Web Services Security Interoperability between SAP NetWeaver 7.0 and Microsoft Windows Communication Foundation (WCF) – Part II

Applies to:

SAP NetWeaver Application Server 7.0 ABAP/Java, SAP NetWeaver Developer Studio 7.0

Summary

This three-part article series takes a deep-dive into a Web Services interoperability scenario that requires secure integration between a Microsoft .NET/WCF-based application consuming a service provided by the SAP NetWeaver platform and vice versa. Starting with a short primer on Web Services Security (WS-Security) and its support in SAP NetWeaver 7.0 and Microsoft .NET 3.0, the first part lays the foundation for the solution developed in the second and third part of this series by providing an understanding of the underlying causes of the interoperability-related issues.

Author: Martin Raepple

Company: SAP AG

Created on: November 6th, 2007

Author Bio

Martin Raepple has practiced as an Information Technology professional for over 12 years and has experience in applying technology in a wide range of industries including telecommunications, financial services, manufacturing and transport. As a Standards Architect with SAP's Industry Standards team, Martin works in the area of standardization and interoperability testing of new Web Services technologies, focusing on security and identity management. Martin is a frequent speaker at SAP TechEd and conferences relating to information security, integration middleware and application development. Martin is the author of the SAP PRESS book "The Developer's Guide to SAP NetWeaver Security".

Table of Contents

Introduction

SAP NetWeaver and Microsoft .NET support advanced Web Services protocols, and this support is the basis of technical interoperability between the two architectures. Both platforms offer interoperability based on existing or upcoming open standards like Simple Object Access Protocol (SOAP), Web Services Reliable Messaging (WS-RM) or Web Services Security (WS-Security).

The first part of this article series [4] introduced Web Services Security and its support in SAP NetWeaver Application Server 7.0 and the Windows Communication Foundation (WCF), a key component in Microsoft .NET 3.0. Part I also examined the interoperability issues of an example scenario that requires advanced security capabilities and proposed solutions to solve them. In this part, all steps to build, deploy and run a WCF-based application consuming a Web Services deployed onto SAP NetWeaver Application Server will be explained in detail.

This series is an update of two related articles that were published in SDN around two years ago on interoperability between SAP NetWeaver 6.40, Microsoft .NET 1.1 and the Web Services Enhancements (WSE) 2.0 SP3. You can find the links to the online references of these articles on SDN at [1] and [2]. The code that accompanies the .NET 1.1/WSE 2.0 scenario can be found at [3].

Scenario overview

The application used to illustrate the Web Services Security scenario between SAP NetWeaver and Microsoft WCF is an online tracking system. A cargo airline (service provider) offers a Web Service interface that allows freight forwarders (service consumers) to track the shipping status of their goods. The cargo airline's security policy does not permit anonymous access to its backend systems. Therefore, any freight forwarder must register itself and provide a valid authentication token to the service provider. In addition, all data transferred to and from the service must be protected against modifications by accident or intent. Confidentiality of the data being transferred is not considered a high risk since it does not contain any personalized or sensitive data.

As shown in Figure 1, both the consumer and the provider possess an asymmetric key pair. They are used to sign the status tracking request and response in order to ensure data integrity. The corresponding X.509 certificates are sent with the messages to authenticate each other. The certificates are issued by a common certificate authority (CA). Consumer and provider trust the identity of the CA's public key. This leads to trust of certificates issued by that CA, because the digital signature on the certificate can be verified using the CA's public key.

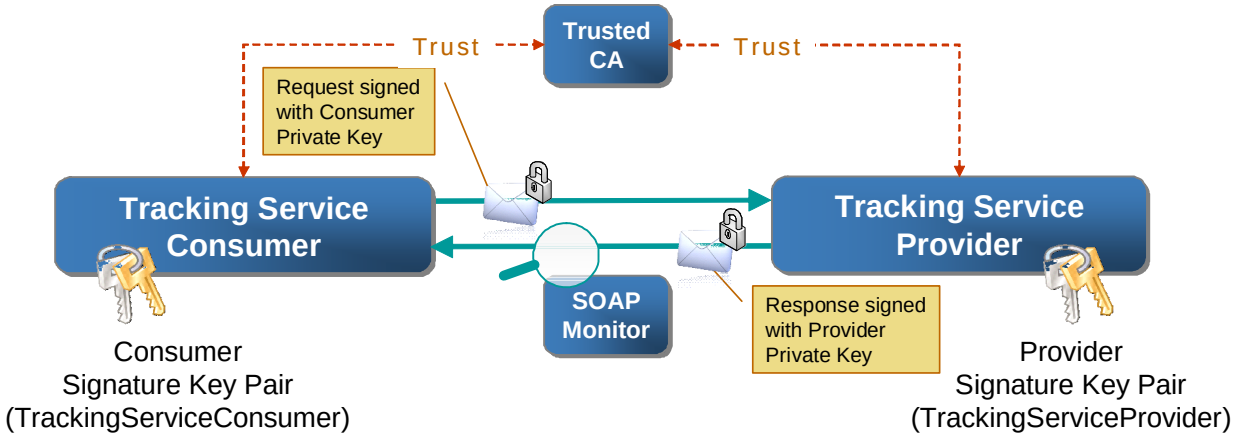

Figure 1 Scenario overview

To visualize the message flow, a network monitor (SOAP Monitor) is used throughout the scenario to capture the messages of the interoperability tests. The scenario will examine both possible variants: Deploying the tracking service consumer onto Microsoft .NET / WCF and the tracking service provider onto SAP NetWeaver (variant #1, discussed in the next section) and vice versa (variant #2, discussed in part 3 [5]).

Scenario Variant #1: Microsoft WCF Consumer and SAP NetWeaver Provider

In this scenario variant, we'll implement a WCF-based consumer that submits signed requests to a service provider deployed on the SAP NetWeaver Application Server.

Installing the keys and certificates

Before any signatures can be created and verified, the consumer's and provider's key pair must be installed in their key stores. In addition, both must possess the public key of the jointly trusted CA. It is out of the scope of this article series to explain the steps of creating the key material and requesting certification from a CA. There is a subfolder */keys* the downloadable code archive of this article series [6] that contains all required keys for the scenario.

Consumer-side installation

Open the Microsoft Management Console (mmc.exe) and select **File • Add/Remove Snap-in ...** . Click on the **Add...** button and choose **Certificates** from the list of Standalone Snap-ins. Again, click the **Add** button and choose **Computer account**. Click on **Next**, choose **Local Computer**, and close the dialog with **Finish**. Click on **Close** in the Add Standalone Snap-in window and **OK** to confirm your changes.

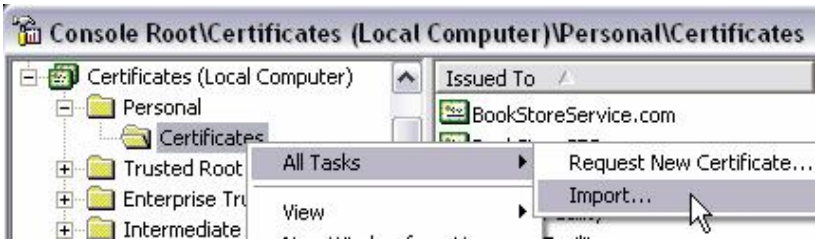

Figure 2 Importing the WCF consumers key pair with the Microsoft Management Console

A new entry (Certificates) is shown in the Management Console. Open the subfolder structure and open the context menu on **Personal • Certificates**. Choose **All Tasks • Import ...** and the Certificate Import Wizard starts (see Figure 2). On the first screen, click on **Next** and choose the *TrackingServiceConsumer.p12* file from the */keys* subfolder in the **File** entry field with the **Browse...** button. On the next screen, enter "secret" for the password and click on **Next**. Confirm the certificate store location (Personal) with **Next** and close the wizard with **Finish**.

Opening the new entry **TrackingServiceConsumer** in the list of personal certificates opens the certificate viewer. The warning message shown in Figure 3 appears because the issuer of the certificate (Trusted CA) is currently not known in the system.

Figure 3 TrackingServiceConsumer certificate

To resolve this issue, the CA's public key must be imported as well. Start the Certificate Import Wizard with **All Tasks • Import ...** again, but this time by opening the context menu on the **Trusted Root Certification Authorities • Certificates** certificate store. Browse for the file *CA.crt* and confirm the import with **Finish**. A new entry with the name "Trusted CA" has been added to the list. Opening the **TrackingServiceConsumer** entry from the **Personal** certificate store once again shows that the trust chain can now be verified by the system.

Finally, the service provider's certificate must be installed in the **Trusted People** certificate store. Open the Import Wizard from the context menu of the store with **All Tasks • Import ...** and choose the file *TrackingServiceProviderCert.crt* from the */keys* subfolder in the code archive.

Provider-side installation

The provider keys and certificates are uploaded to the key store provided by the SAP NetWeaver Application Server Java. Open Visual Administrator and login in as the Administrator of the engine. Select the **Key Storage** service from **<SID> • Server • Services** and open the **WebServicesSecurity** view (see Figure 4). Click on the **Load** button and import the provider's private key with the file *TrackingServiceProvider.p12* from the */keys* subfolder. Next, click on **Load** again to import the corresponding certificate (*TrackingServiceProviderCert.crt*). Two new entries for the provider's signature key pair are now added to the key store view.

To register the consumer's identity as a trusted party, its certificate (*TrackingServiceConsumerCert.crt*) must also be stored in the **WebServicesSecurity** key store view by importing it with **Load**. As for the consumer's environment, the CA public key must also be installed to complete the trust chain when verifying the consumer's signature. The place to store trusted CA certificates is the **TrustedCAs** key storage view. Again, use the **Load** function and choose the *CA.crt* file from the */keys* subfolder to import it.

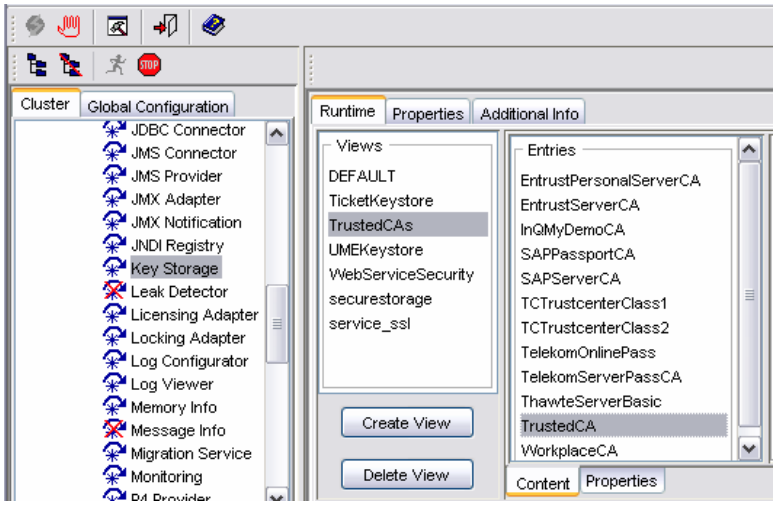

Figure 4 Key Management with the Key Storage service in SAP NetWeaver

Implementing the service provider

The tracking service provider in this scenario variant will be implemented as an Enterprise Java Bean (EJB) that will be exposed as a Web Service on the SAP NetWeaver Application Server Java. Since the focus of this tutorial is on security, the steps to create the EJB and develop the business logic will be skipped and we start with the pre-configured projects. Start SAP NetWeaver Developer Studio (7.0) and select **File • Import**. Choose **Multiple Existing Projects into Workspace** from the list and click on **Next**. Enter the subdirectory */TrackingServiceProviderSAPNetWeaver* from the code archive, select the two projects in the list (see Figure 5), activate the checkbox **Open Projects after import** and click on **Finish**.

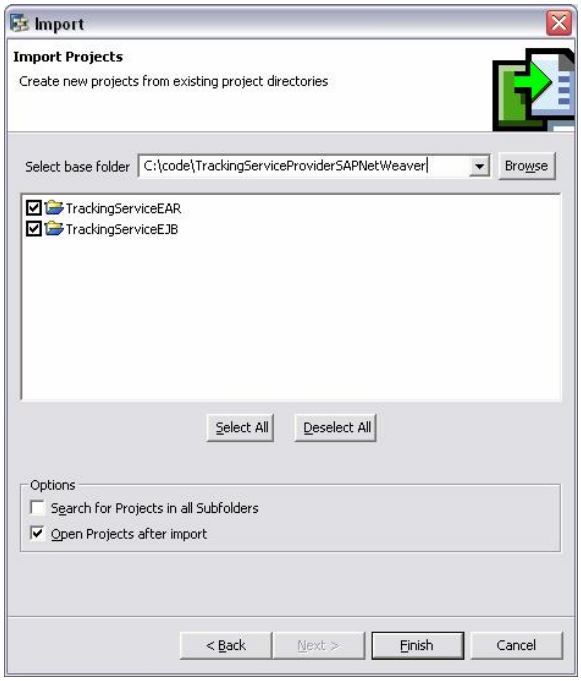

Figure 5 Project import in SAP NetWeaver Developer Studio

Next, start the Web Service Creation Wizard in the J2EE Perspective from the context menu (**New • Web Service ...**) of the **TrackingServiceBean** in the **TrackingServiceEJB** project (see Figure 6) and enter the following data:

- **Web Service Name**: TrackingService

- **Default Configuration Type**: Simple SOAP
- **Configuration Name**: Secure

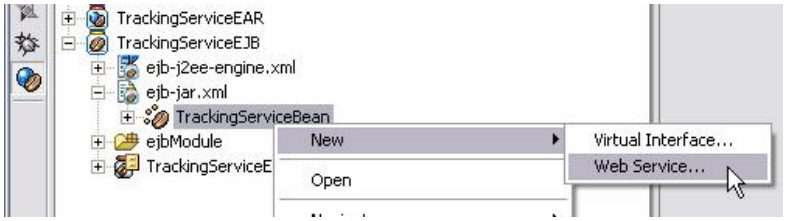

Figure 6 Web Service Creation

Click on **Next** and make sure that method **TrackStatus** is selected. Click **Next** again and choose the **TrackingServiceEAR** project in **EAR Project**. By completing the wizard with **Finish**, all Web Service artifacts (Virtual Interface, Web Service Configuration & Definition) are generated.

The **WS Deployment Descriptor Editor** opens by default. Select **TrackingService • Secure • Security** from the Web Service Configurations and pick **Document Authentication** from the **Authentication Mechanism** dropdown box. Since consumers must provide a valid signature and authenticate with their certificates, choose **X.509 Certificate** and select **Signature** in the **Request** and **Response** field of the **Document Security** tab (see Figure 7).

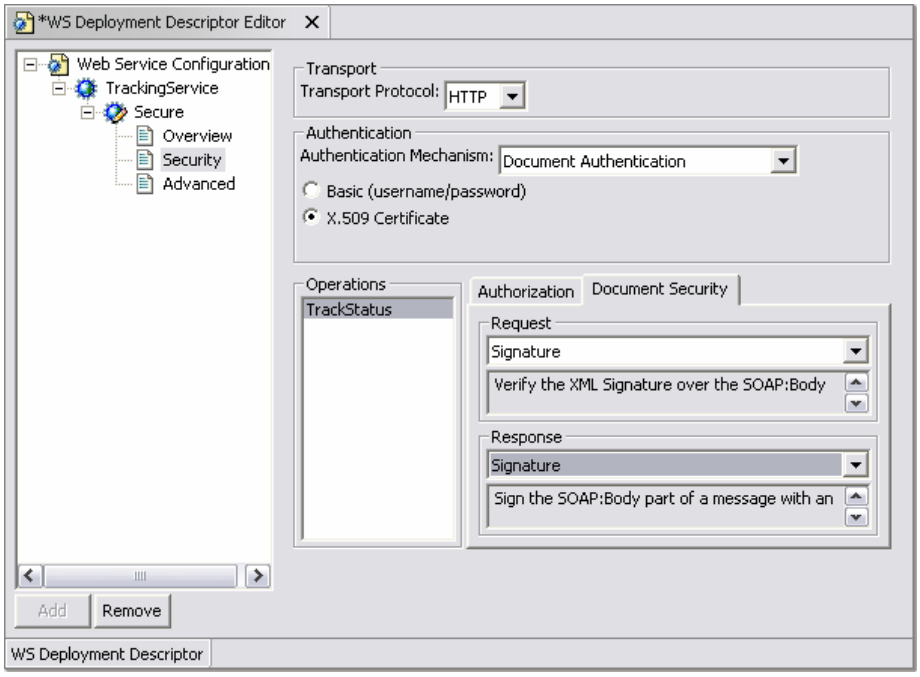

Figure 7 Web Services Security settings in SAP NetWeaver Developer Studio

Save your changes, rebuild the EJB archive from the context menu of the **TrackingServiceEJB** project (**Build EJB Archive**), create the EAR file (**TrackingServiceEAR** project context menu, **Build Application Archive**) and deploy it to the Application Server (right-click on the new *TrackingServiceEAR.ear* file, **Deploy to J2EE Engine**).

WS-Security runtime configuration in Visual Administrator

After successful completion of the deployment process, open the Web Services Security runtime configuration of the new Web Service instance in Visual Administrator (**<SID> • Server • Services • Web Services Security**). Switch to the **Profile Administration** tab and select the **Outbound Messages** list. Create a new outbound Security Profile with **New...** and enter the name “TrackingServiceOutbound”. Choose **Signature** in the **Template** dropdown box and make the following selection in the configuration:

- Body Signature: Keystore View: **WebServiceSecurity**
- Body Signature: Keystore Alias: **TrackingServiceProvider**
- WS Security Version: **WS Security SAP NW 04 Stack 04**

Save the new profile and switch back to the **Security Administration** tab. Open the **TrackingService*Secure** configuration in **Security Configuration • Web Services • sap.com • TrackingServiceEAR** and change the setting of the **Outbound Profile** for the TrackStatus operation from **None** to **TrackingServiceOutbound** (see Figure 8).

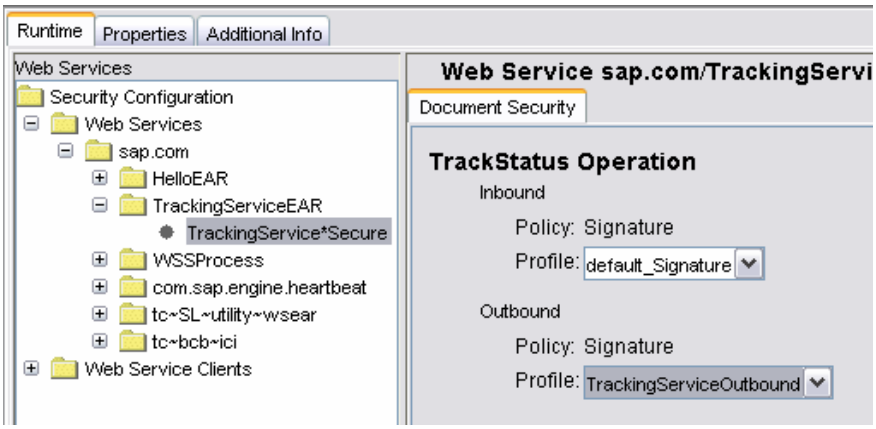

Figure 8 Web Services Security configuration of the TrackingService provider

User mapping

After saving the changes, the service provider will only accept requests that are signed by trusted consumers. However, successful validation of the request also requires a mapping of the consumer's certificate to an existing user account in the Application Server. This can be defined in the **Security Provider** service. In the **Runtime • User Management** tab, click on the pencil icon to change into editing mode. This allows adding a new user account with the **Create User** button. Enter an arbitrary username and password (e.g. "FreightForwarder", "abc123"), and assign the **TrackingServiceConsumerCert** from the WebServiceSecurity view (see Figure 9).

Figure 9 Assigning a certificate to a new account

Implementing the service consumer

The service consumer will be implemented using Visual Studio 2005. As already mentioned in part I [4], developing applications with WCF requires additional components to be installed on Windows XP and Windows Server 2003 SP1.

Start Visual Studio and create a new solution with **File • New • Project**. Select **Windows** from the Project types list and **Console Application** from the installed templates. Choose an appropriate **location** for your new solution and assign "TrackingServiceConsumer" to the project **name** and "TrackingServiceSolution" to the **solution name** (see Figure 10)

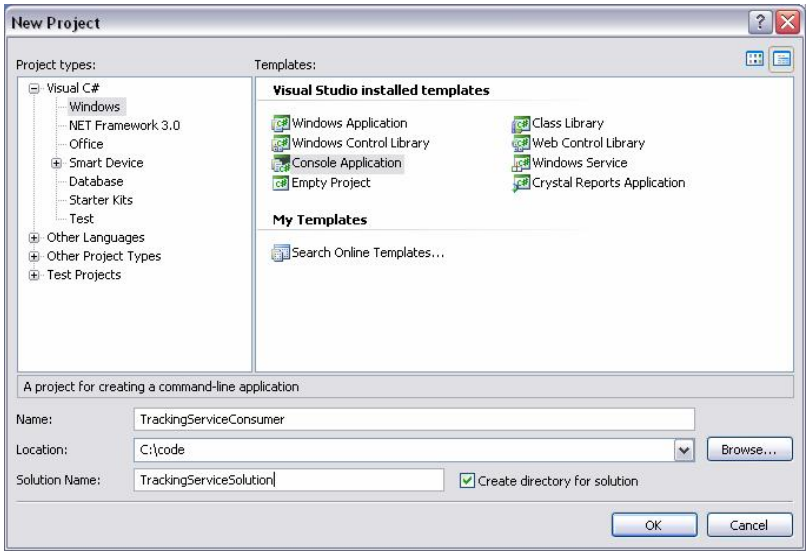

Figure 10 Creating a new solution for the consumer application in Visual Studio 2005

To generate the WCF proxy classes for the tracking service, open the context menu on the **References** folder in the new **TrackingServiceConsumer** project and choose **Add Service Reference...** . In the dialog box, enter the URL of the tracking service WSDL file in **Service URI** and "TrackingService" as the **Service reference name**. The generic format of the WSDL URL in SAP NetWeaver Application Server Java is `http://<host>:<port>/TrackingService/Secure?wsdl&style=document`. Replace `<host>` and `<port>` according to your configuration.

In addition to the proxy class (*TrackingService.cs*), a new WCF application configuration file (*app.config*) has been generated. When you open this file, you'll notice that the system-provided `BasicHttpBinding` (`<basicHttpBinding>`, see [4]) is used with security turned off by default (`<security mode="None">`).

According to Listing 7 in part I [4], replace the `<security>` element (lines 14 to 18) with the following configuration (see Listing 1):

```
<security mode="Message">
  <message clientCredentialType="Certificate"
           algorithmSuite="Basic128Rsa15"/>
</security>
```

Listing 1 BasicHttpBinding security configuration for the WCF consumer proxy

The certificate to be used by the proxy at runtime to sign the message is configured in a separate element in *app.config*, called a Behavior. In general, Behaviors modify or extend service provider or consumer functionality. For a WCF consumer proxy, an `<endpointBehavior>` configuration section represents all the behaviors defined for a specific endpoint. This includes the credentials used to authenticate the consumer to a provider (`<clientCertificate>`) as well as specifying a default certificate (`<defaultCertificate>`) that the provider uses to authenticate to the consumer (`<serviceCertificate>`). Listing 2 shows the behavior configuration for the tracking service proxy that should be appended after the closing `</client>` element:

```
...
</client>
<behaviors>
  <endpointBehaviors>
```



```
      <behavior name="proxyCredentials">
        <clientCredentials>
          <clientCertificate findValue="TrackingServiceConsumer"
                             storeLocation="LocalMachine"
                             storeName="My" x509FindType="FindBySubjectName"/>
          <serviceCertificate>
            <authentication certificateValidationMode="PeerOrChainTrust"
                            revocationMode="NoCheck" />
            <defaultCertificate findValue="TrackingServiceProvider"
             storeLocation="LocalMachine" storeName="TrustedPeople"
             x509FindType="FindBySubjectName" />
          </serviceCertificate>
        </clientCredentials>
      </behavior>
    </endpointBehaviors>
  </behaviors>
  ...
```

Listing 2 EndpointBehavior of the tracking service proxy

To activate the new behavior configuration, its name ("proxyCredentials") must be referenced in the endpoint (`<endpoint>`) declaration using the `behaviorConfiguration` attribute. Since WCF expects the DNS name in the URL of the `address` attribute (e.g. 127.0.0.1) to be equal with the name in the X.509 certificate (e.g. TrackingServiceProvider), validation of the provider's identity in the response would fail. This problem can be fixed by explicitly specifying the identity 'TrackingServiceProvider' as an Identity property (`<identity>`) of the endpoint configuration (`<dns value="TrackingServiceProvider">`).

The complete `<client>` element configuration is shown in Listing 3:

```
  <client>
    <endpoint address="http://127.0.0.1:53000/TrackingService/
      Secure?style=document" binding="basicHttpBinding"
      bindingConfiguration="SecureBinding"
      contract="TrackingServiceConsumer.TrackingService.
      TrackingServiceVi_Document" name="SecurePort_Document"
      behaviorConfiguration="proxyCredentials">
      <identity>
        <dns value="TrackingServiceProvider"/>
      </identity>
    </endpoint>
  </client>
```

Listing 3 Service endpoint configuration of the tracking service proxy

In the final step, we'll use the generated class to create an instance of the proxy and send a request to the tracking service. Rename the auto-generated *Program.cs* file to *TrackingServiceConsumerApp.cs*, and open it in the C# editor. Adding the code from Listing 4 to the `Main` method will create a new proxy instance (`proxy`) based on the endpoint configuration with the name `SecurePort_Document` (see Listing 3) from the *app.config* file. Once the proxy object is created, the `BasicHttpBinding` default protection level (`EncryptAndSign`, see also [4]) can be throttled by assigning the `ProtectionLevel.Sign` value to the `proxy.Endpoint.Contract.ProtectionLevel` property to match the inbound profile of the `TrackStatus` operation which uses the `Signature` Security Template to validate the requests.


```
Console.WriteLine("Starting client");

TrackingService.TrackingServiceVi_DocumentClient proxy = new
  TrackingService.TrackingServiceVi_DocumentClient("SecurePort_Document");

proxy.Endpoint.Contract.ProtectionLevel =
  System.Net.Security.ProtectionLevel.Sign;

String result = proxy.TrackStatus("12345");

Console.WriteLine("Current status: " + result);

Console.WriteLine("Press return to exit");

Console.ReadLine();
```

Listing 4 C# code to create the proxy and send a signed request to the service provider

Testing the scenario

To capture the messages that are being sent to and from the tracking service consumer and provider, a SOAP message monitor based on the widely used Apache TCPMon is used. The tool can be downloaded from `http://ws.apache.org/commons/tcpmon/` and started with `tcpmon.bat` from the */bin* subfolder. TCPMon comes up with the Admin dialog where one can start new message monitors, each listening on a distinct TCP port. Pick a **Listen Port** that is not in use by any other application (e.g. 8888) and configure the new monitor to act as a listener, forwarding any requests/responses to the **Target Hostname** and **Target Port #** of the SAP NetWeaver Application Server instance where the tracking service instance is running (e.g. localhost, 53000). After clicking on the **Add** button, the new monitor is waiting for messages. It is recommended activating the **XML Format** checkbox at the bottom of the monitor dialog to increase readability of the traces.

In order to use the new monitor, the address attribute of the endpoint configuration in the WCF consumer's application file (*app.config*) must be changed according to the new listen port (e.g. `<endpoint address="http://127.0.0.1:`**`8888`**`/TrackingService/...`). After the configuration change, start the client with F5 in Visual Studio and make sure that the monitor is running. It will capture the request and response as shown in Figure 11.

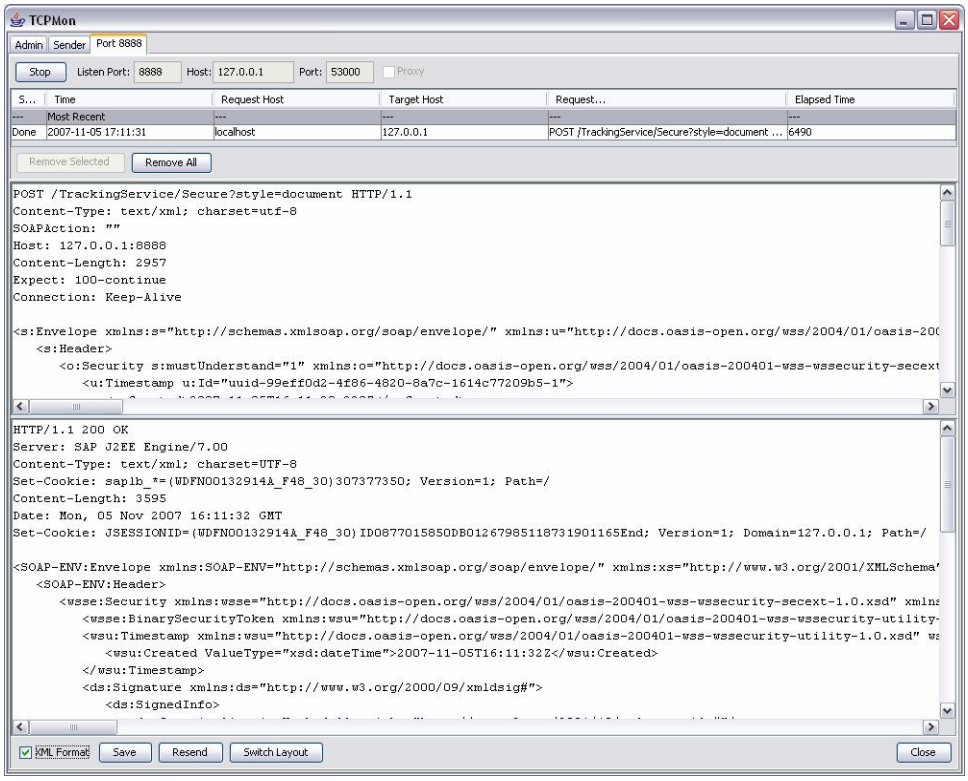

Figure 11 Captured request and response messages with TCPMon

Conclusion and outlook

Part II provided a step-by-step tutorial to build a WCF client for consuming a Web Services on the SAP NetWeaver Application Server. Message integrity and mutual authentication of the partners is achieved by using the widely adopted WS-Security standard. In the last part [5] of this series, the scenario will be implemented the other way round: A deployable proxy on the SAP NetWeaver platform will submit a signed request to a WCF service and process the signed response.

Further Information

For more information and examples how to implement secure Web Services-based scenarios between SAP NetWeaver and other platforms (e.g. NET/WCF, Apache Axis2/Rampart), a new SAP PRESS book "The Developer's Guide to SAP NetWeaver Security" provides a practical guide for developers, system integrators, and software architects. The German version ("Programmierhandbuch SAP NetWeaver Sicherheit", ISBN 978-3-8362-1019-5) is available now – the English translation will follow soon. The book covers all security technologies in conjunction with SAP NetWeaver Application Server up to and including Release 7.0. In addition to describing the basic principles of the different technologies (Web Services Security, Single Sign-On, SAML, SPML etc.), the book focuses on providing practical exercises and examples that help you gain a profound understanding of the standards used, enabling you to better assess their intended purpose. The book serves as an A-to-Z reference book that enables you to use - and benefit from - open security standards that are based on an enterprise service-oriented architecture (Enterprise SOA).

Related Content

[1] Web Services Security Interoperability with SAP NetWeaver 6.40 and Microsoft .NET 1.1 (Part I), SDN, Sept. 2005

[2] Web Services Security Interoperability with SAP NetWeaver 6.40 and Microsoft .NET 1.1 (Part II), SDN, Sept. 2005

[3] Downloadable code sample for the article series on Web Services Security Interoperability with SAP NetWeaver 6.40 and Microsoft .NET 1.1, SDN, Sept. 2005

[4] Web Services Security Interoperability between SAP NetWeaver 7.0 and Microsoft Windows Communication Foundation (WCF) – Part I, SDN, November 2007

[5] Web Services Security Interoperability between SAP NetWeaver 7.0 and Microsoft Windows Communication Foundation (WCF) – Part III, SDN, November 2007

[6] Downloadable code sample for the article series on Web Services Security Interoperability between SAP NetWeaver 7.0 and Microsoft Windows Communication Foundation (WCF), SDN, November 2007

Copyright