI is a directory service where businesses can register and search for Web Services. It is built into the Microsoft.NET platform and a directory of Web Service interfaces described by WSDL and communicates through SOAP.

How Does it Work?

To develop a Web Service, you need to create Java Web Service and Java Web Service Client.

Creating Java Web Service

For example, let's consider creating Java Web Service for 'Hello World'. Follow the steps given below:

1. Click 'File'→'New'→'Other'→'Web'→'Dynamic Web Project'
2. Enter the project name, say for example, 'Helloworld' and click 'Finish'.
3. Create a java file, say, 'HelloWorld' in src folder. For Example:

```
package zsl;
public class HelloWorld
{
    public String sayHello(String name)
    {
        return "Hello "+name;
    }
}
```

4. Right-click 'HelloWorld.java' and select 'File'→'New'→'Other'→'Web Services'→'Web Service'.
5. Click 'Next' in the subsequent screens and ensure that selection is made for 'sayHello method'.

6. Start the server.
7. Click 'Finish'. Now the server is started; Web Service and HelloWorld.wsdl file is created.
8. Go to WebContent→WEB-INF→wsdl and open the -->HelloWorld.wsdl file.
9. Change the wsdlsoap:address from localhost to your IPaddress.
10. Copy the wsdlsoap:address, for example,
<http://IPaddress:9080/HelloWorld/services/HelloWorld> to notepad and append
"/wsdl".
<http://IPaddress:9080/HelloWorld/services/HelloWorld/wsdl>, which can be served
as wsdl address.

Creating Java Web Service Client

Once Java Web Service is created, follow the below steps to create Java Web Service Client:

1. Click 'File'→'New'→'Other'→'Web'→'Dynamic Web Project'
2. Enter the project name ,say for example, 'HelloWebClient'
3. Click 'Finish'.
4. Right-click 'TestClient' and 'New'→'Other'→'Web Service Client'.
5. In the service definition, paste the wsdl url as we created above.
6. Click 'Finish'. Now you can view the list of files generated in src folder.
7. Right-click 'WebContent' and select 'New'→'Web Page'. Enter the file name, here, 'TestClient' and click 'Finish'.
8. Write the following code in 'TestClient.jsp' to access the Web Service.

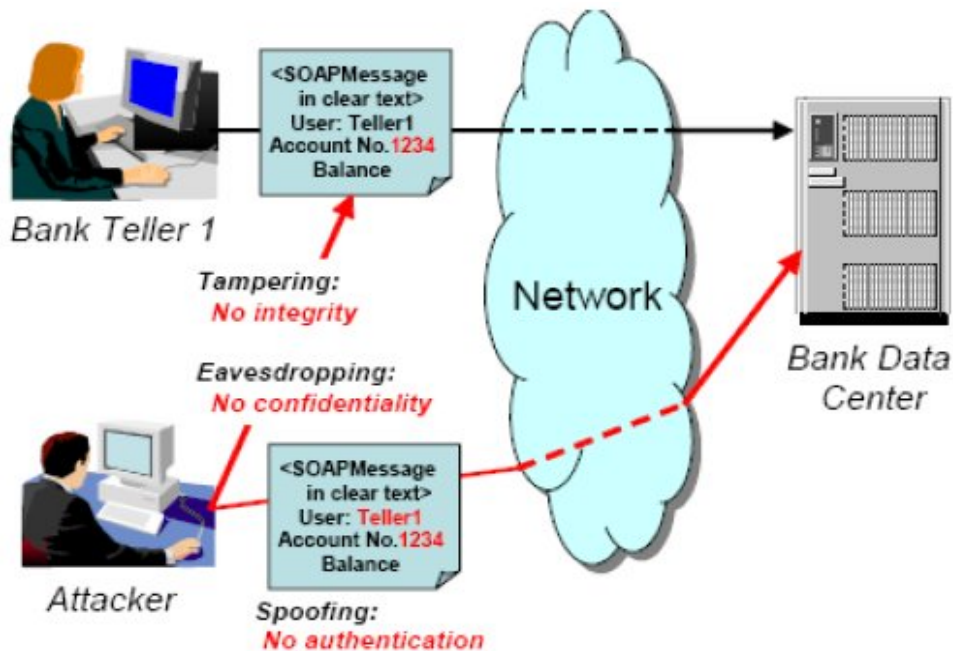
```
<%@ page import="zsl.HelloWorldProxy"%>
Body:
<%
HelloWorldProxy hwp=new HelloWorldProxy();
out.println(hwp.sayHello("zylog"));
%>
```
9. Right-click 'TestClient.jsp' and select 'Run as'→'Run on server' and check the result.

Web Services Security

Web Services Security Risk Exposures

Web Services, too, possess security threats as other Internet, middleware-based applications and communications. Hence securing Web services is an essential requisite.

Let's consider a scenario of bank application built on unsecured Web Services and a bank teller using the application for providing the banking services to the customers.



Common Security Exposures in a Sample Web Services Application

In this case, the bank teller may have the probability of facing the following three major risks:

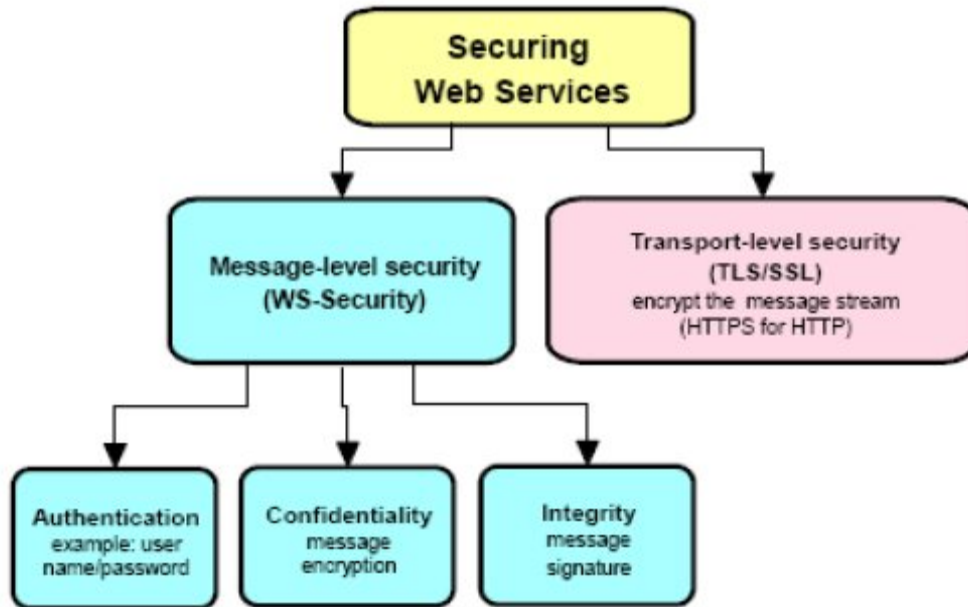
- **Spoofing:** This occurs when the Web Services are not provided with authentication security measures. In this case, any unauthorized person can pretend as bank teller send a modified SOAP message to the service provider to get confidential information or to withdraw money from another customer's account. To prevent this

type of risk exposures, the Web Services has to be provided with authentication security measures.

- Tampering: This type of risk occurs, when there is no integrity between the Web Service and server. In this case, the SOAP messages can be intercepted between the Web Service client and server and can be modified. For example: if a SOAP message is for transferring money from account 'A' to account 'B', in the absence of integrity security measures, the message can be modified for transferring money from account 'A' to account 'C' and be accepted without any validation. To arrest this risk exposure, the Web Services need integrity mechanism.
- Eavesdropping: This type of risk occurs when the Web Services is not provided with confidentiality mechanism. In this case, the SOAP message can be intercepted and its information can be read and in the absence of encryption, the confidential or bank information may reach the hands of irrelevant or wrong person. This happens because the message is sent as a plain text. To prevent this security risk exposure, the Web Services has to be provided with confidentiality mechanism.

To prevent the security risk exposures, the Web Services has to be provided with the following security mechanisms:

- Transport-level security — TLS/SSL
- Message-level security — Authentication/Confidentiality/Integrity



Securing Web services

Transport Level Security for Web Services

The HTTP is the most commonly used protocol for Web Services, which is highly insecure because of its information transmission in plain text format between the unauthenticated peers over an insecure network. To prevent this HTTP's has to be secured with transport-level security mechanism is needed. Transport-level security is based on Secure Sockets Layer (SSL) or Transport Layer Security (TLS) that runs beneath HTTP. HTTPS allows client-side and server-side authentication through certificates signed either by self or certification agency. The Web services bound to the HTTP protocol, HTTPS/SSL can be applied in combination with message-level security (WS-Security).

To secure transport level security for the Web Services, follow the process described in the below sections.

HTTP Basic Authentication

In Websphere Application Server V6.1 global security is split into administrative security and application security, each of which can be enabled separately.

To enable HTTP basic authentication, run WebSphere Application Server when both administrative security and application security are activated.

Enable Security on the Server (Global Security)

1. Go to 'Programs' → 'IBM Software Development Platform' → 'IBM Rational Application Developer' configure and start the server.
2. In Servers view, right-click the server and select 'Run Administrative Console'.
3. In administrative console, expand 'Security' → 'Secure Administration' → 'Applications' → 'Infrastructure'. Click 'Security Configuration Wizard'.
4. Select 'Enable application security'.
5. Click 'Next'.
6. Select 'Local operating system'.
7. Click 'Next'.
8. Enter the primary administrative user name that has administrative privileges for the local operating system.
9. Click 'Next'.
10. In the 'Summary' screen, click 'Finish' and then 'Save'. Both the administrative security and application security are enabled.
11. Click 'Logout' to log off from the administrative console.
12. Stop the server.
13. In the Servers view, double click the server.
14. In 'Server Overview', expand Security and check if Security is enabled on this server.
15. Enter the same user ID and password that was entered in the 'Security Configuration Wizard' dialog box of the 'WebSphere Administrative Console'.
16. Save and close.
17. Start the server and open the administrative console. The administrative console is now secured.

HTTP Basic Authentication for WebService

1. In the Project Explorer, expand the 'Web Service' (Ex: Helloworld), then double click 'Deployment Descriptor: Web Service'. The 'Web Deployment Descriptor' opens.
2. Click 'Servlets' tab. Expand 'Servlets' and 'JSPs'. Click the 'service'. In URL Mappings a URL will be displayed. Note that URL. (Ex: Servlet mapping (services/Helloworld ->package_Helloworld))
3. Click 'Pages' tab. Expand 'Login'
 - Authentication Method : Basic
 - Realm Name : HTTP Basic Authentication

4. Click 'Security' tab. Expand 'Security Roles'. Click 'Add'.
5. In 'Add Security Role' dialog box, enter the 'security role name' and 'description'. Click 'Finish'.
6. Click the newly defined security role, then under 'Security Constraints', click 'Add'.
7. In 'Add Security Constraints' dialog box, enter the 'constraint name'. Click 'Next'.
8. In 'Add Web Resource' dialog box, enter the 'resource name' and 'description'.
9. In HTTP methods, select the 'get and post' method.
10. Under 'pattern' click 'Add'. Type /services/service name. (from URL mapping under servlet tab). Click 'Finish'.
11. Expand 'Authorized Role'. Click 'Add'. Specify the description and select the newly created role name. Click 'Finish'.
12. Save and Close the Web Deployment descriptor.
13. In the Project Explorer, expand the 'Web service EAR' (Ex: HelloworldEAR), then double click 'Deployment Descriptor: Web serviceEAR'. The 'Application Deployment Descriptor' opens.
14. Click 'Security' tab.
15. Click 'Gather'. This gathers all the security roles defined in all the modules. Note that the 'Webservicesrole' is added.
16. Click 'Webservicesrole' and expand 'Websphere Bindings', select 'Users/Groups'.
17. Click 'Add'. In the 'User' dialog box, enter a valid user name. Because WebSphere Application Server is configured to use the local operating system user registry, the system requires a valid user account that can be authenticated by the operating system. To make it simpler, enter the primary administrative user name used in the 'Security Configuration Wizard'.
18. Click 'Finish'. Save and close the editor.
19. Now run a client to call this web service. It provides an unauthorized exception.

HTTP Basic Authentication for WebService Client

1. In the Project Explorer, expand the client project (here 'HelloWebClient'), and double click 'Deployment Descriptor: Web serviceClient'. The 'Web Deployment Descriptor' opens.

2. Click the 'WS Binding' tab. Under the 'Port Qualified Name Binding Details' section, locate HTTP basic authentication. Enter the valid user ID and password that was configured for the Web Service.
3. Save and close the descriptor and test the Web Service with a client.

HTTP/SSL (HTTPS) Authentication

SSL Basics

The SSL protocol is based on public key cryptography and relies on the existence of digital certificates. A digital certificate contains both public and private keys and it reveals the information about its owner including identity. The messages encrypted with one of the keys can be decrypted only with the corresponding key in the key pair. The public key (called the signer certificate) can be extracted to a file and imported to the client's trust store. The client requires the signer part of a digital certificate for SSL communication.

IBM WebSphere Application Server supports the concept of two types of key store:

- Key store file, which contains a collection of certificates and the associated private key for each certificate.
- Trust store file, which contains a collection of certificates that are considered trustworthy and against which the presented certificate is matched during an SSL connection initiation to assure identity.

Create Keystore, Truststore and Certificate for Web Service

1. In Servers view, right-click and select 'Run Administrative Console'.
2. In 'administrative console', expand 'Security'→'SSL Certificate'→'Dey Management'. Under 'Related Items', select key stores and certificates.
3. Click 'New' and enter the following values:
 - Name : serverkeystore
 - Path : serverkeystore.jks (create this file to your workspace)
 - Password : server
 - Type : JKS
4. Click 'OK' and 'Save'. The key store is created.
5. Click 'New' and enter the following values:
 - Name : servertruststore
 - Path : servertruststore.jks (create this file to your workspace)

- Password : server
 - Type : JKS
6. Create the personal certificate for the server key store. It holds both public and private keys.
 7. Select the 'server key store'.
 8. Under 'Additional Properties', click 'Personal Certificates'.
 9. Click 'Create a self-signed certificate' and enter the following values:
 - Alias : server_rsa
 - Common name : Server
 - Organization : IBM
 10. Click 'OK'.

Create SSL Configuration for the Web Services

1. In 'administrative console', expand 'Security' → 'SSL certificate' → 'key management'. Under 'Related Items', select 'SSL configurations'.
2. Click 'new'. Enter a name and select 'keystore' and 'truststore'.
3. Click 'Get certificate alias', then 'OK' and 'Save'.

Create Web Container Transport Chain to Use the New SSL Configuration

1. Expand 'server' → 'application server' → 'server1'.
2. Under 'Container settings' → 'web container settings' → 'web container transport chain', click 'New' and enter a name. Click 'Next'.
3. Enter the port name, host and port (a new port no).
4. Click 'Next', then 'Finish' and 'Save'.
5. Click the web container transport chain and SSL inbound channel (SSL_4).
6. Select 'SSL configuration' → 'ssl configuration name' → 'OK' → 'Save'.
7. Expand 'Environment' → 'Virtual hosts' → 'Default host'. Under 'Additional Properties' → 'host aliases', click 'New'.
8. Enter the newly created port and click 'OK', then 'Save'.

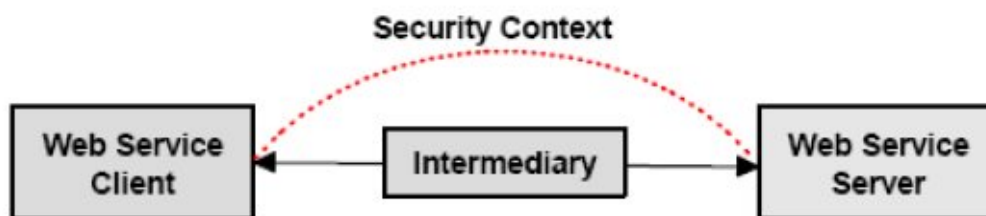
Configure SSL for the Client

1. In 'administrative console', expand 'Security' → 'SSL certificate and key management'. Under 'Related Items', select 'key stores and certificates'.
2. Click 'new'. Enter a name and select 'keystore' and 'truststore'.

3. Now import the Web Services provider's certificate directly into the client's default trust store.
4. Click 'NodeDefaultTrustStore'.
5. Go to 'Additional Properties', click 'Signer Certificates'. The list of the signer certificates is displayed.
6. Click 'Retrieve from port' and enter the values.
7. Click 'Retrieve signer information', then 'OK' and 'Save'.
8. Test the Web Service with a client.

Message Level Security for Web Services (WS- Security)

The WS-Security specification, namely, Web Services Security: SOAP Message Security 1.0 (WS Security 2004), is proposed by the OASIS WSS Technical Committee. This specification defines a standard set of SOAP extensions and is flexible. The specification is designed to be used as the basis for securing Web Services within a wide variety of security models, including PKI, Kerberos and SSL. It provides support for multiple security token formats, multiple trust domains, multiple signature formats and multiple encryption technologies based on XML signature and XML encryption to provide integrity and confidentiality. The specification includes security token propagation, message integrity and message confidentiality at the message level. The advantage of using WS-Security over SSL is that it provides end-to-end message level security, which means that the message security is protected even when the message goes through multiple services, called intermediaries.



End-to-end Security with Message Level Security

The message level security helps the strangers refrain from accessing an application and to retain and secure the confidentiality of the information in the messages sent through the Web Services. There are three types of Message Level Security mechanisms:

- Authentication: The mechanism to validate and provide authentication for a user ID and password to access the Web Service and perform an activity using the Web Service.
- Confidentiality: The messages are encrypted or coded to secure the confidentiality of the information in the messages sent through the Web Services.
- Integrity: Is the assurance that information can only be accessed or modified by those authorized to do so.

Authentication

To secure the Web Services with authentication security mechanism, follow the process described in the below sections.

Configure Web Services Client for Security Token

Once Web Services security is enabled, the client-side SOAP security handlers generate the SOAP headers and insert them in the SOAP message request. After the server-side SOAP security handlers parse the SOAP request message for the security token, use the user ID and password in the token to authenticate the user with the user registry configured in Application Server.

1. In the Project Explorer, expand the 'client project' (HelloWebClient) and double click 'Deployment Descriptor: WebClient'. The Web deployment descriptor opens.
2. Click 'WS Extension' tab. Expand 'Request Generator Configuration'→'Security Token'. Click 'Add'.
3. In the 'Security Token' dialog box, enter the name and token type. Enter the user name token for token type. Click 'OK'.

Now callback handler has to be added to add the security headers username token to the soap request.

4. Click 'WS Binding' tab. Expand 'Security Request Generator Binding Configuration'→ 'token Generator' and click 'Add'. The 'Token Generator' dialog box opens.
5. In the 'Token Generator' dialog box, type the token generator name and security token.
6. Select 'Use Value Type'. Enter the value type (username token), user ID and password. Click 'OK'.
7. Save and close the descriptor.

Configure Web Services for Security Token

1. In the Project Explorer, expand the Web Service (here 'HelloWorld') and webcontent — WEB-INF webservices.xml.
2. In the 'Web Services Editor', click 'Extensions' tab. Expand 'Request Consumer Service Configuration Details' → 'Required Security Token' and click 'Add'.
3. In the dialog box type the name and type. Click 'OK'.
4. Expand 'Caller Part' and click 'Add'.
5. In the 'Caller Part' dialog box, enter the name and token type. Click 'OK'.
6. Click 'Binding Configurations' tab. Expand 'Request Consumer Binding Configuration Details' → 'Token consumer' and click add. A dialog box opens.
7. In the dialog box, enter the name and select the security token.
8. Select use value type.
 - a. Value type: select username token.
9. Select use jaas.config. jaas. Config name : system.wssecurity.UsernameToken (This validates a username token with the user name and password.)
10. Click 'OK'.
11. Save and close the editor.
12. Run the Web Service and client. (Ensure that global security was configured on the server).

Enable TCP Monitor to Review SOAP Headers

The TCP monitor can be used to intercept and examine the SOAP traffic coming in and out of a Web Service. This monitor is a proxy to the server hosting the Web Service. Two steps have to be completed two steps to tool: i) configure the TCP/IP monitor and ii) redirect the client to send messages to the monitor instead to the Web Service.

1. Go to 'Windows' → 'Preferences' → 'Run'/'Debug' → 'TCP/IP Monitor'. Click 'Add'.
2. Local monitoring port is a proxy port. Enter the following"
 - New port no : 9089 (Local monitoring port which is a proxy port.)
 - Host name : local host (the host where the service is running)
 - Port : 9080 (actual port where the service is running)
 - Type : HTTP

3. Click 'OK' and then click 'Start'.

Now run the client and change the endpoint from 9080 to 9089 (monitoring port).

Confidentiality

To provide confidentiality on the client's request message, the request message has to be encrypted. Before starting the WS-Security configuration, prepare the client and server key stores to sign or encrypt the message.

Follow the process described in the below sections to build confidentiality security mechanism for the Web Services.

Generate Key Stores and Certificates

The WebSphere Application Server administrative console can be used to generate key stores.

Create Key Store and Certificate for Web Services

1. In Servers view, right-click and select 'Run Administrative Console'.
2. In 'administrative console', expand 'Security'→'SSL Certificate and Key Management'. Under 'Related Items', select key stores and certificates.
3. Click 'New' and enter the following values:
 - Name : server
 - Path :server.jks (Create this file in your workspace ex: d:\myworkspace\server.jks)
 - Password : server
 - Type : JKS
4. Click 'OK' and 'Save'. The key store is created.

Now create the personal certificate for the server key store. A personal certificate holds both public and private keys.

5. Select the server key store.
6. Go to 'Additional Properties', click 'Personal Certificates'.
7. Click 'Create a self-signed certificate' and enter the following values:
 - Alias : server_rsa
 - Common name : Server
 - Organization : IBM
8. Click 'OK'.

Create Key Store and Certificate for Client

1. In 'administrative console', expand 'Security'→'SSL Certificate and Key Management'. Go to 'Related Items', select key stores and certificates.
2. Click 'New' and enter the following values:
 - Name : client
 - Path :client.jks (Create this file in the workspace ex: d:\myworkspace\client.jks)
 - Password : client
 - Type : JKS
3. Click 'OK' and 'Save'. The key store is created.

Now create the personal certificate for the client key store. A personal certificate holds both public and private keys.

4. Select the client key store. Go to 'Additional Properties', click 'Personal Certificates'.
5. Click 'Create a Self-signed Certificate' and enter the following values:
 - Alias : client_rsa
 - Common name : Client
 - Organization : IBM
6. Click 'OK'.

Create the Certificate Files

The personal certificate has both public and private keys. However, while distributing the certificate, ensure only public key is sent and not the private key. To export the public key from the client key store, perform the following steps:

1. Go to 'Security'→'SSL Certificate and Key Management'→'Key Stores and Certificates'→'Client'→'Personal Certificates'.
2. Select the check box next to the newly created certificate client_rsa.
3. Click 'Extract'.
4. For certificate file name, specify:d:\RAD\converter\client_rsa.cer
5. Click 'OK'.

Follow the same steps to export the public key from the server key store.

Add the Public Certificates to the Opposite Key Store

Next, add the public server key certificate to the client key store, and the public client key certificate to the server key store:

1. Select 'SSL Certificate and Key Management'→'Key Stores and Certificates' → 'Client'.
2. Click 'Signer Certificates'.
3. Click 'Add'. and enter these values:
 - Alias: server_rsa
 - File name: server_rsa.cer (Create this file in your workspace
 - ex: d:\myworkspace\server_rsa.cer)
4. Click 'OK'.
5. Select 'SSL Certificate and Key Management'→'Key Stores and Certificates'→'Server'.
6. Click 'Signer Certificates'.
7. Click 'Add' and enter the following values:
 - Alias: client_rsa
 - File name: client_rsa.cer (Create this file in your workspace ex: d:\myworkspace\client_rsa.cer)
8. Click 'OK'.

Configure the Client to Specify the Confidentiality Part

1. In the Project Explorer, expand the client project (here, 'HelloWebClient'), and double click 'Deployment Descriptor: WebClient'. The 'Web Deployment Descriptor' opens.
2. Click 'WS Extension' tab. Expand 'Request Generator Configuration'→'Confidentiality'. Click 'Add'.
3. Enter the confidentiality name.
4. Enter order as '1'. (Enter the order as '2 if integrity and confidentiality is to be configured).
5. Go to message parts, click 'Add'. Go to parts keyword and select username token.
6. Go to message parts, click 'Add'. Go to parts keyword and select body token. Now in the soap request both the header and body is encrypted.
7. Click 'OK'.
8. Click 'WS Binding' tab.

9. Expand 'Security Request Generator Binding Configuration'→'Key Locators'. Click 'Add'.
10. In the 'Key Locator' dialog box, enter the key locator name.
11. Select 'use keystore' and enter the password, path and type from the client key store.
12. Go to 'Key', click 'Add'. Enter the alias as 'server_rsa' and key name as 'CN=Server', 'O=IBM', 'C=US'. Key pass should be empty, because a client does not know the key password of a server key in the client key store.
13. Click 'OK'.
14. Expand 'Key Information' and click 'Add'. Enter the key information name and type as KEYID.
15. Select 'Use Key Locator'.
16. Select key locator and name.
17. Click 'OK'.
18. Expand 'Encryption Information' and click 'Add'.
19. Enter encryption name and key information name.
20. Select key information element and confidentiality part. Click 'OK'.
21. Save and close the descriptor.

Configure the Server to Specify the Confidentiality Part

1. In the Project Explorer, expand the Web service (Converter), webcontent - WEB-INF webservices.xml.
2. Click 'Extension's tab. Expand 'Request Consumer Service Configuration Details'→ 'Required Confidentiality' and click 'Add'.
3. Enter the required confidentiality name. Go to message parts, add parts keyword for username token and body content. Click 'OK'.
4. Click 'Binding Configurations' tab. Expand 'Request Consumer Binding Configuration Details' →'Token Consumer' and click 'Add'.
5. Enter a name.
6. Select the class as 'com.ibm.wsspi.wssecurity.token.X509TokenConsumer'.
7. Select Use Value Type'
8. Select 'X509 certificate' token.
9. Select 'Use jaas.config' and enter the name as 'system.wssecurity.X509BST'.
10. Select 'Use Certificate Path Settings'.
11. Select 'Trust any certificate'. Click 'OK'.
12. Expand 'Key Locators'. Click 'Add'.

13. Enter a name.
14. Select the class as 'com.ibm.wsspi.wssecurity.keyinfo.KeyStoreKeyLocator'.
15. Select 'Use key store' and enter the password, path and type from the server key store.
16. Go to 'Key' and click 'Add' and enter the alias as 'server_rsa', key pass as 'server' and key name as 'CN=Server', 'O=IBM', 'C=US'.
17. Expand 'Key Information' and click 'Add'.
18. enter the key information name and type as KEYID
19. Select 'use key locator'.
20. Select key locator and name.
21. Select 'Use Token' and 'Token'.
22. Click 'OK'.
23. Expand 'Encryption Information' and click 'Add'. Enter the encryption name.
24. Go to 'Encryption Key Information', click 'Add'.
25. Enter key information name and select key information element.
26. Select the required confidentiality part. Click 'OK'.
27. Save and close the descriptor.

Now test the Web Service and client and enable TCP/IP monitor to view the encrypted request SOAP message that is sent from the client to the service.

Confidentiality for the Response Message

To add confidentiality for the response message from the service, configure the Response Generator in the server configuration and the Response Consumer in the client configuration.

For the server configuration, configure the Response Generator Service Configuration Details on the Extensions page and the Response Generator Binding Configuration Details on the Binding Configurations page (refer 'Configure the Client to Specify the Confidentiality Part' section to have a detailed description on the configuration process).

For the client configuration, configure the Response Consumer Configuration on the WS Extension page and the Security Response Consumer Binding Configuration on the WS Binding page (refer 'Configure the Server to Specify the Confidentiality Part' section to have a detailed description on the configuration process).

Integrity

To provide integrity for the client's request message, add a digital signature to the request message. Perform the following process to provide integrity security mechanism for the Web Services.

Configure the Client to Specify the Integrity Part

1. In the Project Explorer, expand the client project (here, 'HelloWebClient') and double click 'Deployment Descriptor: WebClient'. The 'Web Deployment Descriptor' opens.
2. Click 'WS Extension' tab. Expand 'Request Generator Configuration'→'Integrity'. Click 'Add'.
3. enter the integrity name.
4. Enter the order as '2'.
5. Go to message parts, click 'Add'. Go to parts keyword and select security token.
6. Go to message parts, click 'Add'. Go to parts keyword and select body.
7. Click 'OK'.
8. Click 'WS Binding' tab.
9. Expand 'Security Request Generator Binding Configuration'→'Token Generator'. Click 'Add'.
10. Enter the token generator name
11. Select the X509TokenGenerator for the 'Token Generator Class'.
12. Select 'Use Value Type' and then select 'X509 certificate token v3' and the 'X509CallbackHandler'.
13. Select the 'Use Key Store'.
14. Enter the password, path and type from the client key store.
15. Go to 'Key' and click 'Add' '
16. Enter the alias as 'client_rsa' , key pass as 'client' and key name as 'CN=Client', 'O=IBM', 'C=US'.
17. Click 'OK'. The token generator is created.
18. Expand 'Key Locators'. Click 'Add'.
19. Enter a key locator name.
20. Select the class as 'com.ibm.wsspi.wssecurity.keyinfo.KeyStoreKeyLocator'.

21. Select 'Use key store' and enter the password, path and type from the client key store.
22. Go to 'Key' and click 'Add'.
23. Enter the alias as 'client_rsa' , key pass as 'client' and key name as 'CN=Client', 'O=IBM', 'C=US'.
24. Click 'OK'.
25. Expand 'Key Information' and click 'Add'. Enter the key information name and type as STRREF.
26. Select 'use key locator'
27. Select the key locator and name.
28. Select 'Use Token' and then 'Token'.
29. Click 'OK'.
30. Expand 'Signing Information' and click 'Add'.
31. Enter the signing information name, key information name and select the key information element.
32. Click 'OK'.
33. Expand 'Part References' and click 'Add'.
34. Enter the part reference name and select an integrity part.
35. Click 'OK'.
36. Expand 'Transforms' and click 'Add'.
37. Enter a transform name and click 'OK'.
38. Save and close the descriptor.

Configure the Server to Specify the Integrity Part

1. In the Project Explorer, expand the Web Service (here, 'HelloWorld'), webcontent - WEB-INF webservices.xml.
2. Click 'Extensions' tab. Expand 'Request Consumer Service Configuration Details'→'Required Integrity' and click 'Add'.
3. Enter the required integrity name.
4. Go to 'message parts', add parts keyword for security token and body content. Click 'OK'.
5. Click 'Binding Configurations' tab.
6. Expand 'Request Consumer Binding Configuration Details' →'Token Consumer' and click 'Add'.
7. Enter a name.
8. Select the class as 'com.ibm.wsspi.wssecurity.token.X509TokenConsumer'.

9. Select 'Use value type' and 'X509 certificate token v3'.
10. Select 'Use jaas.config' and enter the name as 'system.wssecurity.X509BST'.
11. Select 'Use certificate path settings'.
12. Select 'Trust any certificate'. Click 'OK'.
13. Expand 'Key Locators'. Click 'Add'.
14. Enter a key locator name.
15. Select the class as 'com.ibm.wsspi.wssecurity.keyinfo.X509TokenKeyLocator'.
16. Select 'Use key store' and enter the password, path and type from the server key store.
17. Go to 'Key' and click 'Add' and enter the alias as 'server_rsa' , key pass as 'servert' and key name as 'CN=Server', 'O=IBM', 'C=US'.
18. Click 'OK'.
19. Expand 'Key Information' and click 'Add'. Enter the key information name and type as STRREF
20. Select 'use key locator'
21. Select key locator and name.
22. Check 'Use Token' and select 'Token'.
23. Click 'OK'.
24. Expand 'Signing Information' and click 'Add'.
25. Enter the signing information name, key information name and select key information element.
26. Click 'OK'.
27. Expand 'Part References' and click 'Add'.
28. Enter the part reference name and select an integrity part.
29. Click 'OK'.
30. Expand 'Transforms' and click 'Add'.
31. Enter a transform name and click 'OK'.
32. Save and close the editor.

Now test the Web Service and client and enable TCP/IP monitor to view the XML signature in the request soap message that is sent from the client to the service.

Integrity for the Response Message

To add integrity for the response message from the server, configure the Response Generator in the server configuration and the Response Consumer in the client configuration.

For the server configuration, configure the Response Generator Service Configuration Details in the Extensions page and the Response Generator Binding Configuration Details in the Binding Configurations page (refer 'Configure the Client to Specify the Integrity Part' for a detailed description on the configuration process).

For the client configuration, configure the Response Consumer Configuration in the WS Extension page and the Security Response Consumer Binding Configuration in the WS Binding page (refer 'Configure the Server to Specify the Integrity Part' for a detailed description on the configuration process).

References

1. Web Services Handbook for WebSphere Application Server Version 6.1
<http://www.redbooks.ibm.com/abstracts/sg247257.html>
2. IBM WebSphere Developer Technical Journal: Web services security with WebSphere Application Server V6
http://www.ibm.com/developerworks/websphere/techjournal/0603_cowan/0603_cowan.html
3. IBM WebSphere Developer Technical Journal: Using Web Services Security in WebSphere Application Server
http://www.ibm.com/developerworks/websphere/techjournal/0404_bose/0404_bose.html