Applies to:
Enterprise Services Repository & Registry together with SAP NetWeaver Process Integration and SAP NetWeaver Composition Environment.
For more information, visit the SOA Management homepage.

Summary
In the article, we will provide a step-by-step guide for definition of Services in ES Repository, generation of proxies along with implementation, publication and discovery of defined Services in Services Registry and testing of the endpoints in WSNavigator.

Authors: Goyal Arpit, Nedelcho Delchev
Company: SAP
Created on: 30 June 2009
# Table of Contents

**Overview**.............................................................................................................................................................3  
**Defining Service in the ES Repository**................................................................................................................3  
  1. Modelling Service Interfaces in the ES Repository: .................................................................4  
     1.1. Preparation .......................................................................................................................................................4  
     1.2. Create a New Model .........................................................................................................................................8  
     1.3. Create a Service Interface ...............................................................................................................................9  
     1.4. Adding Operations ........................................................................................................................................10  
     1.5. Adding Connections .......................................................................................................................................11  
     1.6. Create Assignment ........................................................................................................................................11  
     1.7. Create Message Types ...............................................................................................................................15  
     1.8. Create Data Types ........................................................................................................................................18  
     1.9. Create Fault-Message Types ..........................................................................................................................20  
     1.10. Activation of Objects ...............................................................................................................................22  
     1.11. Publish Service to Services Registry ..........................................................................................................22  
     1.12. Publishing Service Interface .......................................................................................................................23  
     1.13. Discovery via Services Registry .................................................................................................................24  
  **Proxy Generation and Implementation in Backend**..........................................................................................25  
  1. Create Enterprise Application and EJB projects .............................................................................25  
     1.1. Create Enterprise Application Project .............................................................................................................25  
     1.2. Create EJB Project ........................................................................................................................................25  
     1.3. Configure SAP NetWeaver Developer Studio .................................................................................................26  
     1.4. Create Java Bean Skeleton Using Enterprise Services Browser .................................................................27  
     1.5. Alternative Approach to Generate Java Bean Skeleton Using Service Registry.............................................32  
     1.6. Implement the generated methods, build and deploy on AS Java .................................................................37  
     1.7. Deploy the Application ...............................................................................................................................37  
  **Publication of Services in Services Registry**....................................................................................................38  
  1. Publication Rules...................................................................................................................................................38  
     2. Check the Publication of the Service ...............................................................................................................39  
        2.1 Configuration States in Services Registry ....................................................................................................39  
     3. Test the Endpoint ............................................................................................................................................41  
  **Conclusion** ........................................................................................................................................................43  
  **Copyright** ...........................................................................................................................................................44
Overview

SAP delivers an Enterprise Services Repository & Registry together with SAP NetWeaver Process Integration and SAP NetWeaver Composition Environment.

The Enterprise Services Repository stores the definitions and metadata of enterprise services and provides an integrated modeling environment for defining enterprise services, data types, and other design objects for SOA-based business processes in a standards-compliant manner.

The Services Registry supports the publication, classification, discovery, and testing of enterprise services (SAP-defined, partner-defined or custom-defined) across the IT landscape. This UDDI-compliant registry also enables the management and governance of enterprise services.

In the article, we will provide a step-by-step guide for definition of Services in ES Repository, generation of proxies along with implementation, publication and discovery of defined Services in Services Registry.

Development Steps

1. Definition of Service Interfaces in ES Repository
2. Proxy generation and implementation in backend
3. Publication of services in Services Registry
Defining Service in the ES Repository

1. Modelling Service Interfaces in the ES Repository:

Before we begin with the modelling, look at the diagram below describing the structure of the model we are about to create:

As you can see, the model specifies the objects that build an enterprise service and the relationships between them:

- A business object represents a specific view on a well defined business content
- A service operation belongs to exactly one business object
- A service interface is a group of one or more service operations

With this knowledge, we will now model our enterprise service. In our example the enterprise service is provided by Airlines industry depicting search functionality of available Economic, Business and First Class seats based on Flight ID. Additional parameters are the date of travel and connection ID in case of break journey.

1.1. Preparation

- Log on to the ES Repository using the URL `http://<server>[:<port>]`/rep/start/index.jsp
- For modeling in the ES Repository, Software Component version (SWCV) is needed. They are imported from System Landscape Directory (SLD).
  - Click on the `Create Object` icon from the toolbar to open the `Create Object` dialog in Enterprise Services Builder.
o Select Work Area → Software Component Version on the left side navigation pane and Import from SLD radio-button on the right side navigation pane.

o Choose the Display button and select the required SWCVs (say SAPDEMO 7.11 of testdemo) as shown in List of Software Component Versions.
- Set Original Language to English and save the Software Component Version as shown in the figure below.

- Right-click on the Software Component Version to open the context menu and select New.
- Add a new namespace as shown in the figure below

![Create object](image)

**Note:** I have used sap.com in the namespace definition. In customer scenario it is strongly recommended to use customer namespace.

- Click on the Save button to save namespace definition.

![Edit Namespace Definition](image)
To create an object under the namespace (http://sap.com/WSTest), right-click on the namespace to open context menu and select New as shown below. In the Create Object dialog box, select the required entity type.

1.2. Create a New Model

As shown below create a new Model of type “SAP Integration Scenario Model”. Make sure the Model Type is “SAP ProComp Model”.

We can now model the process component. We will create a service interface, a business object and a respective operation, which relates to this business object.
1.1.1. Create a business object by selecting the business object icon from the left side panel and draw it in the canvas area. Call this object “Flight Booking”.

1.3. Create a Service Interface

By selecting the service interface icon from the left side panel and draw it in the canvas area. Call it “Query Flight Booking In”.

[Diagram of a business object and service interface]
1.4. Adding Operations

Click on operation icon from the left side panel and draw it in the canvas area. Call it "Find Flight Booking ForBusiness".

A Create relationship popup window will come. Select the Connection occurred check box and click OK.

Repeat step 1.2.3 to create another operation. Call it “Find Flight Booking By ForEconomy”.

1.5. Adding Connections

Select the Connection icon and draw it from the business object to the service operation. Click to save the model.

Now that we modelled our process component, we will assign the model objects to technical objects.

1.6. Create Assignment

From the model, right-click on the service interface “Query Flight Booking In” box and choose: Service Interface Assignment → Create Assignment.
Rename the Name to FlightSeatAvailabilityQueryIn. From the value list, select the namespace http://sap.com/WSTest. Click to save the model.

1.1.2. From the model, right-click on the service interface “Query Flight Booking In” and choose Service Interface Assignment → Open Assignment. Click on the Create button on the popup window.
Change the Category to “Inbound”, Mode to “Synchronous” and name of the Operation as FlightSeatAvailabilityEconomy in the service interface.

Click icon to add another operation to FlightSeatAvailabilityQueryIn. Name it FlightSeatAvailabilityBusiness and change Mode to “Synchronous”.

**Note:**
- Inbound: Determines the proxy behavior as provide proxy.
- Synchronous: The consumer of the service is blocked until the service call is returned.
1.1.3. From the model, right-click on operation “Find Flight Booking By ForEconomy” and choose Operation Assignment → Open Assignment.

Use the Display Input Help and select FlightSeatAvailabilityQueryIn.FlightSeatAvailabilityEconomy operation.

Repeat step 1.8 on operation “Find Flight Booking By ForBusiness” from the model and select FlightSeatAvailabilityQueryIn.FlightSeatAvailabilityBusiness operation.

Click ✋ to save the model.
1.7. Create Message Types

We need to define Request and Response Message Types for our service interface operations, but as message types comprise of data type, we need to define them first. From the left side menu, right-click on the namespace node and select New.

In the Create Object dialog, choose Interface Object → Data Type. Name the data type as FlightID. Default Classification is Free-Style Data Type, do not change it.

**Note:** Data Type represents the XML Schema based on which the service request-response is sent. Using Free-Style Data Type both simple and complex type XML Schema can be defined.

Now we will define the structure of the Data Type.
1.1.4. Click on the plus sign, choose Insert Subelement. Call it AirlineID. Also set the type to be xsd: string by selecting the XSD Types from the dropdown list and then select xsd: string.

1.1.5. Repeat step 1.9.1 to create FlightID data type as shown below:

Click to save the data type object.
Note: a) You can use ▲▼ to move Subelement Up/Down.

b) There are two options available – Subelement and Attribute when icon is clicked. Insert Subelement is similar to like adding a sub node in XML Schema node. Similarly Insert Attribute is like adding attribute to any XML Schema node.

For more information about how to model Data Types in ES Repository: http://help.sap.com/saphelp_nwpi71/helpdata/en/45/607415b5b33bdbe10000000a1553f7/content.htm

1.1.6. Create two more Data Type FlightSeatAvailabilityQuery and FlightSeatAvailabilityResponse as shown below.

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
<th>Type</th>
<th>Details</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FlightID</td>
<td>Element</td>
<td>FlightID</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
<th>Type</th>
<th>Details</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FlightID</td>
<td>Element</td>
<td>FlightID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AirlineID</td>
<td>Element</td>
<td>string</td>
<td>minLength=3</td>
<td>Airline ID</td>
</tr>
<tr>
<td>ConnectionID</td>
<td>Element</td>
<td>string</td>
<td>minLength=5, maxLength=20</td>
<td>Connection ID</td>
</tr>
<tr>
<td>FlightDate</td>
<td>Element</td>
<td>date</td>
<td></td>
<td>Flight Date</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
<th>Type</th>
<th>Details</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FlightID</td>
<td>Element</td>
<td>FlightID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EconomyMaxSeg</td>
<td>Element</td>
<td>integer</td>
<td></td>
<td>Max Economy Class Seat</td>
</tr>
<tr>
<td>EconomyFreeSeg</td>
<td>Element</td>
<td>integer</td>
<td></td>
<td>Free Seat Economy Class</td>
</tr>
<tr>
<td>BusinessMaxSeg</td>
<td>Element</td>
<td>integer</td>
<td></td>
<td>Max Business Class Seat</td>
</tr>
<tr>
<td>BusinessFreeSeg</td>
<td>Element</td>
<td>integer</td>
<td></td>
<td>Free Seat Business Class</td>
</tr>
<tr>
<td>FirstMaxSeg</td>
<td>Element</td>
<td>integer</td>
<td></td>
<td>Maximum First Class Seat</td>
</tr>
<tr>
<td>FirstFreeSeg</td>
<td>Element</td>
<td>integer</td>
<td></td>
<td>Free Seat First Class</td>
</tr>
</tbody>
</table>
### 1.8. Create Data Types

Now we have defined the data types, we will now create message types. For technical reasons, a data type alone is not sufficient to describe the instance of a message: Data types are defined in XML Schema as abstract types that are not yet tied to an element. You can only describe an instance of a message when you have specified a data type as an element type. Therefore, a message type defines the root element of a message.

From the left side menu, right-click on the namespace node and select New:

1.1.7. Create message type and name it FlightSeatAvailabilityQuery.

#### Message Type

<table>
<thead>
<tr>
<th>Name</th>
<th>FlightSeatAvailabilityQuery</th>
</tr>
</thead>
<tbody>
<tr>
<td>Namespace</td>
<td><a href="http://sap.com/MSTest">http://sap.com/MSTest</a></td>
</tr>
<tr>
<td>Software Component Version</td>
<td>SAFDEMO 711 cfestdemo</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>Folder</td>
<td></td>
</tr>
</tbody>
</table>
Drag and Drop the FlightSeatAvailabilityQuery data type into the icon.

1.1.8. Repeat step 1.10.1 to create two more message type as shown below. Name them FlightSeatAvailabilityQueryBusiness and FlightSeatAvailabilityQueryResponse.

FlightSeatAvailabilityQueryBusiness using data type FlightSeatAvailabilityQuery

Click to save the message type object.
FlightSeatAvailabilityResponse using data type FlightSeatAvailabilityResponse

For more information on message types:
http://help.sap.com/saphelp_nwpi71/helpdata/en/2d/c0633c3a892251e10000000a114084/content.htm

1.9. Create Fault-Message Types

When an application-specific error occurs, the provider (inbound services) can send a fault message to report the error. For specifying this information we will create Fault Message Type.

From the left side menu, right-click on the namespace node and select New:

In the Create Object dialog, choose Interface Object → Fault Message Type

Create Fault Message type ‘FlightNotFound’ as shown in figure below:
For more information on fault message types:

1.10. **Activation of Objects**

For activating all the objects, go to Change List tab page and Activate the changelist as shown below:

![Change List View](image)

1.11. **Publish Service to Services Registry**

1.1.9. Configuring a central Services Registry:


1.1.10. You can also use the Wizard based configuration:

1.12. Publishing Service Interface

All active service interfaces can be published to Services Registry. In the WSDL tab page of Service Interface Editor, click on Publish button.
1.13. Discovery via Services Registry

After publishing the service interfaces from the ES Repository, you can discover the model via the Services Registry:

1.1.11. Log on to the Services Registry using the URL

http://<server>:<port>/sr_central

The state of just published Service Interface is modelled.
Proxy Generation and Implementation in Backend

1. Create Enterprise Application and EJB projects

To create an implementation based on that interface, you can use development tool like SAP NetWeaver Developer Studio, which have to be installed on your development workstation. After launching NWDS you can follow the steps below:

1.1. Create Enterprise Application Project

![Enterprise Application Project creation](image1.png)

1.2. Create EJB Project

![EJB Project creation](image2.png)
1.3. Configure SAP NetWeaver Developer Studio

Windows ➔ Preferences

Ensure that the Secure Connection (HTTPS) is unchecked. Click Apply and OK.
1.4. Create Java Bean Skeleton Using Enterprise Services Browser

1.1.1. Open the Enterprise Services Browser view

Select *Window* ➔ *Show View* ➔ *Other…*

Select *Enterprise Service Browser* from the *Show View* dialog.

In the Enterprise Service Browser click icon to connect to Enterprise Service Repository. Enter the credentials to logon to Enterprise Service Repository.
1.1.2. Generate Java Bean Skeleton

Search for the service under the software component version in the view (in this case FlightSeatAvailabilityQueryIn under SAPDEMO, 711 by testdemo). Right-click on the service FlightSeatAvailabilityQueryIn and select Generate Java Bean skeleton.
Reduce the slide bar to Develop Service and select *Web service runtime: Apache Axis* to change the Service Deployment Configuration. Make it to *SAP Netweaver*. 

![Web Service Configuration Diagram](image-url)

![Service Deployment Configuration Diagram](image-url)
Change the Service project from WebServiceProject to FlightSeatAvailabilityQueryIn_ejb
Web Services

Review your Web service options and make any necessary changes before proceeding to the next page.

Configuration:

Server: SAP Server
Web service runtime: SAP Netweaver
Service project: FlightSeatAvailabilityQueryIn_ejb
Service EAR project: FlightSeatAvailabilityQueryIn_ear

Press Next.

Update WSDL dialog pops-up informing WSDL doesn't contain any services. Press OK.

Go through the next steps with default settings and press Finish.
1.5. Alternative Approach to Generate Java Bean Skeleton Using Service Registry

1.1.3. Import the WSDL using ‘Import’ feature

1.1.4. Select Web Services → WSDL.
1.1.5. Choose the Services Registry option:

1.1.6. Search for the specific model by its name and state:
1.1.7. Choose it from the Service Definitions table and click *Finish.*
1.1.8. Generate Java bean skeleton

To generate the Java bean skeleton based on the imported WSDL, right-click on the WSDL item and select from the popup menu:

Inspect the parameters Server, Web service runtime, Service (EJB) project and EAR project:
Usually the model WSDLs contains only wsdl:portType definition without wsdl:binding and wsdl:service. Since the binding information is needed for the proxy generation hence you have to press the:

Update WSDL button in this step:
Go through the next steps with default settings and press Finish.

1.6. Implement the generated methods, build and deploy on AS Java.

1.7. Deploy the Application

Use Add and Remove Projects dialog:

More information about building composite applications with SAP NetWeaver Developer Studio you can find at: [http://help.sap.com/saphelp_nwce711/helpdata/en/48/d05f3ddc3035be10000000a42189b/frameset.htm](http://help.sap.com/saphelp_nwce711/helpdata/en/48/d05f3ddc3035be10000000a42189b/frameset.htm)

How to implement the service provider based on ESR model in ABAP you can find at:
Publication of Services in Services Registry

The default publication rule should automatically register your service to Services Registry after the next execution. If it is ‘inactive’ you always can create your own publication rule:

1. Publication Rules

Logon to NetWeaver Administrator → SOA Management → Application and Scenario Communication → Publication Rules

Publication Rules: Rules Maintenance

New Publication Rule

Select services that will be included in publication rule

Add Service Definition

Found Service Definitions: 1

WSDL PortName: FlightSeatAvailabilityQueryIn
Namespace: http://sap.com/WSTest
State: Configured
Description: No data is available

Publication Rules for ’Y7D.SystemHome.Iddby7d’ System

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FlightSeatAvailabilityQueryIn</td>
<td>Active</td>
<td>FlightSeatAvailabilityQueryIn</td>
</tr>
<tr>
<td>SAP_DEFAULT_PUBLISH_ALL</td>
<td>Inactive</td>
<td>this is a default publish rule</td>
</tr>
</tbody>
</table>
2. Check the Publication of the Service

You can trigger search again and can see both the service definitions – model and configured.

![Services Registry](image)

2.1 Configuration States in Services Registry

2.1.1 Modelled

Configuration states in Services Registry depend on the simplified lifecycle of the service. The design phase ends with the publishing the service interface as a service definition by ES Repository or any other similar third-party tool. This state is classified as ‘Modelled’.

Modelled - service definition (port type) representing only a model of the service; no endpoints available.

2.1.2 Activated/Deployed

The following steps are the implementation of the provider side in Java/ABAP/.Net/other and respectively the consumer side. Depending on the technology used the already implemented component is activated, deployed or published on the corresponding application server (NetWeaver, IIS). At this point the publication to Services Registry can be triggered and the result will be a set of services with state “Activated/Deployed”.

Implementation is activated (ABAP) / deployed (JAVA) on a physical system; endpoints are not configured.
2.1.3 Configured

Actual configuration at the backend is the step when the endpoint are exposed and ready to be used. Different configuration options are available for transport protocol, authentication mechanism, encryption, etc. which may reflect on the location address of the WSDL itself. One service can be configured to expose multiple endpoints. During the automatic publication process this information about the services and the endpoints is transferred again to the central Services Registry.

Configured - endpoints are configured and ready to use.

Inspection of the service with an endpoint can be done via the Services Registry:
3. Test the Endpoint

You can test a call against the endpoint via WSNavigator by pressing Test button:

3.1 Choose an operation from the list and press Next:
3.2 Fill in the parameters:

![WS Navigator with parameters](image)

3.3 By clicking Next you can perform the actual call to the target address of the endpoint and receive the response:

![Result of the call](image)
Conclusion
To summarize we demonstrated the life-cycle of Enterprise Services by Defining, Implementing, Publishing and testing the web service using Enterprise Services Repository and Registry.