Appplies 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 SAP Web AS. It also gives a brief explanation of the emerging RIA technologies like Adobe Flex for consuming Web Services.

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

Consumption of Web Services ........................................................................................................................... 3
  Java based Methods ....................................................................................................................................... 3
    Proxy Client .................................................................................................................................................... 3
    Web Dynpro .................................................................................................................................................. 3
  ABAP based Methods .................................................................................................................................. 4
    ABAP Proxy .................................................................................................................................................... 4
    Web Dynpro for ABAP (WD4A) ..................................................................................................................... 4
  Additional ..................................................................................................................................................... 4
    Visual Composer ............................................................................................................................................ 4
    CAF Core and CAF GP ............................................................................................................................... 4
    PI – SOAP Adapter ..................................................................................................................................... 5
    Adobe Flex .................................................................................................................................................... 5
  Related Content .......................................................................................................................................... 6
  Disclaimer and Liability Notice ...................................................................................................................... 7
**Consumption of Web Services**

As there are several Web Service creation methods that are supported by SAP Web AS, which are explained in the [Web Services with SAP – Part 2 – Web Service Creation](#) article on SDN, there are several methods that can be employed to access / consume the Web Services available. This section gives information about the most common methods that are used for accessing exposed Web Services.

**Java based Methods**

**Proxy Client**

A Proxy is an executable interface that is generated as a result of converting the non-language-specific interface descriptions in WSDL.

The Java Web Services proxies that are created can be of two types: Deployable and Standalone. The difference in both the proxy types are in the fact that, Deployable Proxy is a client that must be deployed on the J2EE Engine as an application, while, Standalone Proxy is a client that generates stubs and runs without the J2EE Engine. Thus it can be used only with the release for which it has been generated.

For the standalone proxy, a stub must be generated and the names and class names of the transport bindings, protocols, and transports that are used must be provided. The drawbacks of this approach are that if user, address of the endpoint component is changed or requires some modifications, the stub is no longer valid and the whole proxy needs to be regenerated. On the other hand, with the deployable proxy all information is either generated during deployment or is retrieved at runtime. Therefore, the deployable proxies are to a certain degree protected from runtime changes.

The creation mechanism and procedure to consume the Web Service are quite similar in both the proxy types, where the only prerequisite is the WSDL file, which can be selected from the WAS, UDDI, URL or Local File system. With WAS 7.0 the Web Service Proxy is created as a separate project, with CE the clients can be generated in the EJB Module projects.

**Web Dynpro**

Web Dynpro is a SAP’s Standard toolset for UI Development and Execution. It is based on the Model-View-Controller concept of development of user interface, where Model is used for retrieving data as well as integrating the business logic, View represents the graphic display of data and Controller is responsible for the program control. A Web Service can be invoked by the Web Dynpro Application by executing the Web Service based Model.

Adaptive Web Service model can be created in Web Dynpro to consume the Web Service’s WSDL from the WAS, UDDI, URL or Local File system. In such a model, the WSDL is interpreted to generate the Model Classes representing the Web Service and its Complex Types and Dictionary Objects representing the simple types which form up the Web Service Operation parameters. For each operation in the Web Service Model Nodes are generated which denote the Request and / or Response of the Web Service. A method called “execute ( )” is provide to such model nodes to execute the operation. This method is called by the Service Controller, which can be a Custom Controller or the default Component Controller of the Web Dynpro Application. With deployment of Adaptive Web Service model, the Logical Destinations are created on the WAS which can be configured according to the Web Service endpoint configurations.
ABAP based Methods

ABAP Proxy

As one can create Java Proxy for a Web Service, proxies based on ABAP programming can also be generated. The most widely used ABAP Proxy generation method is the consumption of PI Message Interfaces from the Integration Repository. Transaction Code SPROXY is used for this purpose.

The Integration Repository contents displayed in SPROXY are of the Integration Server defined using SXMB_ADM on the R/3 Server. The repository to be accessed can also be defined in the Exchange Profile or with a HTTP External Destination (G type).

The proxy generation process retrieves the WSDL description of the interface from the Integration Repository using HTTP. The address, user and password of the corresponding server are obtained using the same.

Another way of creating ABAP Proxy or updating the generated proxies is through Object Navigator Transaction SE80. The option is Create → Enterprise Service/Web Service → Proxy Object. An important point using this method is that the source of the WSDL can be a file, URL or Integration Repository.

Web Dynpro for ABAP (WD4A)

Web Dynpro was initially only available with its Java implementation, but with Web AS 7.0 or EEC 6.0 and higher it is also available in ABAP. The implementation technique and method remain the same in this ABAP counterpart of Web Dynpro for Java.

Additional

With SAP NetWeaver stack, there are many other components that provide the basic functionality of Web Services access and use. The functionality provided by the following discussed components may not cover all the aspects of Web Service consumption but are mature enough to support the common configurations.

Visual Composer

SAP NetWeaver’s Visual Composer (VC) provides a development environment for rapidly creating and adapting model-based transactional and analytical applications. It has been designed for the business analysts who can create enterprise applications using standardized components that meet SAP standards and quality criteria. VC operates on top of the SAP NetWeaver portal, utilizing the portal's connector-framework interfaces to enable access to a range of data services, including SAP and third-party enterprise systems.

The prerequisite to consume a Web Service is the presence of the Web Service System, i.e. the Web Service provider system need to be defined in VC. This is done by using the Tools → Define Web Service System menu, which requires the Web Service WSDL URL as mandatory parameter. Once the Web Service system is defined, then one can use this System to search all the Web Service operations in the Find Data Services view and then drag and drop the Web Service operation as a Data Service for use.

CAF Core and CAF GP

Composite applications are built and deployed on top of data and functions provided as services by platforms and applications, combining these into user-centric processes and views, supported by their own business logic and specific user interfaces. Composite applications are loosely coupled to the backend systems on which they are based, resulting in a new logical application tier which can be deployed and upgraded independently of the backend infrastructure.

The SAP Composite Application Framework (SAP CAF) provides a set of tools and environment for creating and running composite applications. CAF Guided Procedures (CAF GP) is a framework for modeling and managing workflows using simple and user-friendly tools. As a part of the Composite Application Framework (CAF), GP enables access to multiple backend systems, consistently combining different types of services and applications into processes.
To consume a Web Service with CAF, you need to define an External Service of type Web Service, and then the WSDL selection can be done from WAS, UDDI, URL or Local File system. Once this is done, the Web service operations are ready to be accessed.

With CAF GP, to access and invoke Web Services, we need to define them as Callable Objects (which can be used in a process). We first create a callable object with type as Services and filling out other all basic information like name, package, etc., then while defining the object, one can choose to enter the WSDL URL of the Web Service, or provide the Logical Destination name created on the Web AS. Then from the available list of operation interfaces, select the web service operation to be consumed.

**PI – SOAP Adapter**

SAP Process Integration (PI) as mentioned earlier is used to integrate applications from one or multiple systems. The SOAP adapter enables you to exchange SOAP messages between remote clients or Web service servers and the Integration Server. The receiver SOAP adapter also allows you to send SOAP messages i.e. invoke Web Services. The “SOAP with Attachments” feature is available with the SOAP adapter for Web Services with attachments.

To access the Web Service you will need the WSDL URL which is specified as the Target URL for the channel. Additional settings for authentication (User and / or Certificates), Proxy Server, SOAP Attachments etc. can be provided.

**Adobe Flex**

Adobe Flex is a collection of technologies released by Adobe Systems for the development and deployment of cross platform rich Internet applications based on the Adobe Flash platform. In the process of offering is services to their internals, one of the directions SAP is taking is exposing more and more interfaces in the form of Web Services, and Adobe Flex client is an excellent consumer of such services.

Consuming Web Services in Adobe Flex is facilitated with the `WebService` tag. The WSDL and / or service endpoint URI can be added in the `wsdl` and `endpointURI` attributes of the specified tag. The operation to be access is defined using `operation` tag and its attribute `name`. One can execute the Web Service with the `send()` method called on the operation of the Web Service.
Related Content

SAP NetWeaver Blog: How to consume Web Services with Visual Composer
Web Services – Standalone and Deployable Proxy
Consuming ABAP Web Services using Flex
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.