Providing and Consuming Web Services in ABAP

Peter McNulty
SAP NetWeaver Product Management
Agenda

Overview of the Web Service technology

Inside-out approach: Creating a Web Service from existing business functionality

Demo: Create a Web Service

Consuming a Web Service from an ABAP program

Demo: Consume a Web Service

Summary
Overview of the Web Service technology

Inside-out approach: Creating a Web Service from existing business functionality

Demo: Create a Web Service

Consuming a Web Service from an ABAP program

Demo: Consume a Web Service

Summary
WEB SERVICES are
SELF CONTAINED
and SELF DESCRIBING
APPLICATION FUNCTIONALITIES
that can be
PROCESSED
through
OPEN INTERNET STANDARDS
Web Services Overview: The Nature of Web Services

**Web Services**

- act like a black-box that may require input and deliver a result
- work on top of any communication technology stack
- can be published, discovered and invoked based on open technology standards
- work in synchronous and asynchronous scenarios
- facilitate integration within an enterprise as well as cross enterprises
Web Services Overview: The Paradigm

1. Create Web service
   - Create WSDL
   - Publish WSDL (opt.)

2. Discover Web service
   - Develop Client App

3. Web service Execution

- UDDI Service Directory
- WSDL Document
- SOAP
- Web service Provider
- Bus. Functionality

Create WSDL
Publish WSDL
Web Services Overview: Examples

- Intelligent product catalog search
- Product availability check
- Pricing inquiry
- Customer credit check
- Order status check
- Vendor managed inventory
  - Demand forecasts, stock replenishment …
- Dynamic auctioning and bidding
- Publish and analyze financial reports (XBRL based)
- Electronic bill presentment and payment
- Matching vacancies and job applicants profiles
- Postal service address check
- UDDI registration and discovery
- Automated web searches (Google)
Web Services Overview: Example Get Employee Address Info

Manager

Portal iView

Company

Get Employee Address Information

Service Provider

CRM

Employee Contact Information

HR

Private Address

Archiving Database

○ = Web Service
Web Services Overview: Fundamental Elements of the complete Web Service Solution

Open Technology Standards for Web Services

XML, WSDL, SOAP, UDDI, WSI

Web Service Technology

SAP NetWeaver

Referent to business semantics

RosettaNet, Spec2000, HR-XML, XBRL, IFX, papiNet, ....
SAP NetWeaver is based on Open Standards

SAP NetWeaver Supports Standards on Technology, Framework and Business Level

Built on a consistent technology and application stack

Examples for Supported Standards in SAP NetWeaver

- W3C
  - SOAP 1.1
  - WSDL 1.1
  - XML Schema
  - XML Signature
  - XSL
- BPML
- CIM
- OASIS
  - SAML
  - UBL
  - UDDI 2.0 / 3.0
  - WSRP
  - WS-Security
  - XML.org
- Wf-XML
- WS-I
  - Basic Profile 1.0
- XMLA
- UDDI4J
- JAX-RPC
- JAXM
- ... and many others
Universal Description, Discovery and Integration (UDDI)

Describes how to advertise and discover a Web service

- Differentiates Web service provider, Web service and Web service type
- Holds metadata that can be used to search for services (names, IDs, categories, types, etc.)
- Specifies the interface for Web service registries

UDDI Business Registry

- THE directory for Web services on the Internet
- Publicly available, free of charge
- Operated by SAP, IBM, Microsoft, and NTT Communications
- UDDI Version 3 Beta available now
- SAP’s node at http://uddi.sap.com
Web Service Description Language

Describes the basic characteristics of a Web service

- Supported operations and their data format   e.g. xCBL Order
- Supported protocols       e.g. SOAP
- Network address               e.g. http://a.com/order

Further information

- Specification: http://www.w3.org/TR/wsd1l2/
WSDL Document Structure

<definitions>
  <types>
    The data types used by the web service
  </types>
  <message>
    The messages used by the web service
  </message>
  <portType>
    The operations performed by the web service
  </portType>
  <binding>
    The communication protocols used by the web service
  </binding>
</definitions>
WSDL Types Element

Defines the data types used by the web service

```xml
<?xml version="1.0" encoding="utf-8" ?>
<wsdl:definitions targetNamespace="urn:sap-com:document:sap:rfc:functions"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:tns="urn:sap-com:document:sap:rfc:functions" elementFormDefault="unqualified"
    attributeFormDefault="qualified">
  <wsdl:types>
    <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                xmlns:tns="urn:sap-com:document:sap:rfc:functions"
                targetNamespace="urn:sap-com:document:sap:rfc:functions" elementFormDefault="unqualified"
                attributeFormDefault="qualified">
      <xsd:element name="Z_EMPL_PRIVATE_ADDRESS_QUERY">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="ADDRESS_TYPE" type="xsd:string" />
            <xsd:element name="EMPLOYEE_ID" type="xsd:string" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="Z_EMPL_PRIVATE_ADDRESS_QUERYResponse">
      </xsd:element>
    </xsd:schema>
  </wsdl:types>
</wsdl:definitions>
```
WSDL Message Element

Defines the data elements of an operation

```xml
<?xml version="1.0" encoding="utf-8" ?>
<wsdl:definitions targetNamespace="urn:sap-com:document:sap:rfc:functions"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xsd:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="urn:sap-com:document:sap:rfc:functions"
        elementFormDefault="unqualified"
        attributeFormDefault="qualified">
        <xsd:element name="Z_EMPL_PRIVATE_ADDRESS_QUERY">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="ADDRESS_TYPE" type="xsd:string" />
                    <xsd:element name="EMPLOYEE_ID" type="xsd:string" />
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="Z_EMPL_PRIVATE_ADDRESS_QUERYResponse" />
    </xsd:schema>
</wsdl:definitions>
```

- `<wsdl:message name="Z_EMPL_PRIVATE_ADDRESS_QUERY">
  `<wsdl:part name="parameters" element="tns:Z_EMPL_PRIVATE_ADDRESS_QUERY" />
</wsdl:message>
- `<wsdl:message name="Z_EMPL_PRIVATE_ADDRESS_QUERYResponse">
  `<wsdl:part name="parameters" element="tns:Z_EMPL_PRIVATE_ADDRESS_QUERYResponse" />
</wsdl:message>

+ `<wsdl:portType name="Z_TE_EMPL01">
+ `<wsdl:binding name="Z_TE_EMPL01SoapBinding" type="tns:Z_TE_EMPL01">
+ `<wsdl:service name="Z_TE_EMPL01Service">
</wsdl:definitions>
```
WSDL portType Element

Defines the supported operations that can be performed and the messages that are involved.

- This is an example of a request-response operation

```xml
<?xml version="1.0" encoding="utf-8" ?>
<wsdl:definitions targetNamespace="urn:sap-com:document:sap:rfc:functions"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/
    xmlns:tns="urn:sap-com:document:sap:rfc:functions"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  + <wsdl:types>
    + <wsdl:message name="Z_EMPL_PRIVATE_ADDRESS_QUERY">
      + <wsdl:message name="Z_EMPL_PRIVATE_ADDRESS_QUERYResponse">
    - <wsdl:portType name="Z_TE_EMPL01">
      - <wsdl:operation name="Z_EMPL_PRIVATE_ADDRESS_QUERY">
        <wsdl:input message="tns:Z_EMPL_PRIVATE_ADDRESS_QUERY" />
        <wsdl:output message="tns:Z_EMPL_PRIVATE_ADDRESS_QUERYResponse" />
      </wsdl:operation>
    </wsdl:portType>
  + <wsdl:binding name="Z_TE_EMPL01SoapBinding" type="tns:Z_TE_EMPL01">
  + <wsdl:service name="Z_TE_EMPL01Service">
</wsdl:definitions>
```
WSDL Bindings Element

Defines the message format and protocol details for a web service

The `soap:binding` element has two attributes. In this example ...

- **style** attribute is document
- **transport** attribute defines the SOAP protocol to use HTTP

The operation element defines each operation that the port exposes.

- input and output operations are encoded as "literal"

```xml
<?xml version="1.0" encoding="utf-8" ?>
<wsdl:definitions targetNamespace="urn:sap-com:document:sap:rfc:functions"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <wsdl:types>
    <wsdl:message name="Z_EMPL_PRIVATE_ADDRESS_QUERY">
      <wsdl:element name="Z_EMPL_PRIVATE_ADDRESS_QUERYResponse"/>
    </wsdl:message>
    <wsdl:portType name="Z_TE_EMPL01">
      <wsdl:binding name="Z_TE_EMPL01SoapBinding" type="tns:Z_TE_EMPL01">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
        <soap:operation name="Z_EMPL_PRIVATE_ADDRESS_QUERY">
          <soap:operation soapAction="" />
          <wsdl:input>
            <soap:body use="literal" />
          </wsdl:input>
          <wsdl:output>
            <soap:body use="literal" />
          </wsdl:output>
        </soap:operation>
      </wsdl:binding>
    </wsdl:portType>
  </wsdl:types>
</wsdl:definitions>
```
WSDL Service Element

Defines the set of ports supported by the web service

For each supported protocol there will be one port element

```xml
<?xml version="1.0" encoding="utf-8" ?>
<wsdl:definitions targetNamespace="urn:sap-com:document:sap:rfc:functions"
 xmlns:soap="http://www.w3.org/2001/XMLSchema">
  + <wsdl:types>
    + <wsdl:message name="Z_EMPL_PRIVATE_ADDRESS_QUERY">
    + <wsdl:message name="Z_EMPL_PRIVATE_ADDRESS_QUERYResponse">
    + <wsdl:portType name="Z_TEEMPL01">
      - <wsdl:binding name="Z_TEEMPL01SoapBinding" type="tns:Z_TEEMPL01">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
        - <wsdl:operation name="Z_EMPL_PRIVATE_ADDRESS_QUERY">
          <soap:operation soapAction="" />
          - <wsdl:input>
            <soap:body use="literal" />
          </wsdl:input>
          - <wsdl:output>
            <soap:body use="literal" />
          </wsdl:output>
        </wsdl:operation>
      </wsdl:binding>
    </wsdl:portType>
  </wsdl:types>
</wsdl:definitions>
```
Web Services Overview: Standards – SOAP Structure and Features

Transport Binding

Message Format

Header

Body

Application Data

SOAP

HTTP, SMTP, ...

Protocol specific data (e.g. quality of service)

Application-specific data

Type system
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>

<SOAP-ENV:Envelope
xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SAP="http://sap.com/xi/XI/Message/30">

<SOAP-ENV:Body>

<pns:getEmployeePersonalAddress
xmlns:pns="urn:AddressWSVi">

<employeeID>1001</employeeID>
<addressType>HM</addressType>
<actualDate>2005-08-22</actualDate>

</pns:getEmployeePersonalAddress>

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>
“Accelerating the adoption of Web services by reducing the cost associated with standards adoption”

Focus on Web service interoperability

- Select, integrate and profile existing industry standards
- Provide implementation guidance and tools

Deliverables

- Profiles specify conformance requirements for Web services
- Testing Tools can be used to test profile conformance
- Sample Applications demonstrate interoperable implementations

WS-I Basic Profile 1.0

- Industry’s common denominator for Web service applications
- Covers HTTP, XML, XML Schema, SOAP, WSDL, and UDDI
- Finalized August 12, 2003
Web Services Technology Overview: Providing Web Services based on Open Standards
Web Services Technology Overview: Consuming Web Services based on Open Standards

Development Environments

- ABAP
- Java

Web service Meta Data

Web service Client Application

Web service Proxy

Web service Runtime

SAP Web Application Server

UDDI Registry

WSDL

SOAP

Web service

Bus. Functionality

Web service Provider

© SAP AG 2005, SAP TechEd ’05 / CD153 / 23
Web Services Technology Overview: Value Added Web Services with XI

- "Mediated" Web services with Routing, Mapping and BPM
- Web service Client
  - SOAP
  - SAP XI Integration Server (XI Protocol or SOAP)
  - Adapter
  - 3rd Party
  - SAP System
  - Adapter
  - SAP Web AS ≥ 6.40
- "Basic" Web services
- Point-to-Point Web services

© SAP AG 2005, SAP TechEd '05 / CD153 / 24
Overview of the Web Service technology

**Inside-out approach: Creating a Web Service from existing business functionality**

Demo: Create a Web Service

Consuming a Web Service from an ABAP program

Demo: Consume a Web Service

Summary
Two basic Ways of Developing an Interface

**Inside-Out**
- Start of implementation in the backend system
- Interface semantics pre-defined and reflects directly implementation
- Drawback: Implementation details are visible in the interface

**Outside-In**
- Central design of interfaces that make business sense in SAP XI Integration Repository
- Using global data types
- Reference to Governance Process for PIC
- Generation of proxies from abstractly modeled interface
- Implementation of “glue code” between proxy and business function
- Implementation can be changed without changes in interface
Creating a Web Service in less than ONE Minute

1. Implement Business Application
   - No Web service specific programming
   - New or existing applications
   - Defines standard interfaces
   - Well known programming models

2. Generate Web Service
   - Wizard based approach
   - Based on preconfigured profiles
   - Based on abstract behavior
   - Available for ABAP & Java

3. Activate / Deploy Web Service
   - Deployment in Java
   - Activation in ABAP

*Note:
The terms Service Interface and Variant refer to the ABAP 7.0 implementation. They correlate to Virtual Interface and WS Definition on the Java stack and ABAP 6.40 release.
The Web Service Creation Wizard – Background

- Out-of-the-box Web services enablement of SAP solutions
- Allows to expose an existing endpoint as a Web Service with a few mouse-clicks
- Only shows the most important settings in the wizard
  - Default values (profiles) for other settings
- Hides technical details
- Implicitly creates all necessary objects
- ABAP Web Service Wizard transaction: WS_WZD_START
Demo – Web Service Creation Wizard

Enter a name of the Service Definition and select the endpoint type.

To change the Web Service, use the ABAP Repository Browser (transaction SE80).

Service Definition: Z_WS_Empi_Address_Query
Short Description: Z_WS_Empi_Address_Query
Endpoint Type: Function Module
Demo – Web Service Creation Wizard

Choose Endpoint

Overview
Create Service
Choose Endpoint
Choose Operations
Configure Service
Complete

Select the Function Module

Enter the name of the function module that is to be offered as Web Service.

If you select the checkbox 'Mapping of Names', the existing descriptions of the end point are applied. Initial letters are capitalized and underscores removed.

Function Module: Z_EMPL_PRIVATE_ADDRESS_QUERY

Name Mapping: [ ]
Choose a Profile for Security Settings.

If you select the checkbox 'Release Service for Runtime', the Web Service is immediately released when completed. If the field is not selected, the Service can be released later in transaction WSCONFIG (Release Web Services for SOAP Runtime).

- Profile: Basic Authorization: SOAP Pro
- Profil für SOAP 1.1 mit zustandsloser HTTP Kommunikation und einfacher Authentisierung
- Release Service for Runtime
The Web Service Homepage ...

- ... is available for each deployed Web Service
- ... shows the documentation for the Web Service
- ... allows to retrieve WSDL descriptions in different styles
- ... offers testing capabilities
- ... can be used to test SAP and non-SAP Web services
- ... is a JSP page which requires the Java stack to be installed
Overview page

Z_WS_EMPL_ADDRESS_QUERY

Overview

WSDL:

Description:
Web Service Z_WS_EMPL_ADDRESS_QUERY

UDDI Publications:
No information available

Features:

Design-time features for Port Type: Z_WS_Empl_Address_Query

<table>
<thead>
<tr>
<th>Feature</th>
<th>Property</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

http://www.sap.com/websit/33/soap/features/authenticationsp/authenticationsp?name=Note
Result of the Web service test.
Creating a Web Service Step-by-Step

The Step-by-Step approach

- Create all objects by hand
- Full flexibility and options
- More expenses
- … but still configuration only
Web Service Interface – Background

The Service Interface is the interface visible to clients (via WSDL, UDDI,...)

A Service Interface …

- … provides abstraction from concrete implementation (=endpoint)

- … allows to publish a “view” on existing implementations as Web Service Interfaces
  - i.e. renaming, hiding of parameters/methods, default values

- … allows to define how the Web Service Interface is represented in the SOAP message

Note: The term Service Interface refers to the ABAP 7.0 implementation. It correlates to a Virtual Interface on the Java stack and ABAP 6.40 releases.
Demo – Web Service Interface
Motivation

A Web service is not only described by its interface but also by its behavior with respect to:
- stateless/stateful communication,
- transactional behavior,
- security requirements, ...

Idea

- describe this kind of behavior via abstract features
- leave it to the Web Service Configurator to assign technical protocols to these abstract features

Example:
- Feature: Stateful
- Technical Protocols: Stateful via HTTP-Cookies, Stateful via URL-Extensions

Note: The term Web Service Variant refers to the ABAP 7.0 implementation. It correlates to a Web Service Definition on the Java stack and ABAP 6.40 releases.
**Benefits**

- The same Web Service Variant / Definition can be deployed to various application servers which may differ in their technical capabilities.

- Web Service Client implementations do not depend on technical server information:
  - Clients are generated based on implementation-independent Web Service variants / definitions.
  - Technical details are configured separately in the Web Service Client Runtime.
Demo – Web Service Variant / Definition

Demo
Web Service Configuration & Deployment

A Web Service Configuration is the assignment of technical protocols to abstract features

- Based on the Web Service capabilities of the specific SAP Web Application Server

- Configuration of additional Web Service Runtime features
  - Logging, Tracing, Monitoring
  - Security Roles
  - …

- Deployment as usual
  - SDM deployment in Java
Demo – Web Service Configuration
Agenda

Overview of the Web Service technology

Inside-out approach: Creating a Web Service from existing business functionality

Demo: Create a Web Service

Consuming a Web Service from an ABAP program

Demo: Consume a Web Service

Summary
Creating a Web Service Client Application

1. Retrieve WSDL
   - From UDDI
   - Via URL (from server)
   - From local Server
   - From File System
   - From XI repository

2. Generate Web Service Proxy
   - Environment specific
   - Available for ABAP & Java

3. Develop Client Application
   - Focus on Business Application
   - SOAP / XML handling hidden by proxy
Demo – Creating a Web Service Client

Create Proxy Object
Overview of the Web Service technology

Inside-out approach: Creating a Web Service from existing business functionality

Demo: Create a Web Service

Consuming a Web Service from an ABAP program

Demo: Consume a Web Service

Summary
Summary

- The **Enterprise Service Architecture (ESA)**, SAP’s approach for building services-oriented business applications utilizes Web Services as vital part of its communication and integration strategy.

- SAP’s rich **business functionality**, in conjunction with state-of-the-art technology, enables you to establish cross-company business processes as an integrated part of your development efforts based on Web Services.

- **SAP NetWeaver Application Server** offers an easy, convenient way to build Web services.
Questions?

Q&A