Deployable Web Service Configuration and Usage in SAP NetWeaver Developer Studio (NWDS)

Applies to:

SAP Netweaver, SAP Netweaver Developer Studio, SAP-Java, Web services, Web Services Description Language.

For more information, visit the Web Services homepage.

Summary

This paper will guide you in configuring a deployable web-service in Netweaver Developer Studio environment. After configuration, required settings in J2EE Engine are also briefed. Along-with, the code required to call web service after configuration is also mentioned.

Author: Saurabh Agarwal

Company: Steria, India

Created on: 7 September 2009

Author Bio

Saurabh Agarwal is working as a Consulting Engineer with Steria India Ltd. He has been involved in Java based portal development, and also have knowledge of SAP Enterprise portal, Java WebDynpros. He can also develop code in ABAP.

Table of Contents

Introduction

Web services allow different applications from different sources to communicate with each other without time-consuming custom coding.

To consume a Web service, you have to create a Web service client for it. When you create a Web service client, you generate a Web service proxy and then create a client application, which uses the Web service proxy to call the Web service.

To consume a Web service, you can create a deployable or a standalone proxy for the Web service in the SAP NetWeaver Developer Studio using a WSDL document as a basis. This can be done with just a few mouse clicks:

Deployable proxy – A Web service client that must be deployed on the J2EE Engine as an application. 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.

Standalone proxy – A Web service client that generates stubs and runs without the J2EE Engine. 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The drawbacks of this approach are that if a name of a component is changed or requires some modifications, the stub is no longer valid and the whole proxy needs to be regenerated.

In this article we will go through a step by step procedure for creating a Deployable web service proxy and then consuming the same in our web application.

Configuration of Deployable Proxy

This section demonstrates how to configure a deployable proxy in Netweaver Developer Studio. Following components will be built in this process:

- newWebService (Development Component of type web service)
- newProxy (deployable proxy)

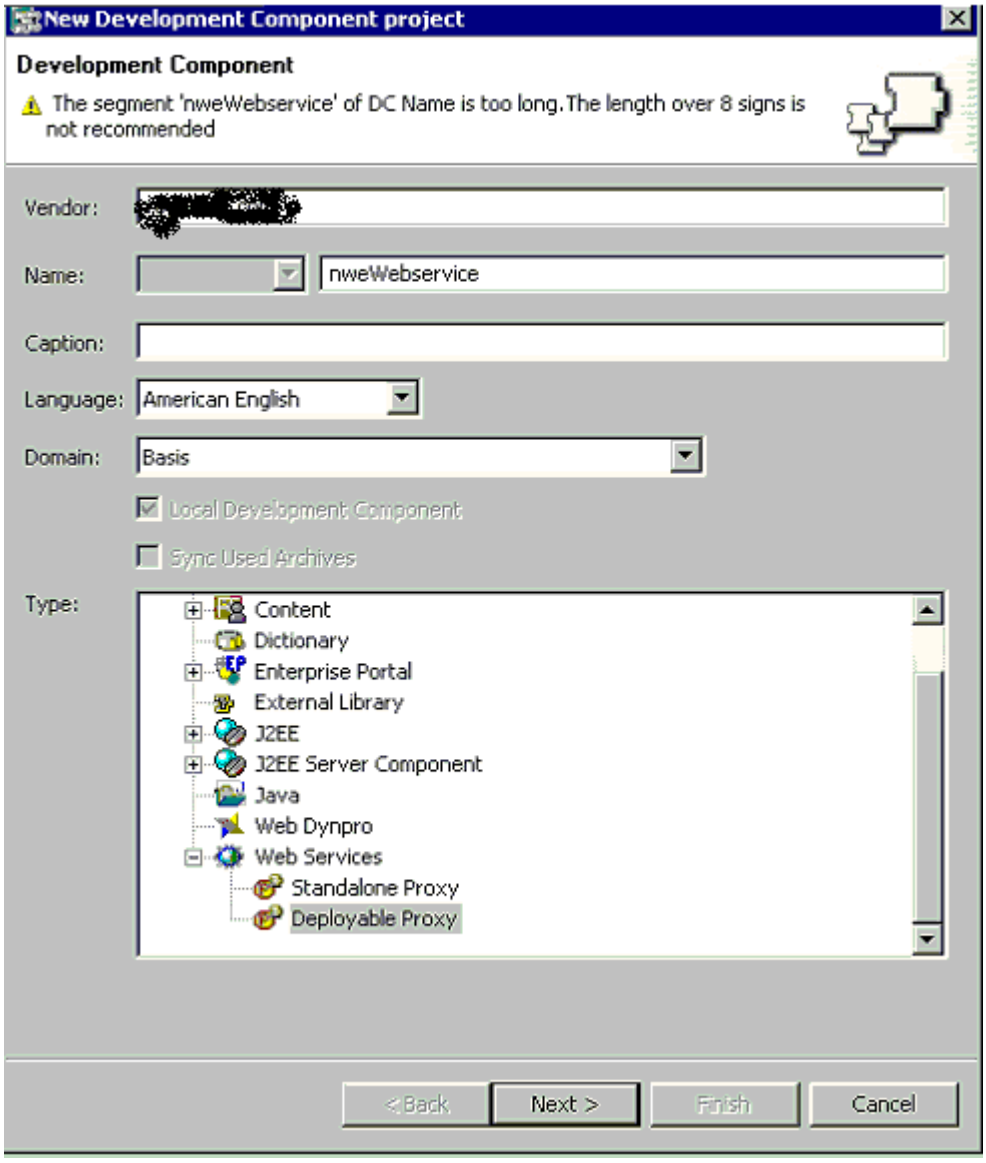
Following components are required initially.

1. UsingWebComp_WA (Web Development Component using above proxy)
2. UsingWebComp_EA (EAR Development Component as deployable unit for web Development Component UsingWebComp_WA)

Step 1 – Create Development Component of Type Web Service

Note: Make sure that WSDL (*.wsdl file) is available with you and is available in your local file system.

Create a new Development Component of type Web Services. Make sure you select Deployable proxy.



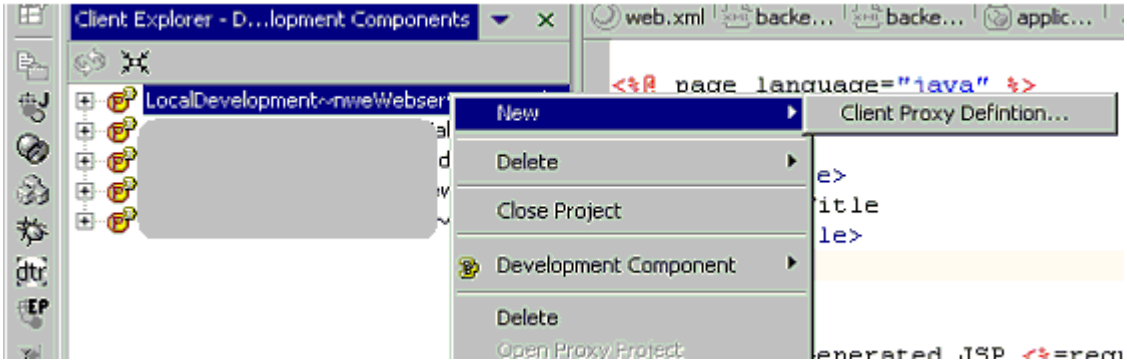
Note: Deployable proxy – a Web service client that must be deployed on the J2EE Engine as an application.

Standalone proxy – a Web service client that generates stubs and runs without the J2EE Engine. This proxy can be used only with the release for which it has been generated.

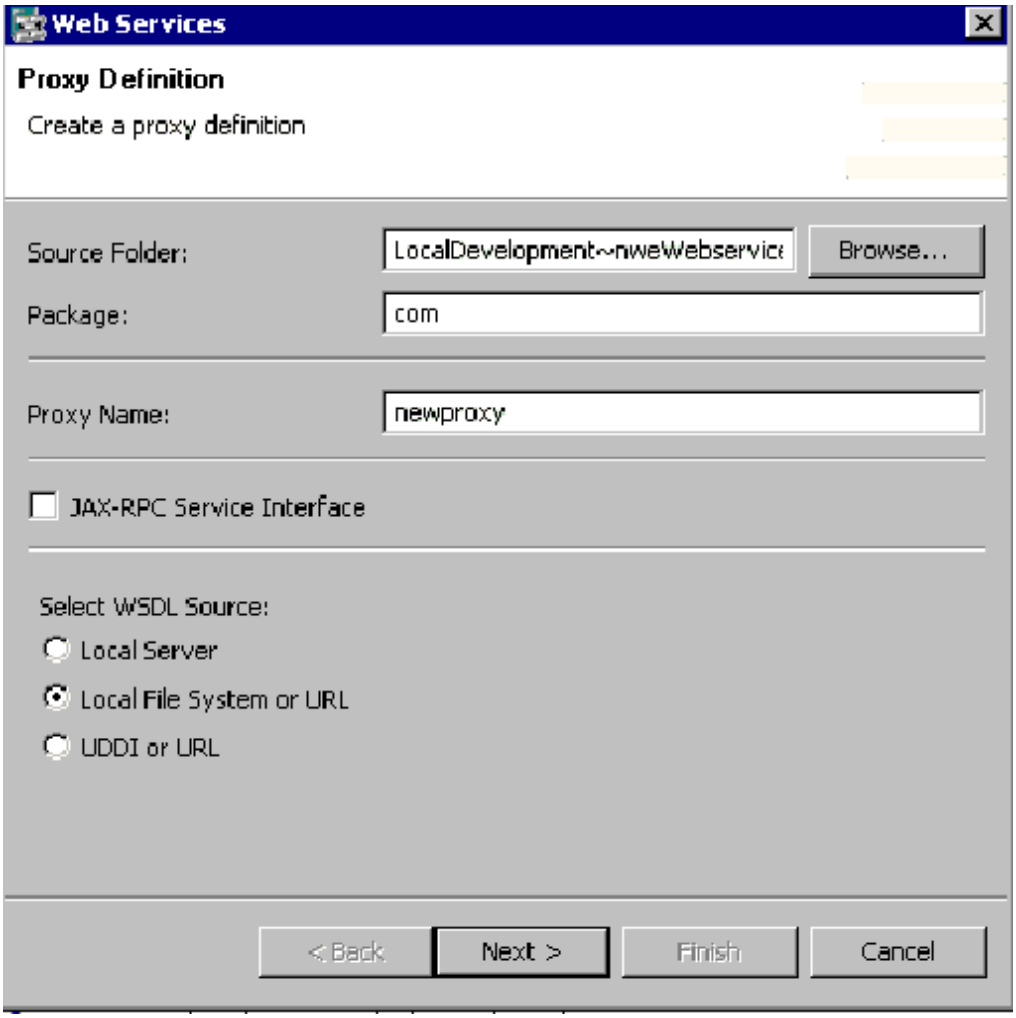
A new web service Development Component project would be created in Netweaver Developer Studio. In Netweaver Developer Studio click on Window -> Open Perspective -> Web services. Now you will be able to view your created web service project.

Step 2 – Create Client Proxy Definition

In above step you have created a Development Component which can hold your proxy definition. Now we will import a proxy in this Development Component. Right click on Created web service Development Component project and select New -> “Client proxy definition”.



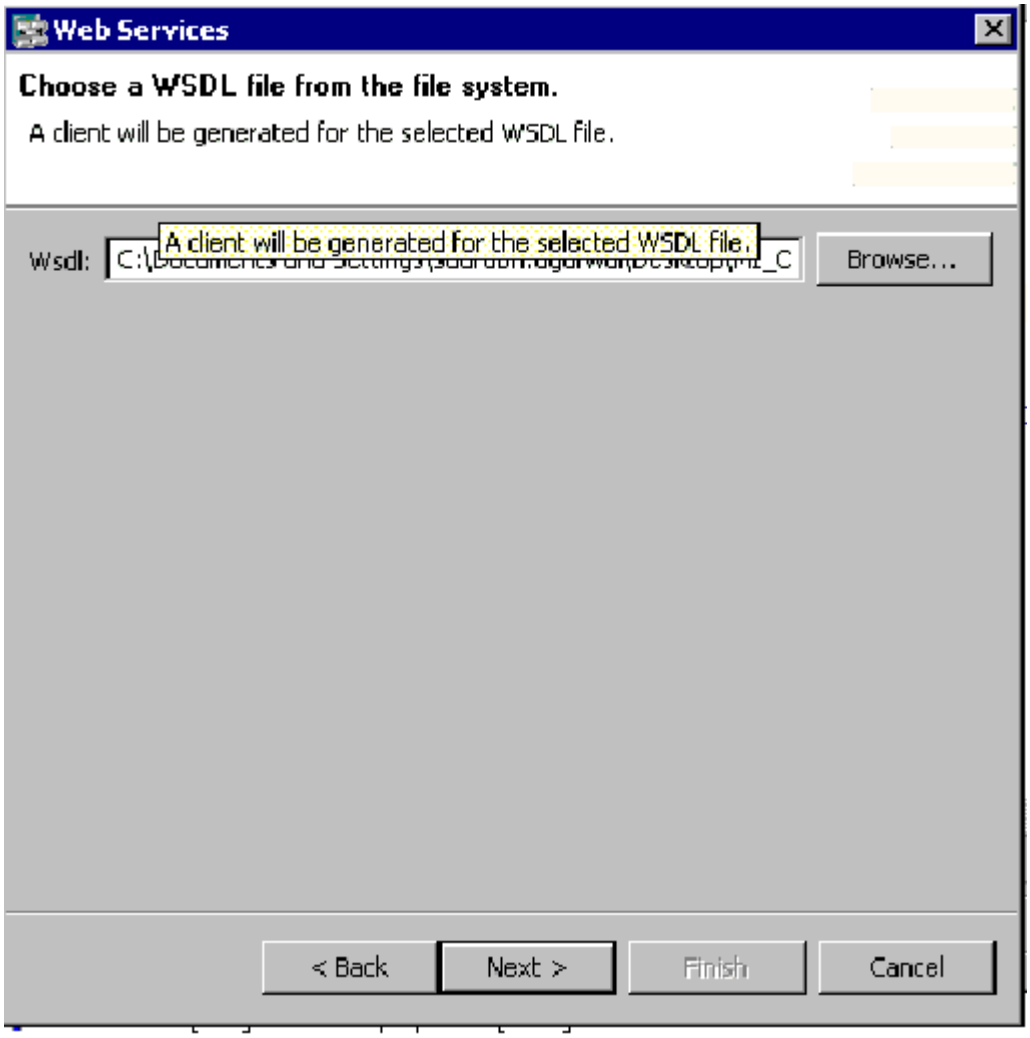
Create a proxy definition dialogue box will get displayed.



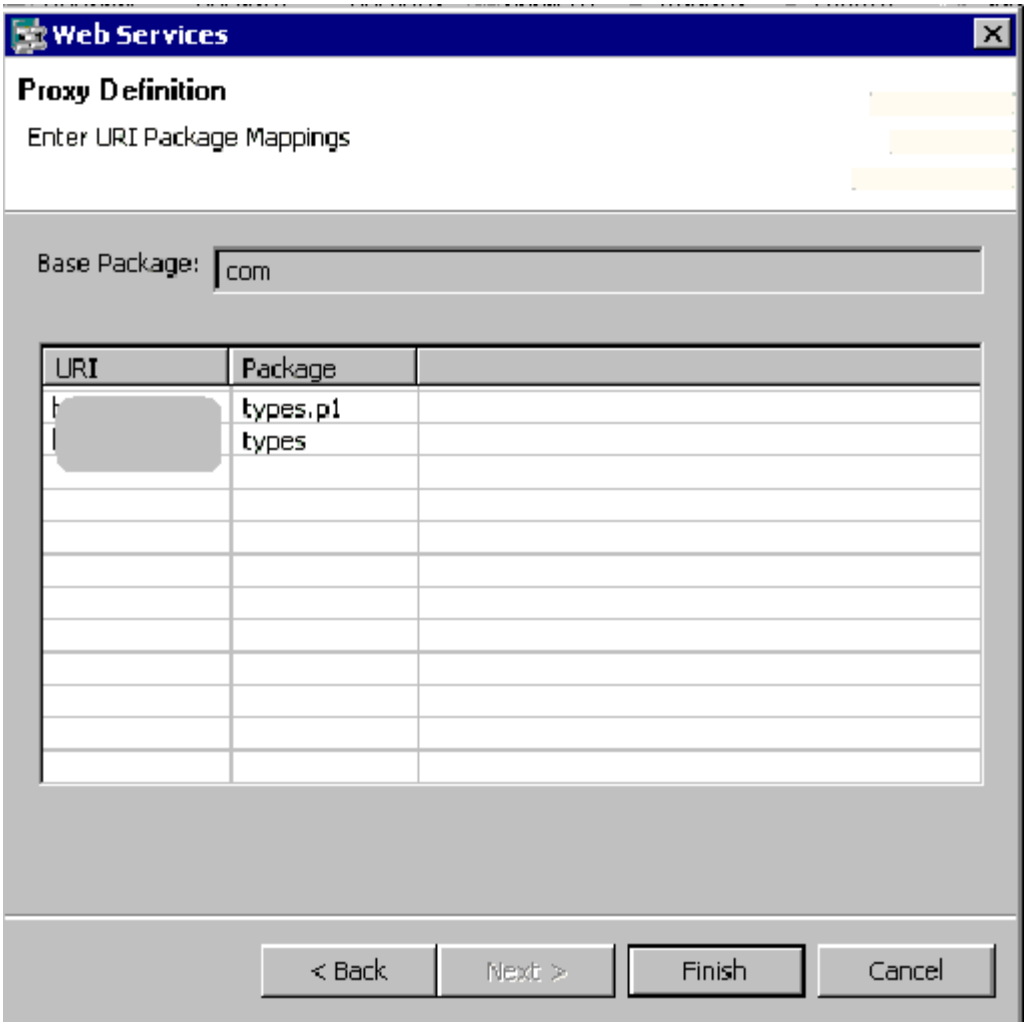
Give package name e.g. com.yourcompany.applicationName and Proxy name.

Note: Do select Local File system or URL.

Point the WSDL file available in your file system.

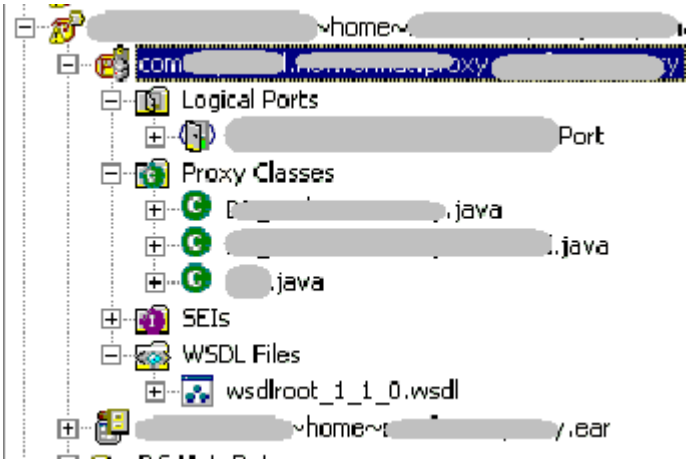


Click on next to view packages of classes generated from proxy definition.



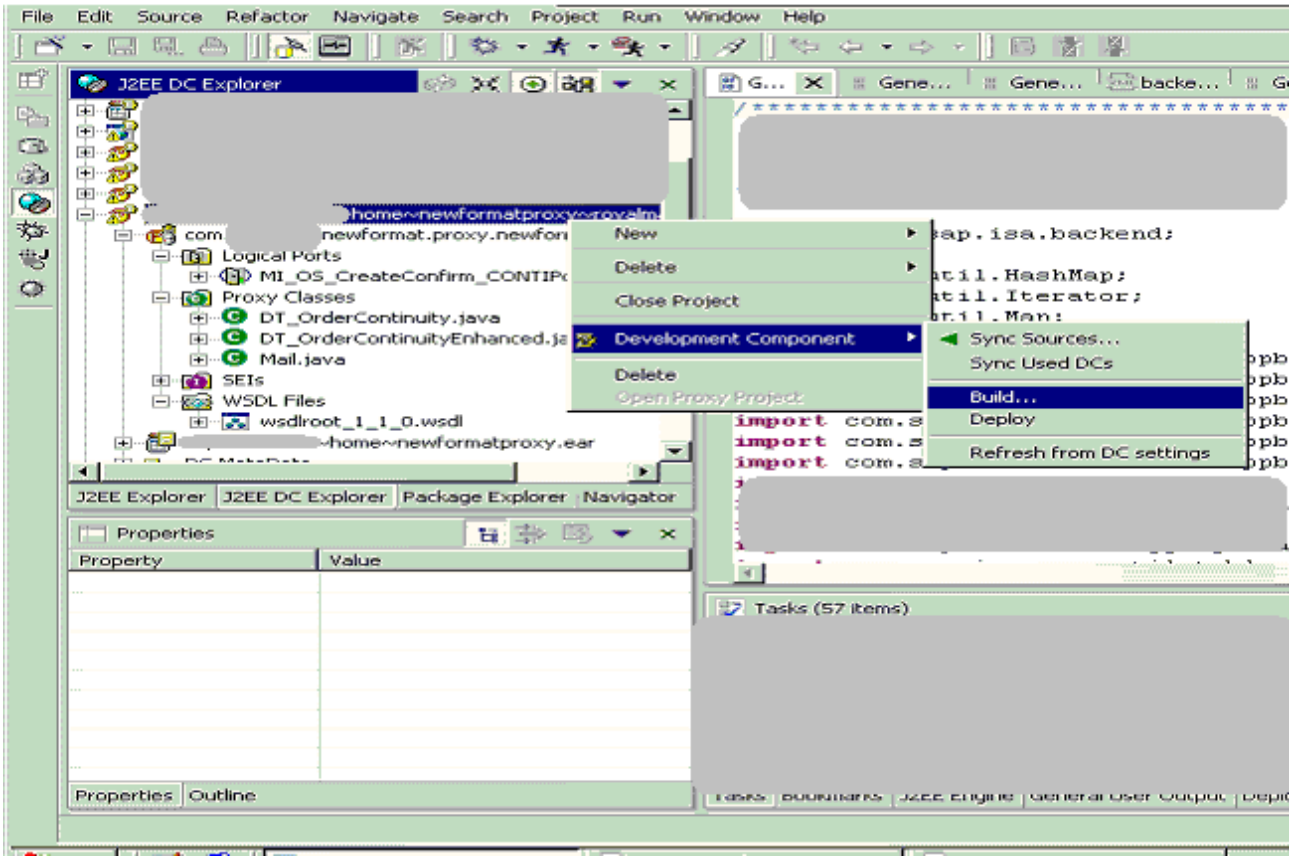
Click on Finish.

You can view generated proxy definition by clicking on Web Service DC -> Proxy Definition



Step 3 – Build and Deploy Web Service Development Component.

You need to build and deploy this web service Development Component.

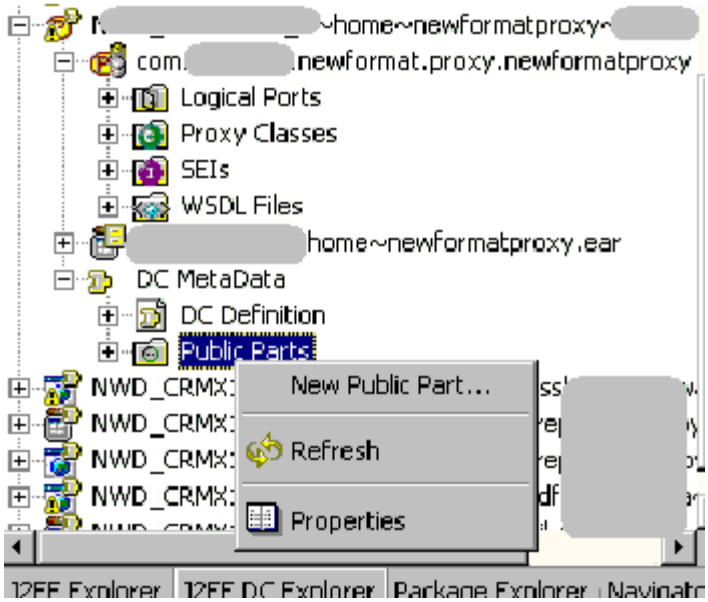


Using proxy in web project

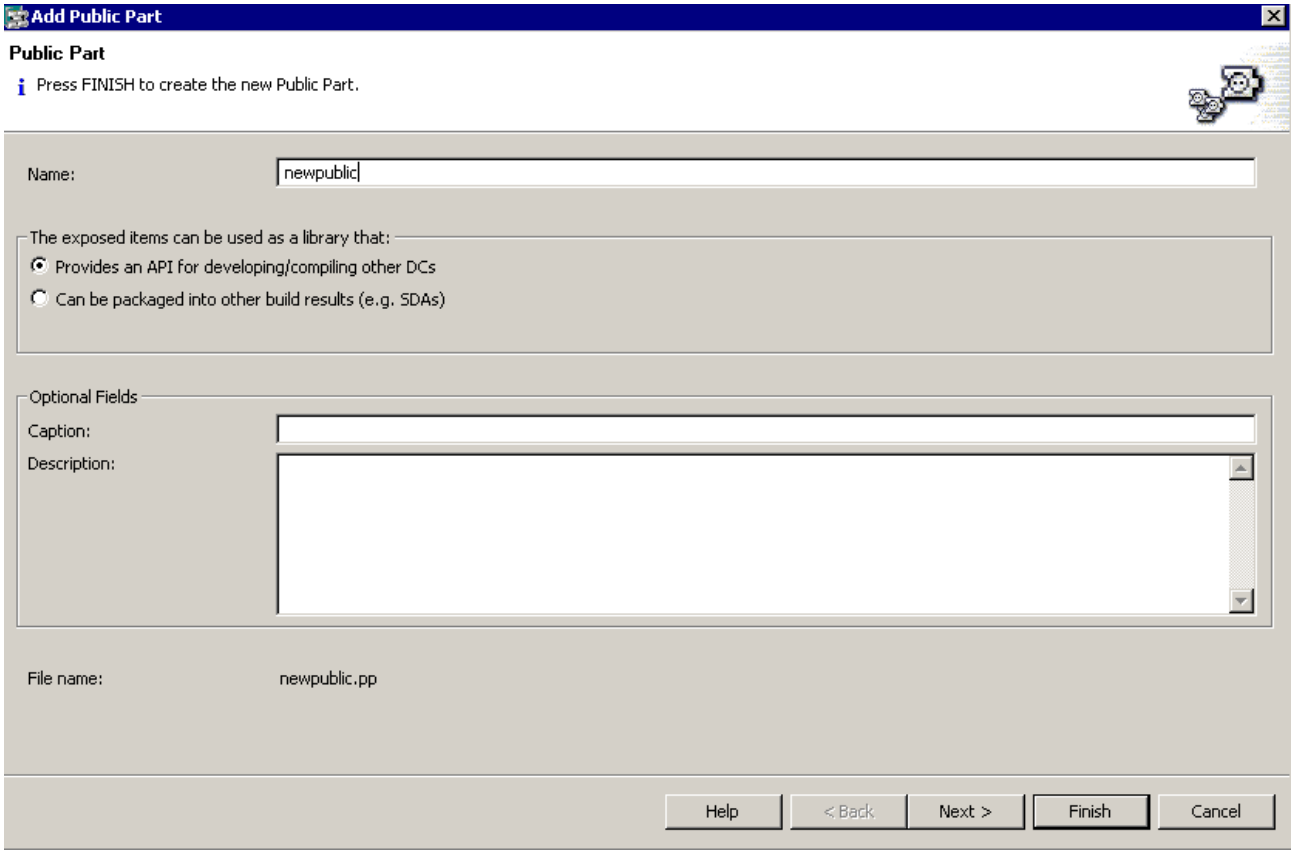
In the above steps you have successfully created and deployed a proxy definition on server. But it is still not available for other Development Component's. For this, you need to add this proxy to public part of web service Development Component containing this proxy.

Step 1 – Create Public Part in Web Service Development Component

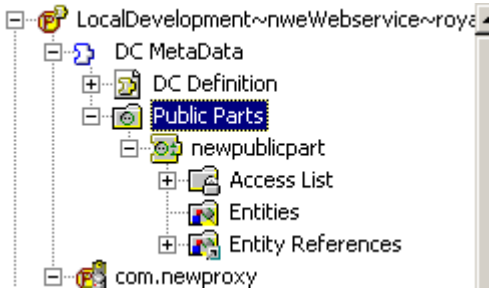
In your web service Development Component (newwebservice) right click on newwebservice -> DC Metadata -> PublicParts, and then click on "new public part".



Give name of public part and click Finish.

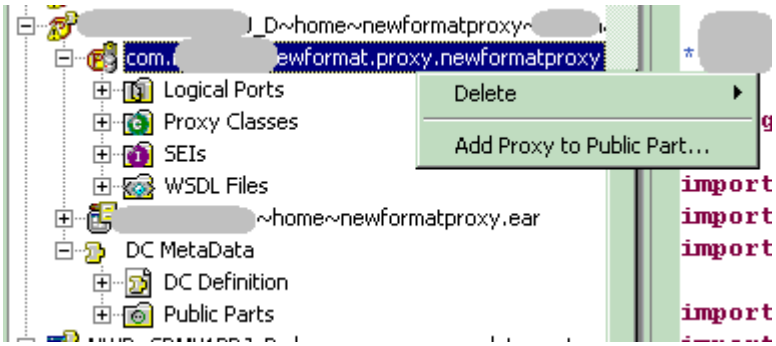


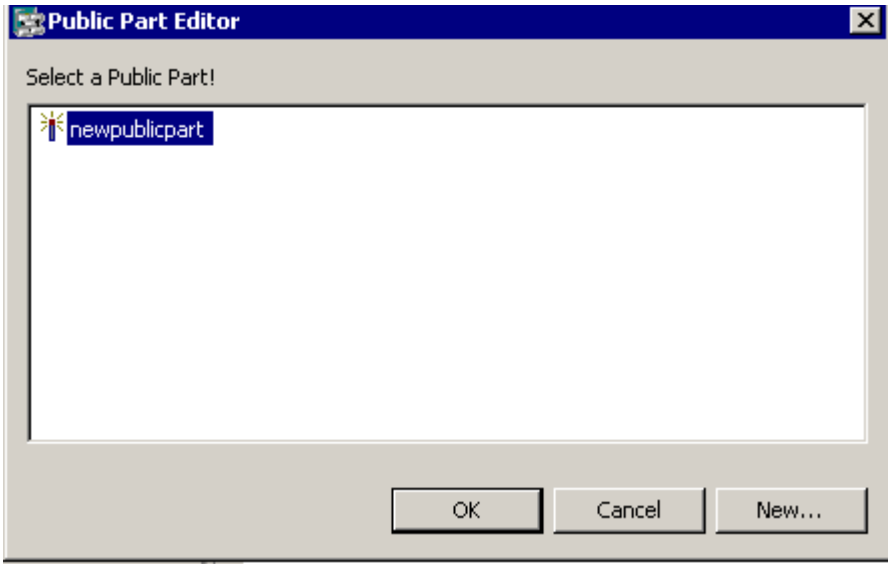
Public part created is now visible.



Step 2 – Add Proxy to this Public Part

Now add your proxy (newProxy) to this public part. Right click on proxy name and click on “Add Proxy to Public Part”.





Click on Ok.

Proxy will be added to public part. Check your proxy in Public Part -> Entities.

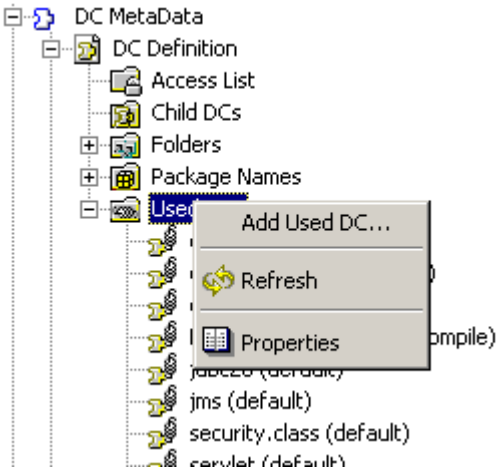
Using Publicly Accessible Proxy in Web Project

In the above steps you have successfully made your proxy available to be used by web component Development Component. In this section we will use this proxy definition (newProxy) in our WebComponent (UsingWebComp_WA).

It is assumed that web components and its EAR are already available in your development environment.

Step 1 – Create Reference of Proxy in Used Development Component Section of Web Development Component

Right click on UsingWebComp_WA -> DC Metadata -> DC Definition -> Used DC and select “Add used DC”

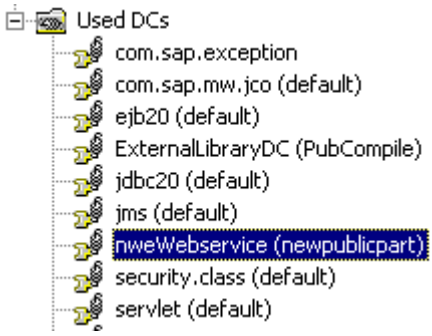


Now in dialogue box opened, select your Software Component -> web service DC (newWebService) -> Proxy (newProxy) -> DC Metadata -> Public parts -> Public Part defined (NewPublic).

Error! Objects cannot be created from editing field codes.

Select “Build Time” and click on Finish.

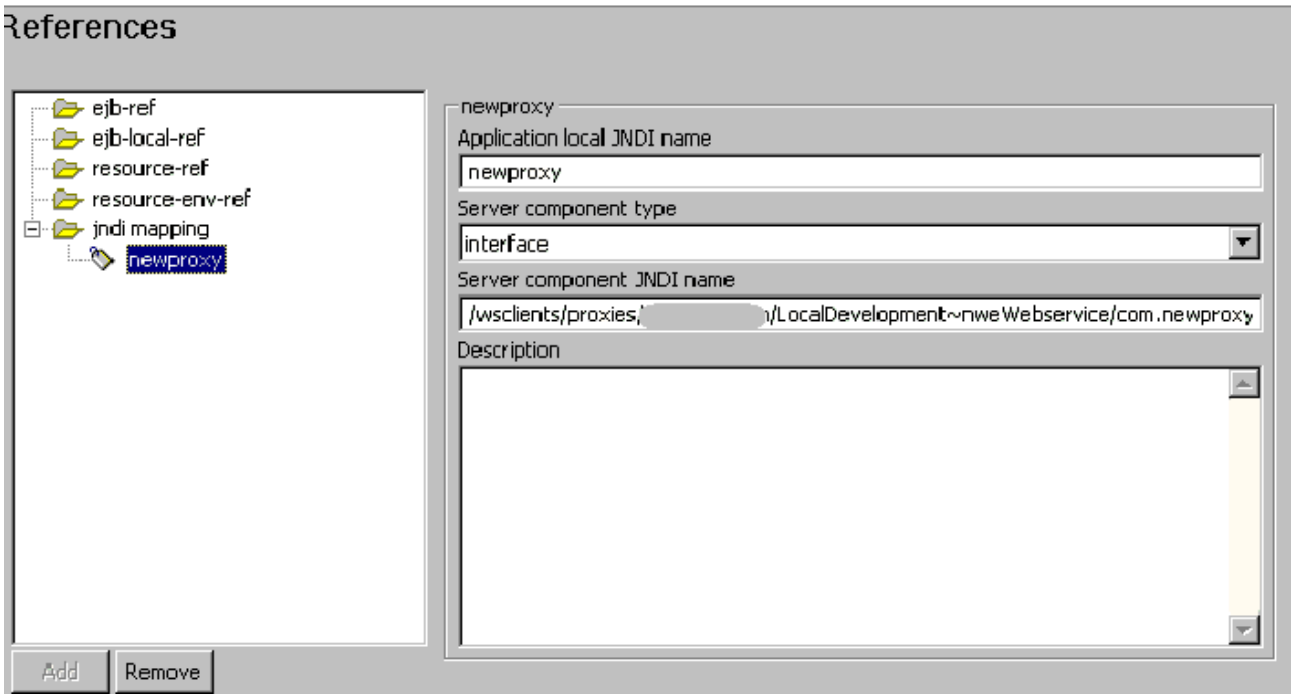
Newly created proxy is now available for use in your Web Component Development Component (UsingWebComp_WA).



Step 2 – Add Reference of Proxy in Web-j2ee-engine.xml

Navigate to web-j2ee-engine.xml in your web component Development Component (UsingWebComp_WA).

Click on References tab and add a reference to this proxy under Jndi-mapping.



Click on jndi mapping and click on Add.

Enter following information:

Application Local JNDI name: newProxy (Any name can be given here. Just remember that this name will be used in your code to lookup for proxy)

Server component type: Select Interface from drop down.

Server component JNDI name:

Following convention is used

/wsclients/proxies/<ProviderName>/< Web service DC name>/<Package containing proxy>.<Proxy name>

In our case we will define following mapping

/wsclients/proxies/vendor.com/home~newWebService/com.newProxy

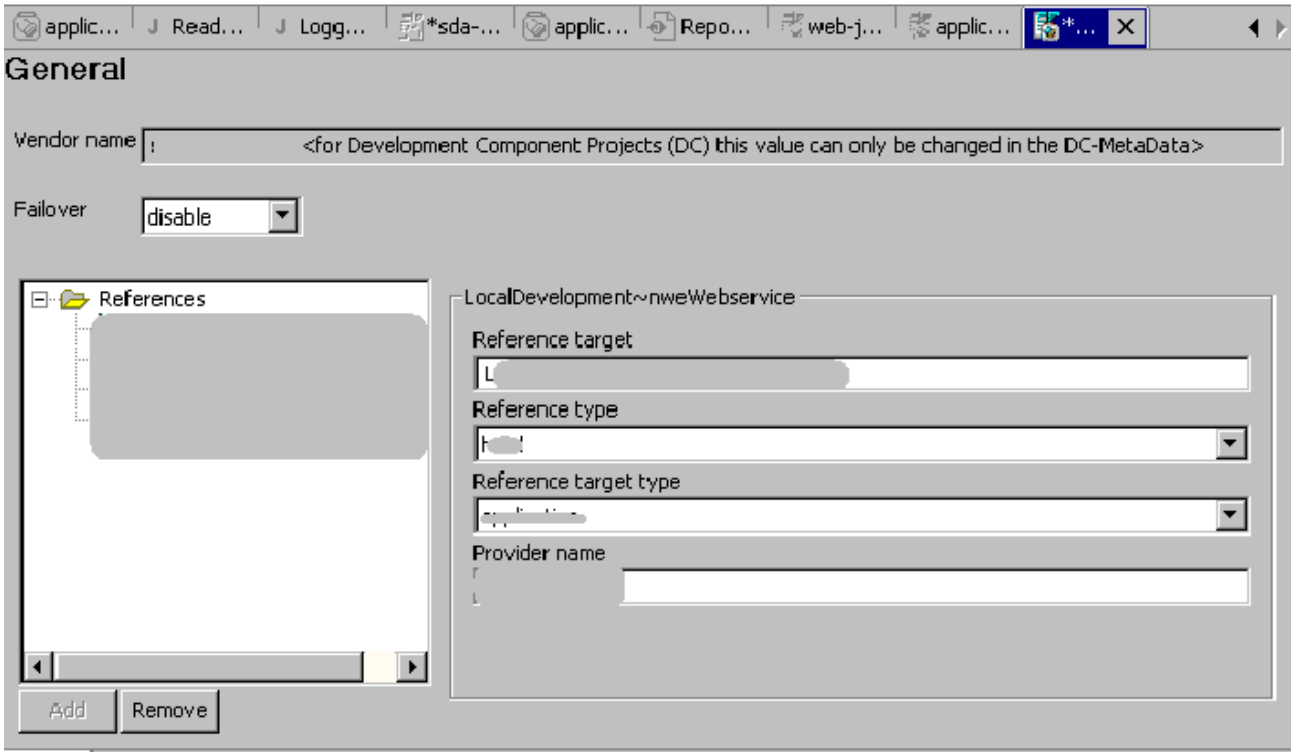
Configuration in EAR Development Component of web Development Component

In the EAR deployable unit Development Component (UsingWebComp_EA) of web Development Component (UsingWebComp_WA), you need to make reference of used Web service Development Component (newWebService) in application-j2ee-engine.xml

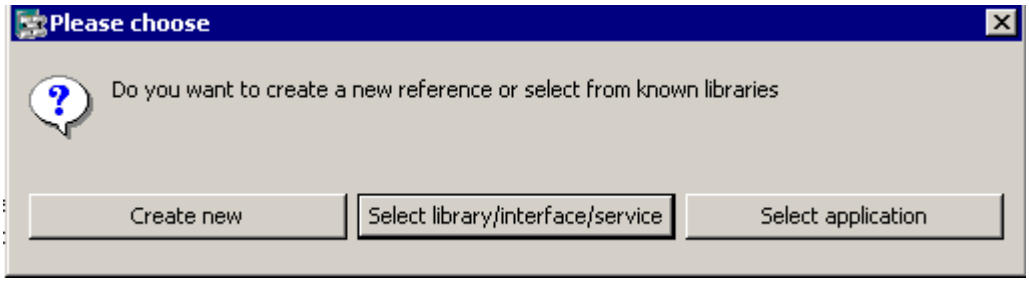
Step 1 – Add Reference of Proxy in Application-j2ee-engine.xml

Navigate to Application-j2ee-engine.xml in your EAR component Development Component (UsingWebComp_EA).

Click on references and click on “Add”.



Following dialogue box will be displayed



Click on “Create new”

Fill the fields with following values:

Reference target: Web service proxy Development Component name (newWebService)

Reference type: Hard

Reference target type: Application

Provider name: vendor name.com

Save xml file.

Step 2 – Built and Deploy EAR Development Component

Built and deploy EAR Development Component project (UsingWebComp_EA).

Configuration in J2EE Visual Admin tool

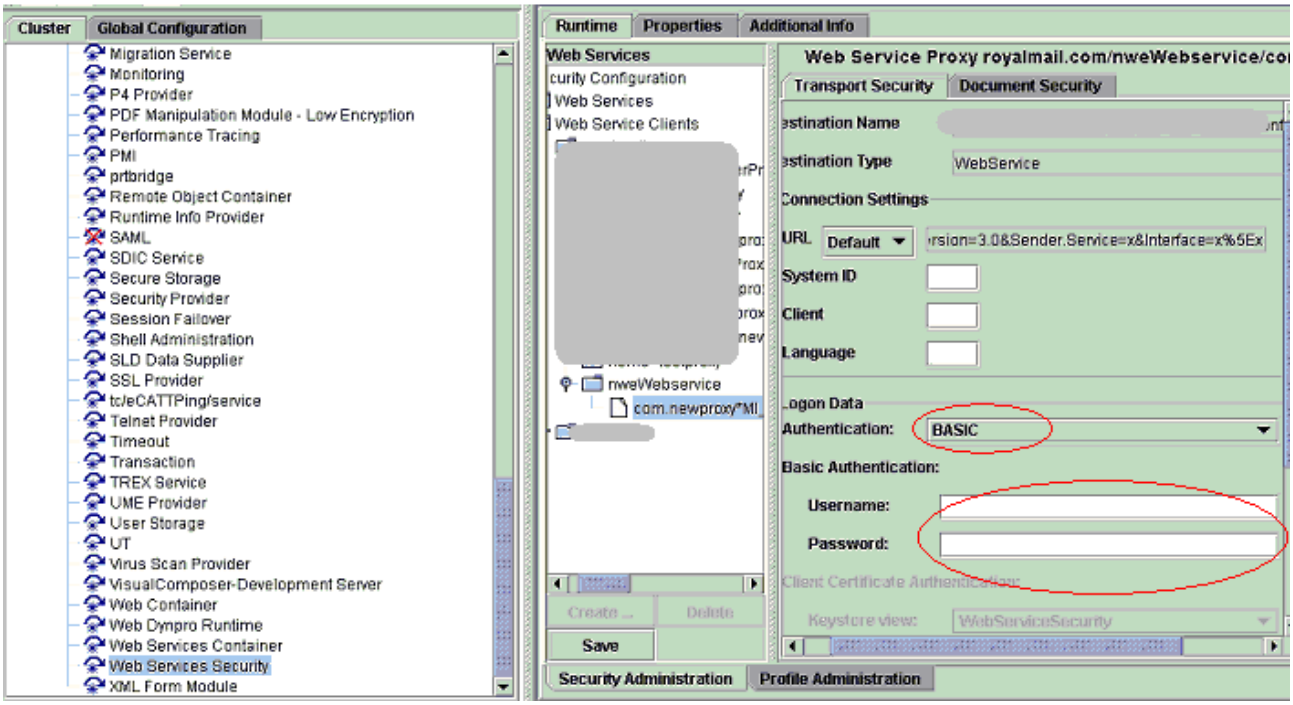
Open your j2ee visual admin tool for required configuration.

Navigate to Cluster -> <server Name> -> server -> services -> Web service security.

Now on right panel of visual admin tool navigate to

Runtime -> Web service clients -> vendor name -> Web service DC (newWebService) -> proxy name (newProxy)

Enter the username and password for authentication if required by your WSDL.



Java Code to Call Proxy

```
InitialContext ctx = new InitialContext();
ClassAService obj = (ClassAService)ctx.lookup("java:comp/env/proxyName");
ClassA port = (ClassA)obj.getLogicalPort("ClassBPort", ClassA.class);
ClassC_Res result = port.classA(input);
```

In above code snippet:

ClassA : You will find name of this class in your proxy -> SEIs

ClassAService : You will find name of this class in your proxy -> SEIs

ClassBPort : You will find name of this class in your proxy -> Logical Ports

ClassC_Res : You will find name of this class in your proxy -> Proxy Classes

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.