Web Services with SAP Part 2 – Web Service Creation

Applies to:
SAP Web Application Server (WAS) 6.40, 7.0 and CE (7.1).

Summary
The article presented here will take you through the various Web Service creation methods available with WAS on Java, ABAP and PI.

Author: Alka Panday
Company: Larsen & Toubro Infotech Ltd.
Created on: 08 December 2008

Author Bio
Alka Panday works at Larsen & Toubro Infotech Ltd., as a SAP Netweaver Consultant. She started with Web Dynpro for Java, and now covers J2EE (EJBs), Web Services, Web Dynpro (ABAP), SOA and Web Services, with special interests in PI.
**Table of Contents**

Web Service Creation.........................................................................................................................................3

EJB Methods Exposed as Web Service .........................................................................................................3
  Virtual Interfaces...........................................................................................................................................3
  Web Service Definition.................................................................................................................................4
  Web Service Configuration ...........................................................................................................................4

EJB Methods exposed as Web Service with CE and EJB 3.0.................................................................4

PI Interfaces exposed as Web Service ...........................................................................................................5
  Technical Procedure......................................................................................................................................5

ABAP Web Services .......................................................................................................................................6
  Technical Procedure......................................................................................................................................6

Enterprise Service Repository and Service Registry.................................................................................8
  Using Service Registry (SR) .........................................................................................................................9

Related Content...............................................................................................................................................10

Disclaimer and Liability Notice...................................................................................................................11
Web Service Creation
The Web Service aware SAP Web Application Server (WAS) offers several ways to create Web Services. The SAP NetWeaver established and common methods that are adapted by most developers are discussed in this article.

Before going through the below listed methods it is advisable to check the basic concepts related to Web Services from SDN and also the Web Services with SAP – Part 1 – Concepts paper.

EJB Methods Exposed as Web Service
This is a purely SAP Web AS Java based technique of creating Web Services which is simple and fast. The Enterprise Java Bean (EJB) framework provides a common deployment strategy for the Web services. The EJBs that are used on the WAS are Session Beans (Stateless, Stateful), Entity Beans and Message Driven Beans. Web Services can only be created for the business functions / methods of Session Beans.

As EJB project is created with a Session bean, the procedure of Creating the Web Service, involves a series of simple steps, viz., start a new Web Service wizard, fill in the required information like name, package and select appropriate security mechanism. Next is the selection of the business methods that are to be exposed as Web Service Operations and finally, select the Enterprise Archive of the EJB project for Web Service deployment.

Note: For a business method to be exposed as Web Service, make sure that, all the parameters of the method are serializable. Also, classes from java.sql.* package are not supported as Web Service parameters.

The following sections will give an insight of the different Web Service files that are created in the EJB project with the above procedure, that deal with various parameters relevant to the Web Service security handling. The sections below are relevant to SAP NetWeaver 2004s and lower, supported by SAP WAS 7.0 and lower.

Virtual Interfaces
A Virtual Interface (VI) is a visual representation of the Web Service to its consumers. Using the VI, one can control different features of the Web Service. One can,

- Hide operations and parameters: ‘Exposed’ Check Box
- Replace names for operations and parameters with meaningful names: Using New Name field.
- Set fixed values for hidden parameters: Using Initial Value and Fixed Value fields.
- Convert parameter data types: Using New Data Type Name field
- Define parameter representation in the SOAP message
- Maintain SOAP Extensions for the Web service operation arguments
Web Service Definition

Web Service Definition (WSD) is a place to define the features of the Web Service in an abstract form. Different features related to the communication type, authentication level, operation level authorizations, etc. The features are abstract in the sense that they are defined in the WSD and then the technical details are added in the Web service Configuration (WSC).

The various controllable options available in the WSD are:

- **Session**: To mark Stateful communication.
- **Authentication**: Authentication level for Web Service access.
- **Authorization**: Web Service operation level authorizations to be added.
- **Transport Guarantee**: To configure the communication as SSL encrypted between Web Service and clients.
- **Message ID**: Additional SOAP Header information to be added to uniquely identify Web service messages.

Web Service Configuration

The Web Service Configuration (WSC) is used to define the runtime properties of the Web service. A Web Service can have more than one WSC. Most of the content that is rendered by the WSC is derived from the underlying EJB and the WSD. The different modifiable sections of a WSC are:

- **Overview**: Add or modify the Web Service URL suffix, using Path field.
- **Security**: Select the WSD compatible authentication type for the Web Service.
- **Advanced**: Add WSDL Port and Binding Name, and the address of target server.

EJB Methods exposed as Web Service with CE and EJB 3.0

With the advent of CE and EJB 3.0 in WAS, the above procedure has come down to a single step implementation for creating Web services with Stateless Session beans. In CE i.e. WAS 7.1 and above, you just need to add the Web Service annotation tag to the Session Bean class to expose it as a Web service.

So, for CE and EJB 3.0, add the following tag to the Session Bean class:

```java
@WebService(serviceName="MyServiceName", name="MyServiceClass", portName="MyLogicalPortName")
```

and you are done with the Web service creation.

For operations that could be performed with the use of VI and WSC with WAS 7.0 can now been done with a series of annotation tags. Some of them are explained as following:

```java
@WebMethod(operationName="myOperation")
Used for all public methods in the EJB those are needed to be exposed as Web Service Operations.

@AuthenticationDT(authenticationLevel = AuthenticationEnumsAuthenticationLevel.BASIC)
It is a Class level tag, to identify the authentication mechanism for the Web Service.

@WebParam(name="myParam")
Used for the argument to provide a meaningful name to the Web Service operation parameters
```
The following table gives the comparison of the WSDL tags with an EJB method:

<table>
<thead>
<tr>
<th>WSDL Tags</th>
<th>EJB Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>wsdl:service</td>
<td>The enterprise bean.</td>
</tr>
<tr>
<td>wsdl:bindings</td>
<td>Similar to the Bean Class, with all exposed methods, their input and output parameter structures.</td>
</tr>
<tr>
<td>wsdl:porttype</td>
<td>EJB Class Name with all its methods and parameters definition. It is closer to the Remote Interface of an EJB.</td>
</tr>
<tr>
<td>wsdl:message</td>
<td>Representation for all Method parameters under single name. Similarly, common name for the Return Parameter.</td>
</tr>
<tr>
<td>wsdl:types</td>
<td>Details of individual EJB method parameters and return parameter</td>
</tr>
</tbody>
</table>

**PI Interfaces exposed as Web Service**

SAP NetWeaver’s Process Integration (PI), formerly known as Exchange Infrastructure (XI), is an Integration Technology, which facilitates the communication of various SAP and Non-SAP systems. The communication is based on the standard XML message formats.

In PI, one can define a **Message Interface (PI 7.0)**, which is a representation of the message type, direction (Inbound, Outbound, Abstract) and mode (Synchronous & Asynchronous) of communication for PI, to communicate with senders or receivers.

With advent of PI 7.1, Message Interface is termed as Service Interface with a more liberal definition as **Service Interface enables you to describe – independently of a platform or programming language – operations that you require later for an implementation in the application system at a later stage.** Depending on the category of a service interface, the following use cases are possible:

- **Inbound (Provider Role):** You want to implement a service in an application system, which can be called by a user.
- **Outbound (Consumer Role):** You want to call a service of a provider. To do so, you require the outbound service interface that matches your inbound service interface.
- **Abstract:** In enhanced communication using the Integration Server, you want to exchange messages with a buffered integration process

A Message Interface or Service Interface can be compared to a java class in several ways with regards to the Web Services concept. For example, the Complex type Method arguments of a Java class are equivalent to the Message Type of a Message Interface, also the methods in java class, which can be Web Service operations are similar to the default operation of the Message Interface or several operations supported by the Service Interface. Plus, the mode of the Message interface indicates the type of the java class method, which means, a synchronous interface indicates a java class method with a return type, or a synchronous web service operation.

**Technical Procedure**

All the Message Interfaces, now Service Interfaces with SAP Process Integration (PI) 7.1, viz., Inbound, Outbound and Abstract defined in the Integration Repository have the capability to be exposed as Web Services.

The WSDL of such Web services is available in the WSDL tab of the specified Service Interface.

The accurate WSDL, i.e. with the Web Service URL, of Web Services exposed from interfaces can also be derived from the Integration Directory. Choose Tools → Display WSDL, and in the appearing wizard, the details to be entered are:

- **URL of the Host Server:** URL including the host name and server HTTP port.
- **Message Interface Name, Name space and Software Component Version**
- **Sender Service, Interface name and Namespace**

It is important to note that, the URL of the Web service available in the soap:address tag of this WSDL is formed from the values entered by the user.
With SAP PI 7.1, concept of Enterprise Service Repository (ESR) has been introduced, which can also be used to model and generate Web Services. The clear advantage of using ESR is the availability of modeling technology and PI Web Services with multiple operations.

**Note:**
1. The Web Services that are exposed from the PI Interfaces, with XI / PI 3.0 and above, are by default defined with Basic Authentication for access. To access these Web Services one is required to have an appropriate user access to the PI Server.
2. Also, the Display WSDL tool provided in PI is not built with the functionality to check the validity of the parameters entered by the user.

The following table gives the comparison of the WSDL tags with the PI Message Interface:

<table>
<thead>
<tr>
<th>WSDL Tags</th>
<th>PI Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>wsdl:service</td>
<td>The PI Message / Service Interface</td>
</tr>
<tr>
<td>wsdl:bindings</td>
<td>The PI Service Interface with all its operations, request and response</td>
</tr>
<tr>
<td>wsdl:porttype</td>
<td>Message types.</td>
</tr>
<tr>
<td>wsdl:message</td>
<td>The Message type of the interface Request, Response and Fault.</td>
</tr>
<tr>
<td>wsdl:types</td>
<td>The Data types involved in the Message interface request and / or response.</td>
</tr>
</tbody>
</table>

### ABAP Web Services

Users can access the available functions, such as an RFC-enabled function module, function group, and BAPI, by creating Web Services for them, even for those stored in a different system. The Service Wizard technique elaborated here for exposing ABAP components as Web Service has been enabled with ECC 6.0.

#### Technical Procedure

The Service Wizard enables the user to expose the remote enabled function modules & groups, and BAPI as Web Services in few simple steps. It is accessible by the Transaction SE80 for different types of ABAP Objects or SE37 for function groups and modules.

In SE80, the Service Wizard is started with:
- Choose the name of the package
- From the context menu, select Create → Enterprise Service and
- Service Provider → Existing ABAP Objects (Inside Out)

With SE37, it is done as:
- Choose the function module or group and display it
- Select Utilities → More Utilities → Create Web Service →
  From the Function Module or From the Function Group

The different steps in the wizard involved are: Give an appropriate Name and End point type, and then select the object to be exposed as Web Service and for BAPI and Function Groups select the operations. An important step is to Configure a Service, i.e. Assign Web Service features for type of communication and security of the data transferred. For example, PRF_DT_IF_SEC_LOW option defines that the communication will be secured with basic username and password authentication and there is no transport guarantee.

Once the Web Service is created using the Service Wizard, the Service Interface is generated. In the interface of the Web service, you can rename or hide operations (for example, methods) and parameters. You can define standard values for parameters and convert parameter types.
The following table gives the comparison of the WSDL tags with a Function Module of ABAP:

<table>
<thead>
<tr>
<th>WSDL Tags</th>
<th>ABAP Function Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>wsdl:service</td>
<td>Function Module Service Name and URL Address</td>
</tr>
<tr>
<td>wsdl:bindings</td>
<td>Association of Service and Function Module names and listed operations</td>
</tr>
<tr>
<td>wsdl:porttype</td>
<td>Details of the Function Module including input and Outout Message Types</td>
</tr>
<tr>
<td>wsdl:message</td>
<td>Complex Type Details which represent the function module input and output. The “Table” parameters are included in both, input and output Complex Types.</td>
</tr>
<tr>
<td>wsdl:types</td>
<td>The definition of all the Data elements and types involved in the Import, Export and Table Parameters.</td>
</tr>
</tbody>
</table>

With this, we have seen the basic and most common ways to create a Web Service on the SAP Web AS.

**Points to Remember:**
1. The messages names, i.e. operation names, should not be very long. Try to keep them within 20 characters for Java based Web Services.
2. Limit the use of underscore ( _ ) in the message, operation and parameters names. In ABAP Web Service always select Name Mapping check box, to eliminate underscores in the WSDL.

**Note:** If the above are overlooked, the Web Service may face errors related to XML Deserialization while consumption.
**Enterprise Service Repository and Service Registry**

The concept of Enterprise Service Repository (ESR) and Service Registry is introduced with SAP Web AS version 7.1. ESR and Registry combine the advantages of language-independent modeling and specification of applications in the *Enterprise Services Repository* with the option of publishing application services in the *Services Registry* and to classify them. Both the services, hence, contribute greatly to the development of applications according to the ESOA.

![Diagram of Enterprise Service Repository Functions]

Fig 1: Enterprise Service Repository Functions

ESR on the SAP Web AS, which can be accessed via the Enterprise Services Builder, is used to model and specify service interface objects that can be implemented later. It provides a modeling environment for model-driven approach of application development. It also facilitates the user with WSDL based editors that aid in designing of service interfaces. The Service Interface, which is an upgrade over its PI 7.0 counterpart Message Interface, can now be designed to hold multiple operations or message types, taking it closer to the Web Service paradigm.

Service Registry is a directory of Web Services which conforms to the UDDI 3.0 specification. Located centrally within SAP Web AS, it contains entries for all services and service definitions in its landscape, with references to the service relevant WSDL metadata and to the locations of the callable service endpoints. The registered services are classified using semantic-rich classification systems to enable browsing of services by classification. Users like Developers, Administrators, Non-SAP Application users, all can find a use to the Service Registry to find Web Services for use or Web Service Endpoints for managing connection between consumer and provider.
Using Service Registry (SR)

The general usage scenario for a Service Registry can be well depicted with the lifecycle of a client application with the Service Registry.

![Diagram: Lifecycle of client application with SR](image)

The high level process flow can be explained as:

1. The consumer queries the SR to find the services to use.
2. The Service Registry delivers the references to the metadata.
3. The consumer / consumption tool retrieves the metadata using this reference from the destinations.
Related Content

Utilizing Web Services with SAP Web Application Server
Providing Web Services in Java
Deep Dive into the Enterprise Services Repository
Consuming Services with the Services Registry
Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.