How to Consume an SAP Enterprise Service using Visual Composer and the Composite Application Framework

Applies to:

- mySAP 2005 ERP Enterprise Services Package ECC-SE 600 Add On SP2

- NetWeaver 2004s Java SP8/9

Summary

In this tutorial we will show how easy it is to consume a mySAP 2005 ERP Enterprise Service's using Visual Composer and the Composite Application Framework (CAF). Visual Composer will be used to model the user interface and CAF will be used as the Service Composition layer. In using CAF, we will compose a new web service which will add additional business logic to the existing Enterprise Service.

Author: Austin Chinn

Company: SAP

Created on: 9 October 2006

Author Bio

Austin Chinn is a Solution Architect in the SAP Market Development Engineering team. He is primarily focused on working with SAP Partners who are developing xApps.

Table of Contents

Introduction

In this example, we will be leveraging the Read Purchase Order Enterprise Service Operation which is contained within the Purchase Order Processing Process Component. In the first part of this exercise we will use CAF to compose a new, simplified web service based on the Read Purchase Order Enterprise Service. In the second part of this exercise, we will use Visual Composer to model a new user interface which will consume the newly composed and enhanced web service which consumes the Read Purchase Order Enterprise Service.

Section 1: Modeling the Service Composition Layer using CAF

The SAP Enterprise Service: "Read Purchase Order" returns the entire Purchase Order business object which is very rich. Because we want to simplify this service for our Business Process Expert (BPX), we will compose a new web service based on this existing Enterprise Service which only returns the fields that the BPX is interested in. This will simplify the modeling of the Visual Composer UI for the BPX. We will also add an additional field that the BPX requires which is not present in the Read Purchase Order Enterprise Service. This new field will be computed by adding new business logic during the service composition. To accomplish this, we will leverage the CAF Service Layer Modeling tool.

To start things off, we will create a new CAF project. After this, we will import the SAP Read Purchase Order Enterprise Service into the CAF project using the web service import wizard. Once this is complete, we will create a CAF Entity Service which will represent a customized view of the Purchase Order for the BPX. This customized Purchase Order view will be bound to the Read Purchase Order Enterprise Service using CAF Remote Persistence. Finally, we will generate a composite web service based on the PurchaseOrderApp CAF Application Service. This new CAF Application Service will add additional business logic to the Read Purchase Order Enterprise Service.

Step 1: Build the CAF project

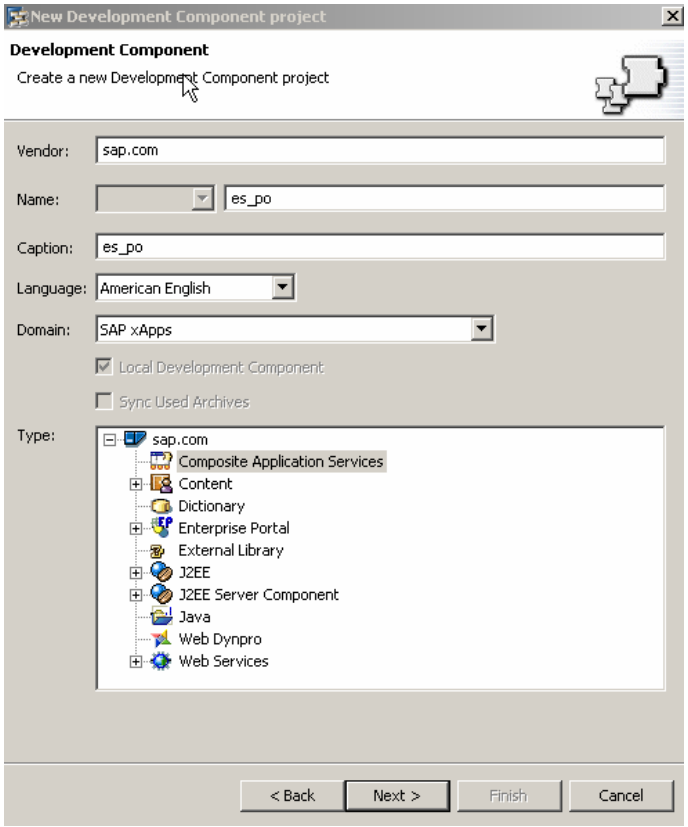
- Open the NetWeaver Developer Studio (NWDS)

- Choose File->New->Project->Development Component->Development Component Project and select the "Next" button

- Select either Local Development or NWDI based development and select "Next"

- Complete the development component information

- Choose "Next" and "Finish"

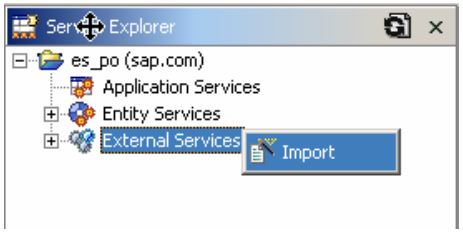


Step 2: Import the SAP Enterprise Service – "Read Purchase Order"

- In the CAF Service Explorer, open the "es_po" project

- Right-click on "External Services" and select "Import"

- Select "Web Service" and choose "Next"



- Select "Local File System or URL" and choose "Next"

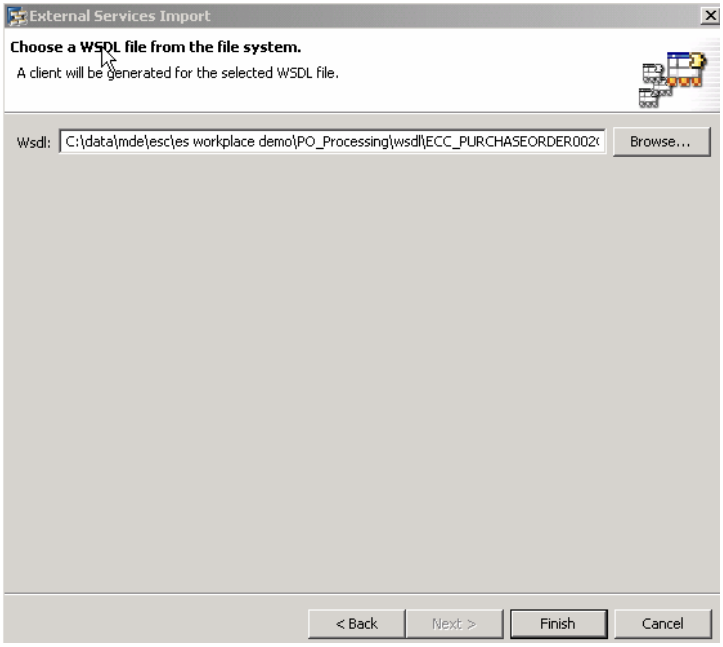
- Browse for the WSDL file locally and select it.

ECC_PURCHASEORDER002QR.xml

- Choose "Finish"

- Choose the "Save All Metadata" button 

Note: this WSDL file is obtained from mySAPERP2005 with the Enterprise Services ECC-SE 600 Add On SP2 using the WSAdmin transaction. If you don't have this installed, register for access to the ES-Workplace which provides a test implementation. (See Related Content at the end of this document for the registration link)

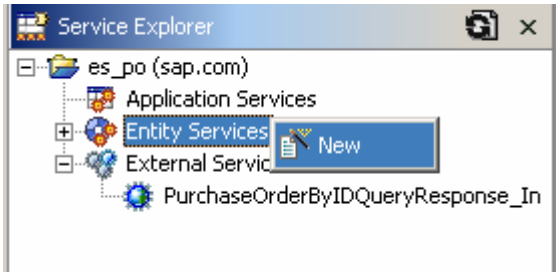


The Read Purchase Order has now been imported into the Composite Application Framework and it is now represented as an External Service. This External Service is a proxy to the Enterprise Service. In the next step, we will model the Purchase Order Entity Service which will contain the most important attributes (according to the BPX's business process) of the Purchase Order Enterprise Business Object.

Step 3: Model the Purchase Order Entity Service

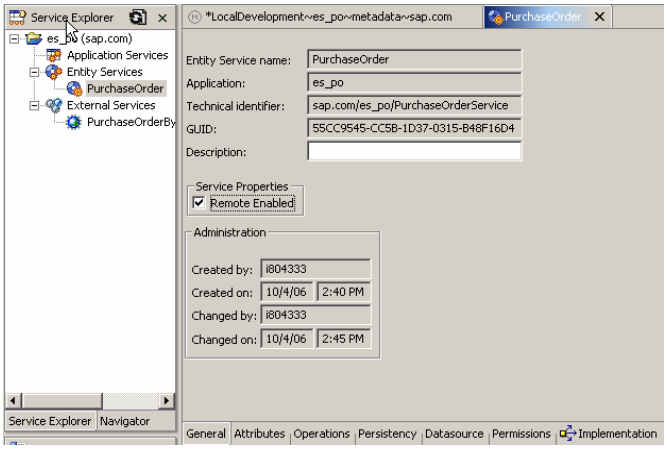
- In the CAF Service Explorer, right-click on Entity Services and select “New”

- Enter “PurchaseOrder” as the name and choose “Finish”



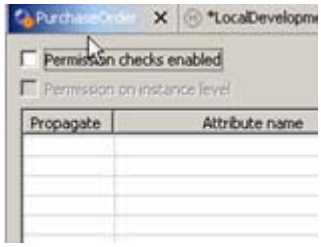
- Right-click on the PurchaseOrder Entity and choose “Edit

- Choose the “General” tab and check “Remote Enabled”



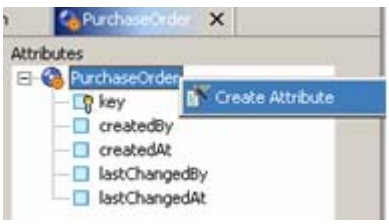
- Choose the “Permissions” tab

- Deselect the “permission checks enabled” checkbox. (For this example, we will not enforce Entity level permissions)



- Select the Attributes tab

- Right-click on PurchaseOrder and choose “Create Attribute”



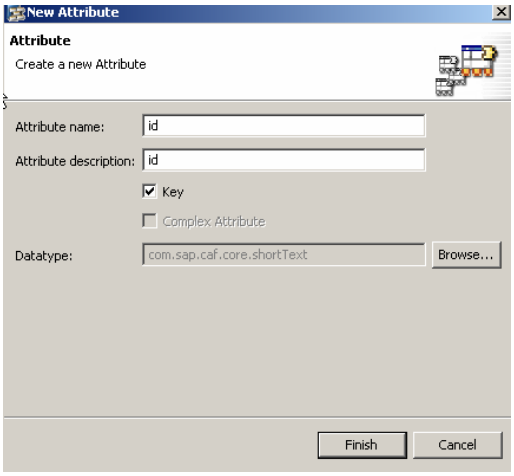
Enter the following values:

- name = id

- description = id

- check “Key”

- datatype = com.sap.caf.core -> shortText

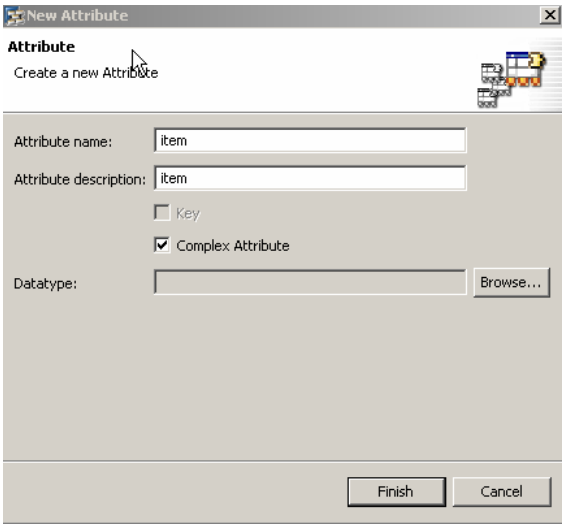


- Choose “Finish”

- Create Attributes for all of the remaining PO Header Attributes:

Name	Description	Datatype (com.sap.caf.core)
purchaseOrderDate	purchaseOrderDate	com.sap.caf.core. timestamp
creationUserAccountId	creationUserAccountId	com.sap.caf.core. shortText
currencyCode	currencyCode	com.sap.caf.core. shortText
price	price	com.sap.caf.core. currency
approvalRequired	approvalRequired	com.sap.caf.core.boolean

Since each Purchase Order can have many Items, we need to create a nested Complex Attribute. (Note, item could be modeled as an Entity Service with a relationship to Purchase Order, but for simplicity we are using a nested complex attribute)



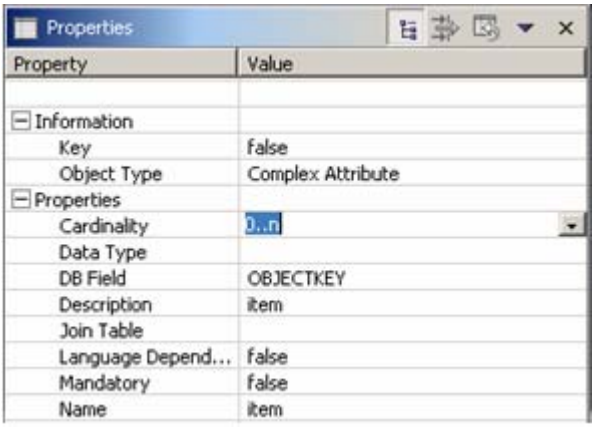
- Right-click on the Purchase Order and select “Create Attribute”

- Enter the values shown and select “Finish”

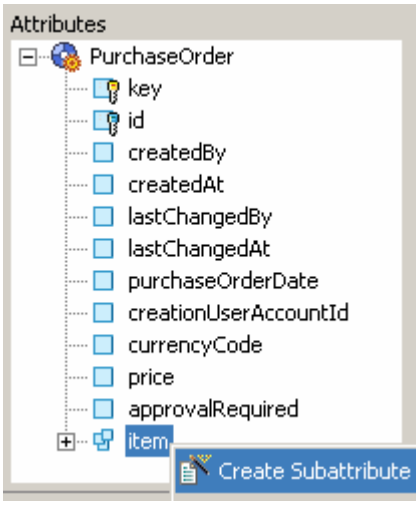
- Click on the new “item” complex attribute and select the “Properties” tab.

- Change the “Cardinality” to “0..n”

Each PurchaseOrder can have 0 or more Items.



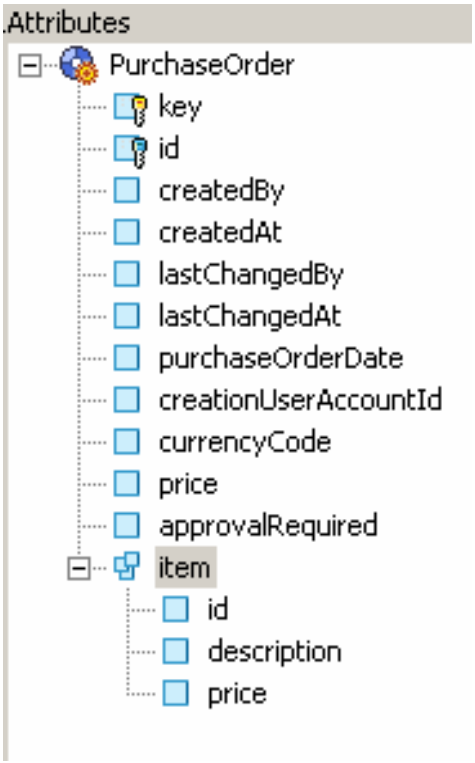
- Right-click on the “item” and select “Create Subattribute”



- Create these Subattributes for “item”:

Name	Description	Datatype
Id	Id	com.sap.caf.core.shortText
description	description	com.sap.caf.core.longText
price	price	com.sap.caf.core.currency

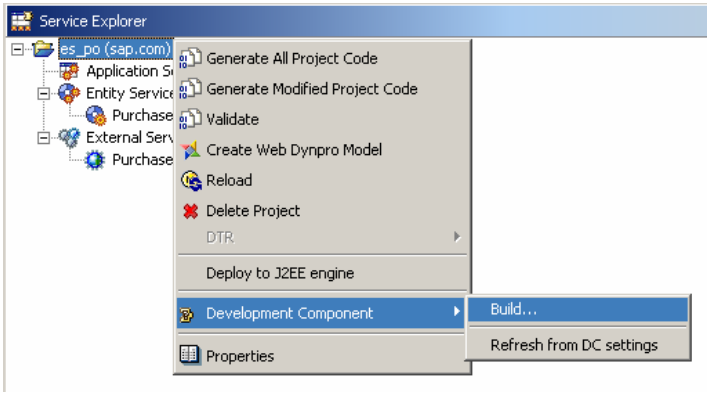
The final structure of the PurchaseOrder should look like this:



- Choose the “Save All Metadata” button 

- Right-click on the project and select Development Component -> Build

- Select “Generate Modified Project Code” if prompted.



We have completed the modeling for our PurchaseOrder Entity Service which contains only the attributes that the BPX cares about (for this scenario). The next step is to map the PurchaseOrder Entity Service operations and attributes to the Read Purchase Order Enterprise Service.

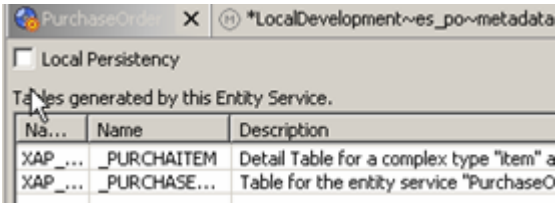
Step 4: Map the SAP Enterprise Service to the new CAF Entity Service Operation

We will use the CAF Remote Persistency functionality to map the Enterprise Service to our new Read Purchase Order Entity Service. To do this, we have enable remote persistency and then create the persistency mappings.

- Select the “Persistency” tab

- De-select the “Local Persistency” checkbox

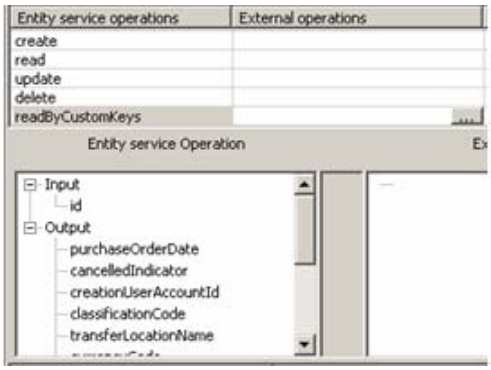
- Choose the “Save All Metadata” button 



- Select the “Datasource” tab

- Select the “readByCustomKeys” Entity Service Operation

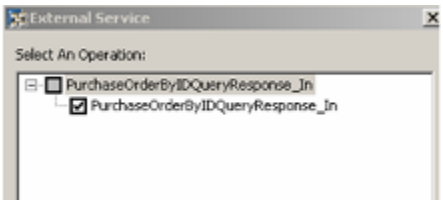
- Select the External Operations “ ... ” button



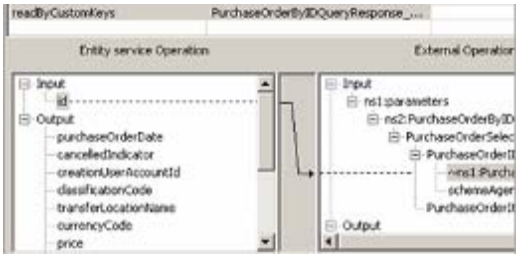
Note: CAF Supports both local and remote persistence of Entity data. If the Entity data is stored locally, CAF will automatically create a guid for it. Since in this case, the Entity data is sourced from an Enterprise Service, we are not using the default “read” operation because this accepts a CAF guid. This is why we are using the “readByCustomKeys” method because here you can define your own custom primary key.

- Choose “PurchaseOrderByIDQueryResponse_In”. This is the technical name for the Read Purchase Order Enterprise Service.

- Select OK



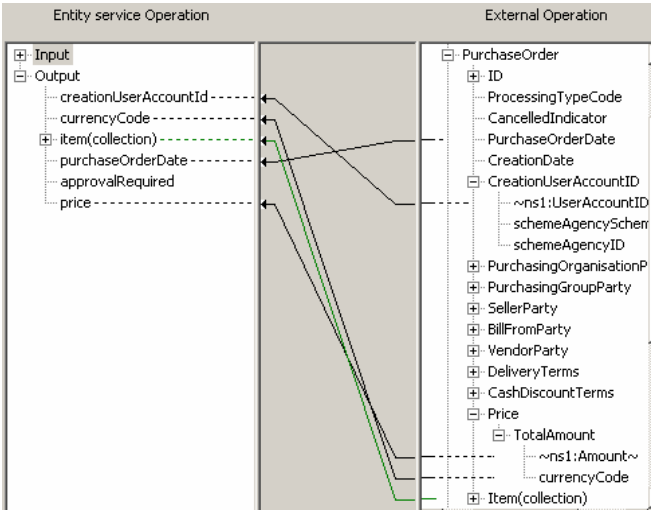
- Drag the Input “id” attribute from the Entity Service Operation and drop it on the “~ns1:PurchaseOrderID~” External Operation attribute. This create the first attribute mapping.



- Repeat the attribute mappings for the following:

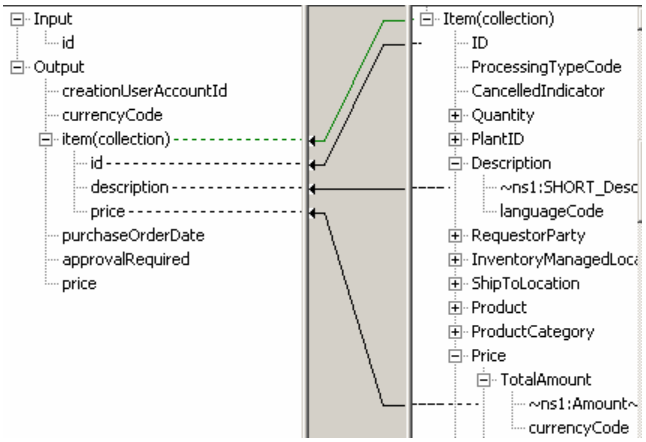
Entity Service Attribute	External Operation Attribute
purchaseOrderDate	PurchaseOrderDate
creationUserAccountId	CreationUserAccountID -> ~ns1:UserAccountID~
currencyCode	Price -> TotalAmount -> currencyCode
price	Price -> TotalAmount -> ~ns1:Amount~
item -> id	Item -> ID
item -> description	Item -> Descrption -> ~ns1:SHORT_Description~
item -> price	Item -> Price -> TotalAmount -> ~ns1:Amount~

- The mappings for the PurchaseOrder header attributes should now look like:



Note: we are not mapping the “approvalRequired” field because this value will be computed by business rules that we will add later.

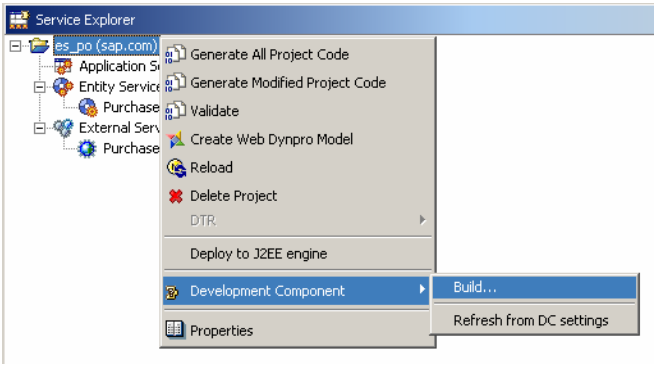
- The mapping for the PurchaseOrder->item attributes should look like this:



- Choose the “Save All Metadata” button 

- Right-click on the project and select Development Component -> Build

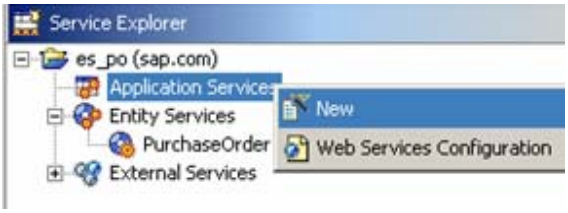
- Select “Generate Modified Project Code” if prompted.



We are now finished modeling the PurchaseOrder CAF Entity Service. The next step is compose a new “Composite Service” which will expose the PurchaseOrder Entity and add new business logic that does not exist in the Read Purchase Order Enterprise Service. This new business logic is required by the BPX for this scenario. We will build this new “Composite Service” using a CAF Application Service. Once we have completed building this composite service, we will expose it as a web service so that it can be consumed from by Visual Composer.

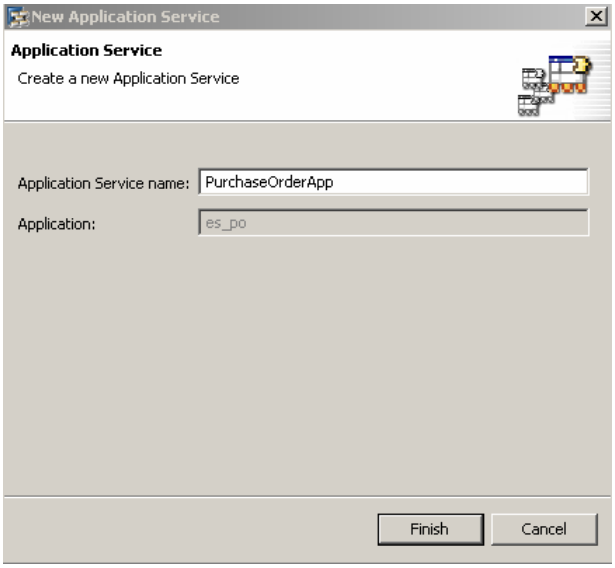
Step 5: Create the CAF Purchase Order Application Service

- Right click on “Application Services” and select “New”

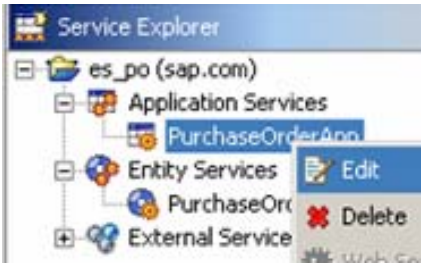


- Enter “PurchaseOrderApp”

- Select “Finish”



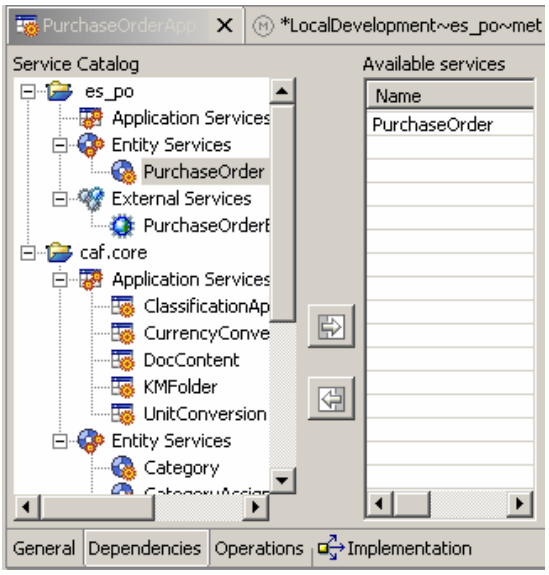
- Right-click on “PurchaseOrderApp” in the Service Explorer and select “Edit”



- Select the “Dependencies” tab

- Select the “Purchase Order” Entity Service in the Service Catalog

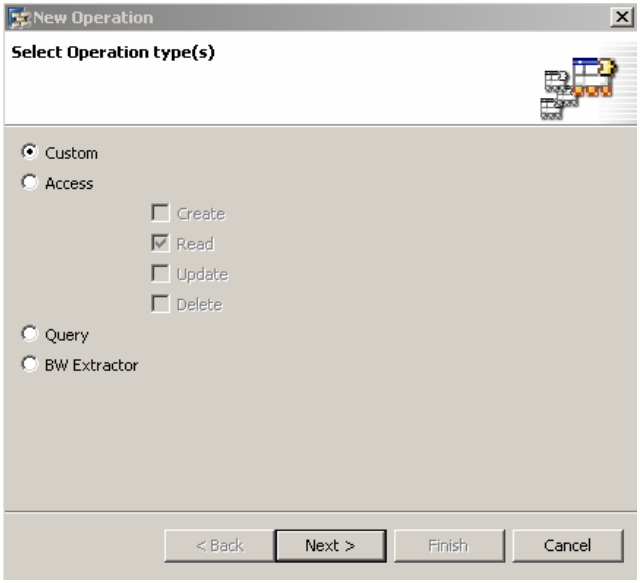
- Select  to add the Purchase Order to the available services



- Select the “Operations” tab

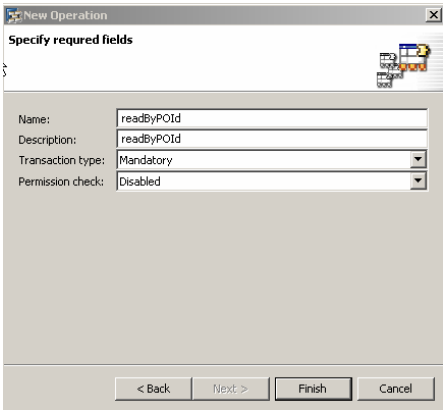
- Select the “Add” button

- Choose “Custom” and select “Next”



- Enter “readByPOId” for the name and description

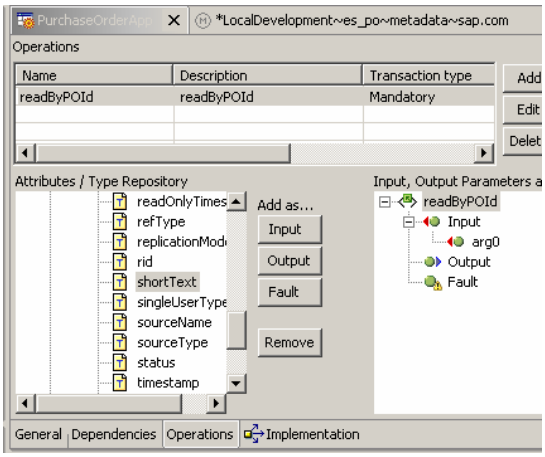
- Choose “Finish”



- Choose the Simple Types -> com.sap.caf.core -> Short Text

- Select the “Input” button

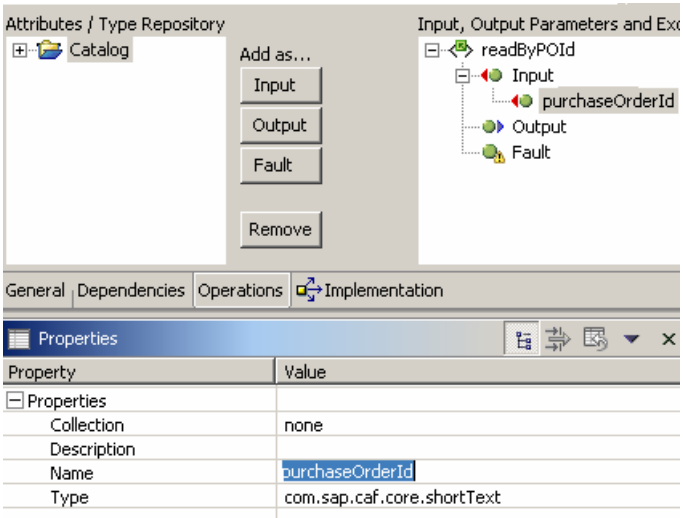
Now we have added one input parameter (ShortText) to our new Operations. We need to give this parameter a meaningful name.



- Select the “arg0” input parameter

- Change it’s name to “purchaseOrderId”

Next, we will create the output parameter for our new operation.

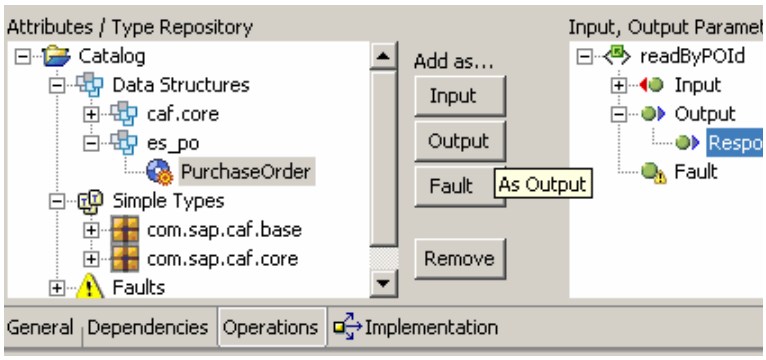


- Select “PurchaseOrder” from the Catalog

- Select the “Output” button

This will add the PurchaseOrder Entity as an output parameter.

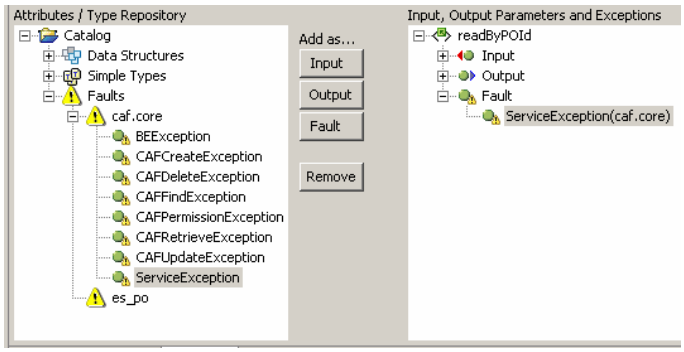
Finally, we have to define the Exception for our operation.



- Select “ServiceException” in the Catalog

- Select the “Fault” button

Now that we have defined the signature for the readByPOId method, we can implement the business logic.



- Select the “Implementation” tab

- Select the “readByPOId” method in the Outline View

- Enter the following code after the “retValue = null;” line

```
retValue =
this.getPurchaseOrderService().readByCustomKeys(purchaseOrderId);

float price = Float.parseFloat(retValue.getPrice());

retValue.setApprovalRequired( price >= 100 ) ;
```


The code should now look like this:

Now we have added a business rule that sets the “approvalRequired” flag if the “price” is greater than or equal to 100.

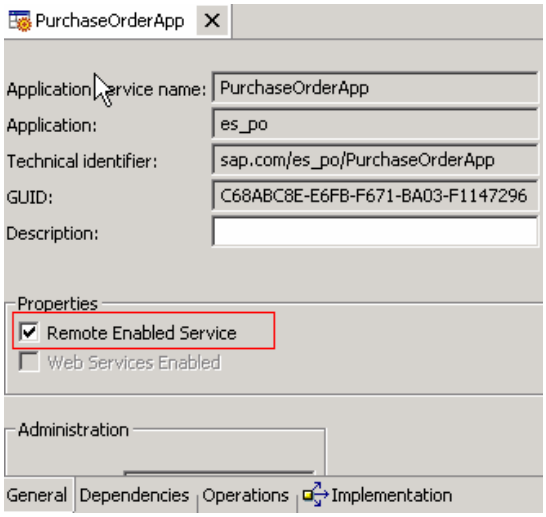
```
    public com.sap.es_po.besrv.purchaseorder.PurchaseOrder readByPOId(java.lang.String p
        // logging
        java.lang.String CAF_user = sessionContext.getCallerPrincipal().getName();
        java.lang.String CAF_methodHeader = PurchaseOrderAppBean.JARM_REQUEST + ":" + "r
        Object[] CAF_parameters = new Object[] {purchaseOrderId};
        com.sap.caf.rt.util.CAFPublicLogger.entering(CAF_user, PurchaseOrderAppBean.JARM

        com.sap.es_po.besrv.purchaseorder.PurchaseOrder retValue;
        try {
//@@custom code start - readByPOId(java.lang.String)
            retValue = null;
            retValue = this.getPurchaseOrderService().readByCustomKeys(purchaseOrderId);
            float price = Float.parseFloat(retValue.getPrice());
            retValue.setApprovalRequired( price >= 100 ) ;

//@@custom code end - readByPOId(java.lang.String)
            return retValue;
        } finally {
            com.sap.caf.rt.util.CAFPublicLogger.exiting(CAF_user, PurchaseOrderAppBean.J
        }
    }
```

The newly composed composite service “PurchaseOrderApp” is almost complete. The next step is to expose it as a Web Service so that the BPX can consume this from Visual Composer.

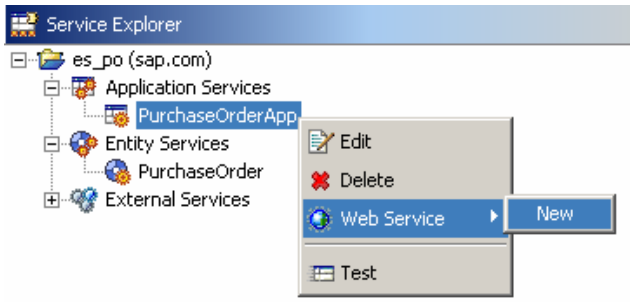
Step 6: Expose the PurchaseOrderApp Application Service as a Web Service and deploy

- Select the “General” tab on the PurchaseOrderApp

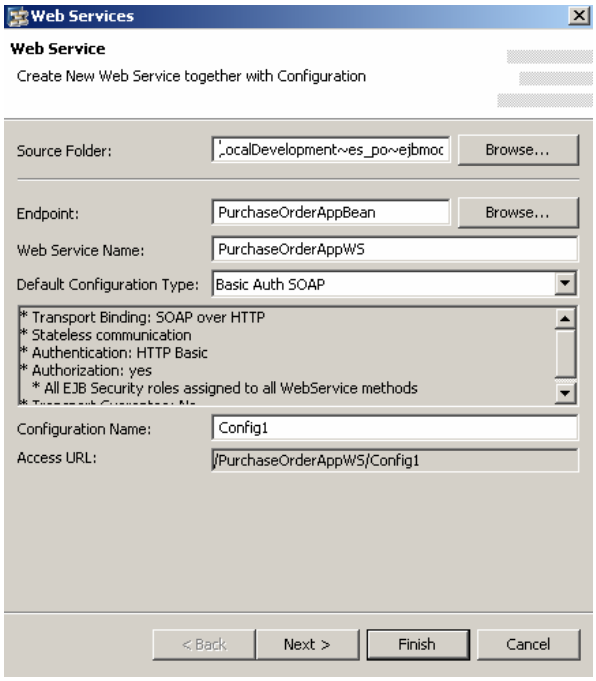
- Select the “Remote Enabled Service” checkbox



- Right-click on the PurchaseOrderApp Application Service and select “Web Service -> New”



- Leave all default values and select “Finish”

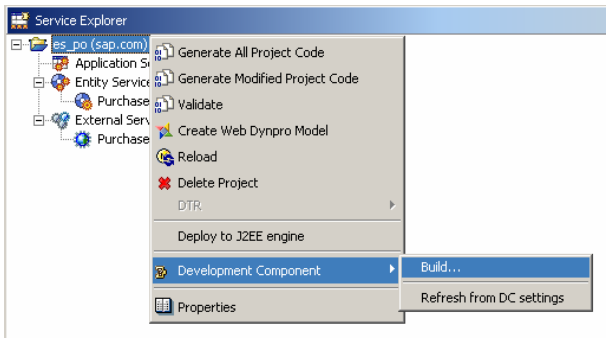


The Web Service has now been generated. Next we will build the project and deploy it to the J2EE Engine.

- Choose the “Save All Metadata” button 

-Right-click on the project and select Development Component -> Build

- Select “Generate Modified Project Code” if prompted.



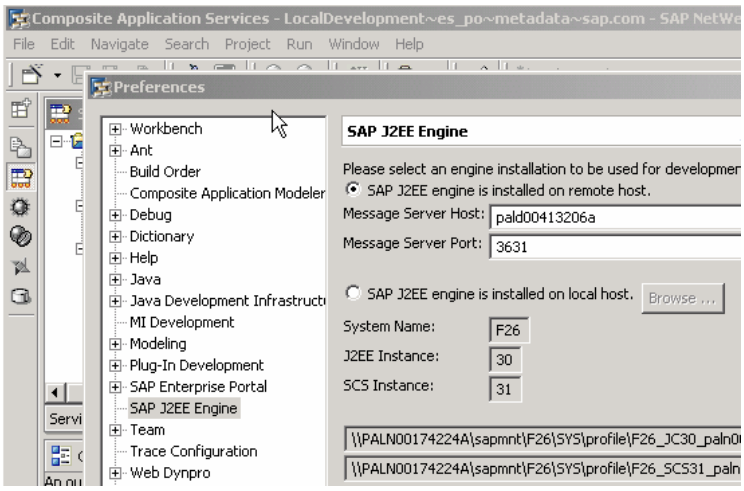
Configure the J2EE Engine in the NWDS

- Select “Window -> Preferences ->SAP J2EE Engine”

- If the J2EE Engine is Remote, then enter the Message Server Host and Port.

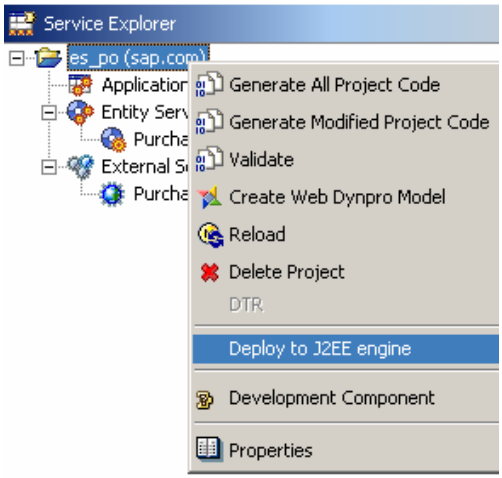
- If the J2EE Engine is local, then click “Browse” and select the local J2EE Engine.

- Select “OK”



Note: If the local J2EE Engine does not show up when you select the “Browse” button, it may need to be started. If it still does not show up after starting, then try configuring the local J2EE Engine as a Remote Engine.

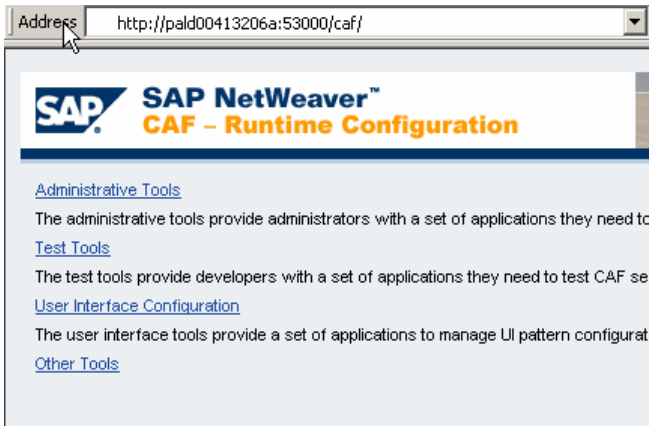
- Right-click on the project and select “Deploy to J2EE Engine”



The CAF project has now been deployed to the J2EE Engine. In order to complete the implementation of our new web service, we have to configure the Enterprise Service endpoint and test it.

Step 7: Configure Enterprise Service end points and test

- Open your web browser and point to the CAF – Runtime Configuration on your J2EE Engine at http://host:port/caf (example: http://xappserver:53000/caf)



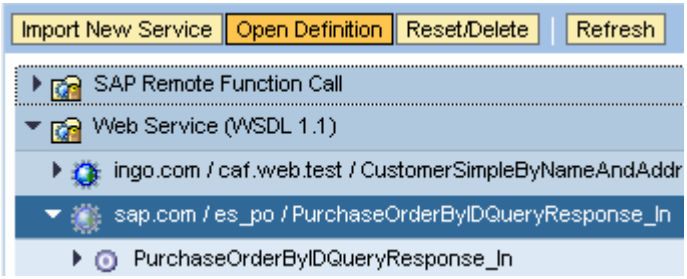
- Select “Administrative Tools”

- Select “External Service Configuration”

- Select “Service Registry”

- Select the “PurchaseOrderByIDQueryResponse_In” Enterprise Service

- Select “Open Definition”



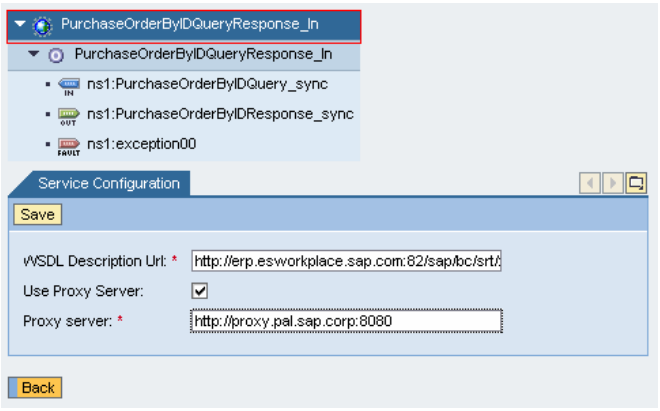
- Select the top (Service) node of the structure

- Enter the WSDL Description URL (example):

http://erp.esworkplace.sap.com:82/sap/bc/srt/xip/sap/ECC_PURCHASEORDER002QR?sap-client=800&wsdl=1.1

- Check proxy server and enter the proxy URL if you are behind a firewall

- Select “Save”



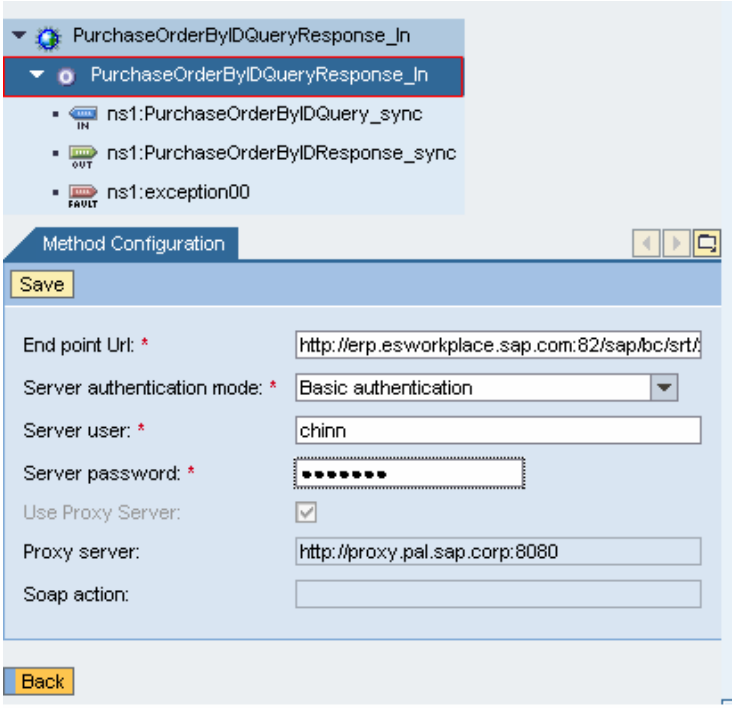
- Select the second (Method) node of the structure

- Enter the End Point URL (example):
http://erp.esworkplace.sap.com:82/sap/bc/srt/xip/sap/ECC_PURCHASEORDER002QR?sap-client=800

- Choose authentication mode = “Basic Authentication”

- Enter your user and password for the ES Workplace

- Select “Save”

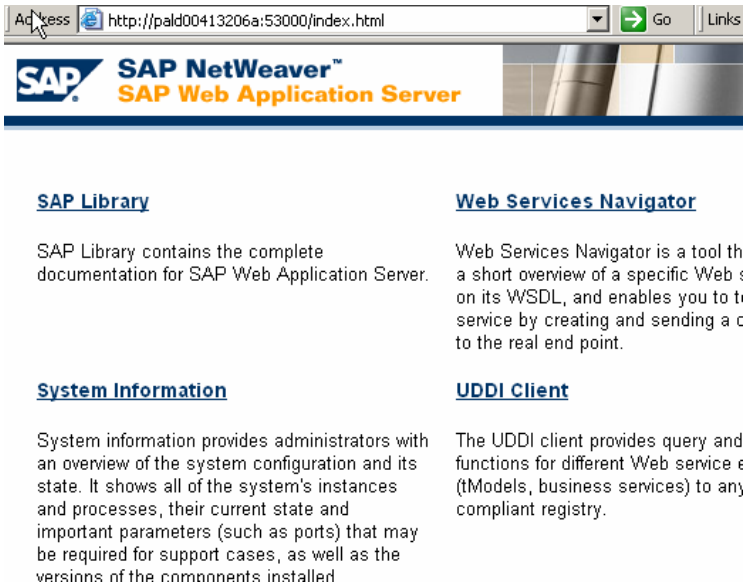


Next we will test the web service.

- Navigate to the J2EE homepage at http://j2eehost:port/

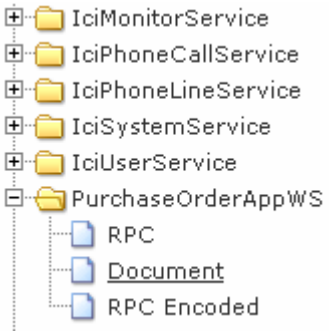
- Select “Web Services Navigator”

Note: if you are behind a firewall, the J2EE Engine proxy must be configured in order to test the web service.

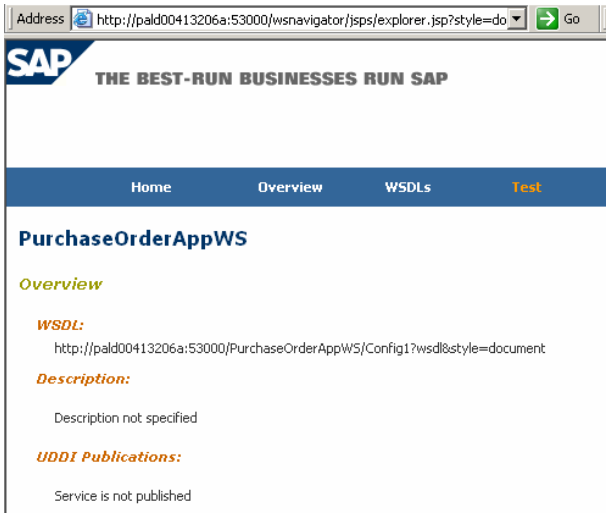


- Open the PurchaseOrderServiceAppWS and select “Document”

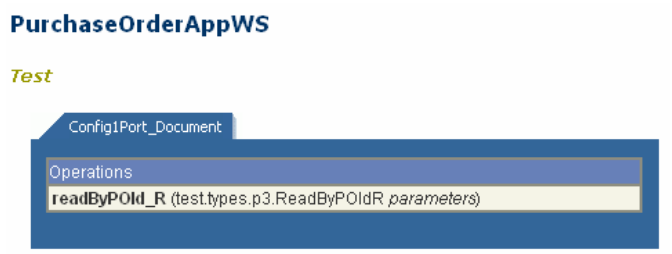
Note: you can also test the Enterprise Service here by entering its WSDL URL and selecting “Next”



- Select “Test”

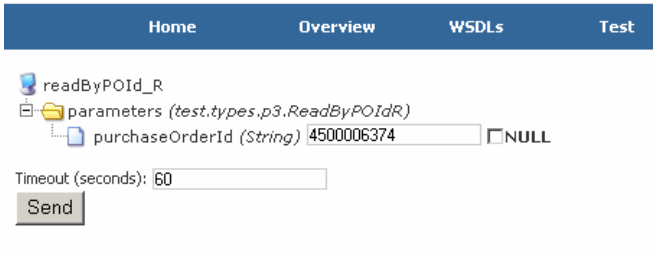


- Select the “readByPOId_R” operation

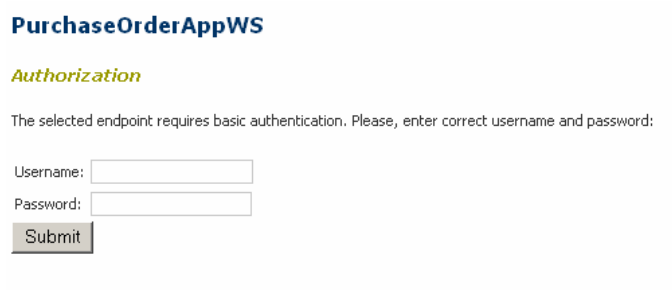


- Enter in a valid Purchase Order Id

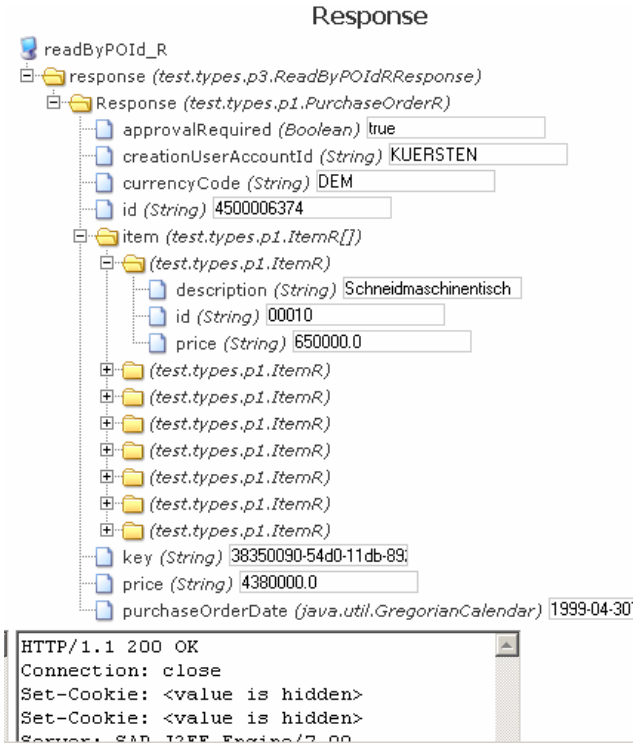
- Select “Send”



- Enter the username and password for the J2EE Engine Administrator



- View the Response message



The newly composed ReadPOById Composite Web Service has now been configured and tested. In the next section we will show how the BPX or developer can quickly model a rich user interface which consumes this.

Section 2: Modeling the User Interface with Visual Composer

Step 1: Configure J2EE Engine Web Service End Points

In order for Visual Composer to read the catalog of Web Services that have been deployed to the J2EE Engine, we need to configure two new J2EE Logical Destinations

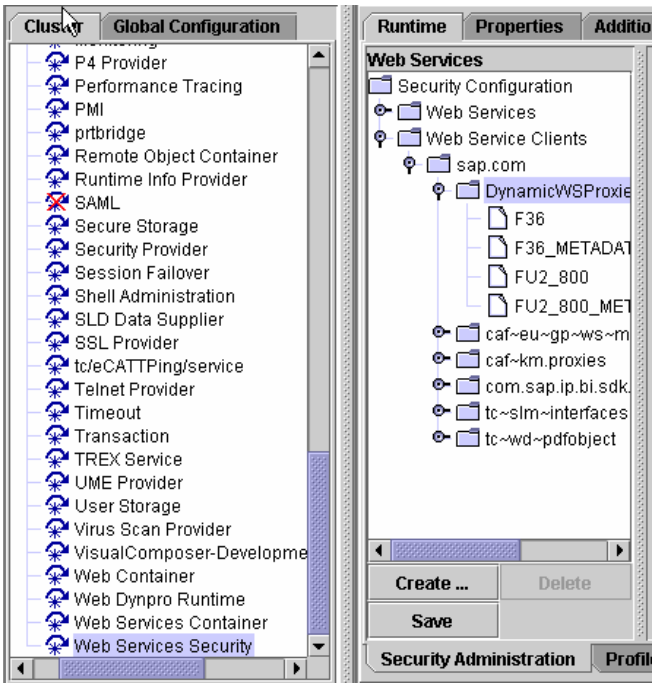
- Login to the J2EE Visual Administrator

- Navigate to Server->Services->Web Service Security

- Select the “Security Administration” tab

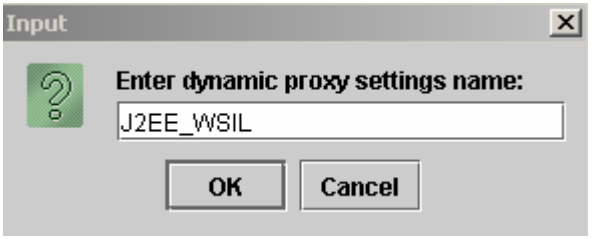
- Open the folders: Security Configuration -> Web Service Clients->sap.com -> DynamicWSProxies

- Select the ”Create” button



- Enter the Logical Destination for the J2EE WSIL address

- Select OK



This is the WSIL (Web Services Inspection Language) Logical Destination for the J2EE Engine. This endpoint will be used by Visual Composer to retrieve a list of all of the web services that have been deployed.

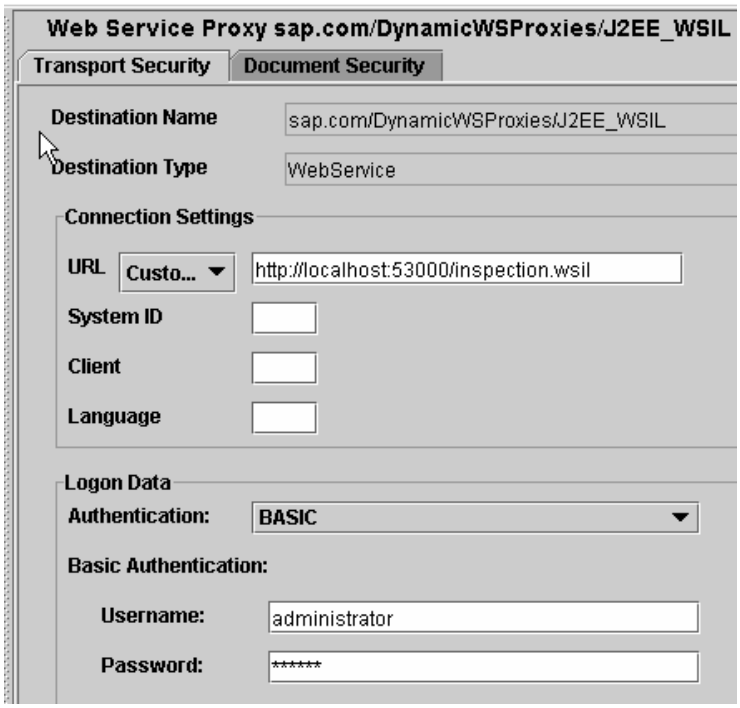
Enter the following information for the J2EE_WSIL logical destination:

- URL= http://j2ee_server:port/inspection.wsil

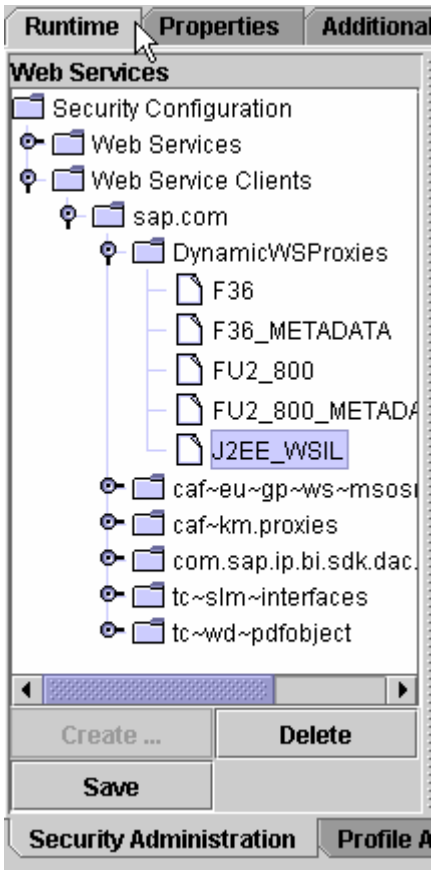
- Authentication = BASIC

- J2EE Administrator Username

- J2EE Administrator Password



- Select the “Save” button



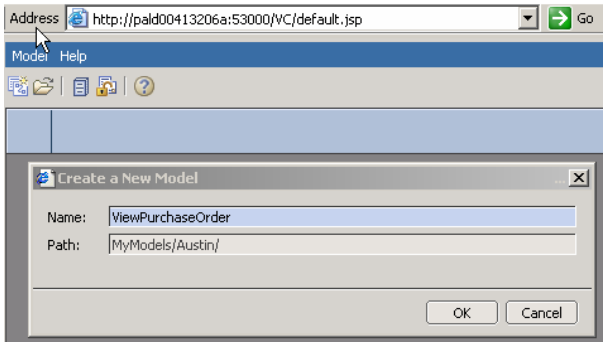
Step 2: Model the Visual Composer User Interface

- Open the Visual Composer URL: http://j2ee_host:port/VC

- Select Model -> New Model

- Enter “ViewPurchaseOrder” for the name

- Select “OK”



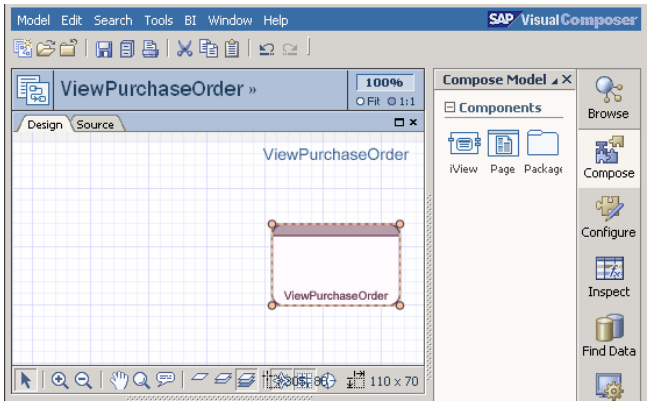
- Select 

- Drag and drop the 

component to the ViewPurchaseOrder model pane. (you may have to click on the model pane once to enable it before dragging and dropping)

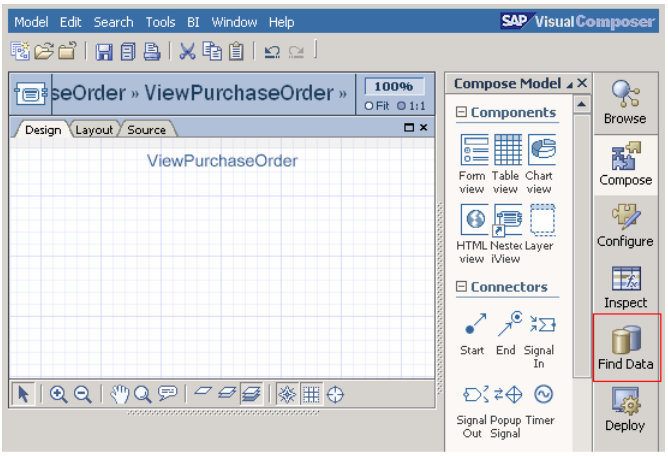
- Name the IView “ViewPurchaseOrder”

You can also rename the IView by right-clicking on it and selecting “Rename”



- Double click on the ViewPurchaseOrder IView to open it

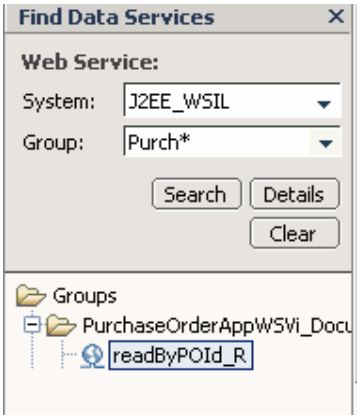
- Select the “FindData” button



- Enter System = J2EE_WSIL

- Enter Group = Purch*

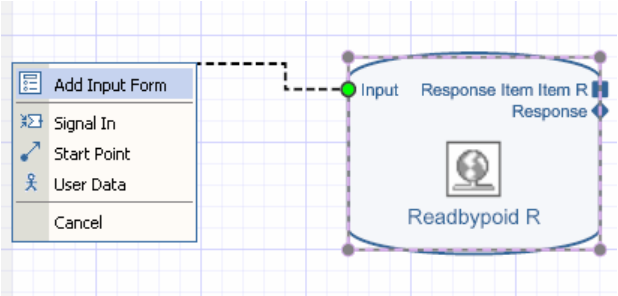
- Select the Search button



- Click on “Input” and drag it to the left and drop

- Select “Add Input Form”

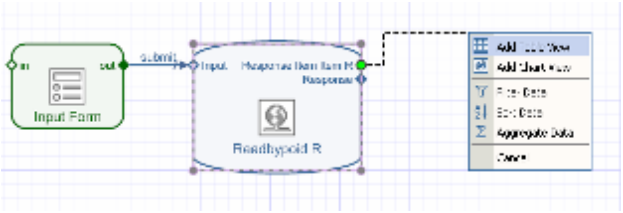
- Press Enter to accept the default name



- Click on the “Response Item Item R” element and drag drop to the right

- Select “Add Table View”

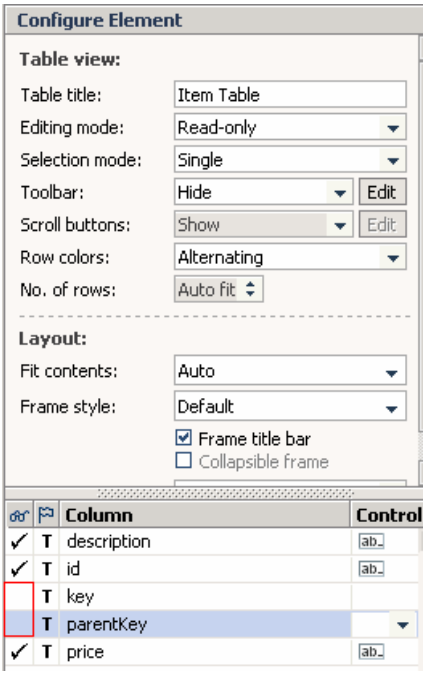
- Enter name = “Item Table”



- Right-click on the “Item Table” and select “Configure Element”

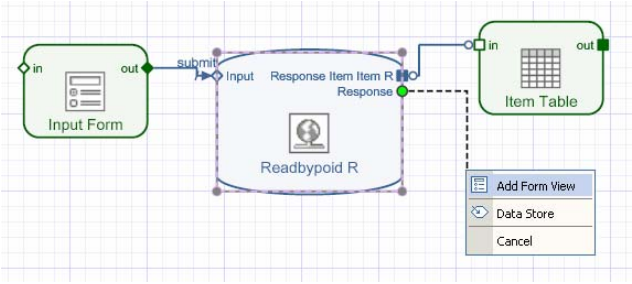
- De-select “key” and “parentKey” columns

This will hide these fields from the display form.



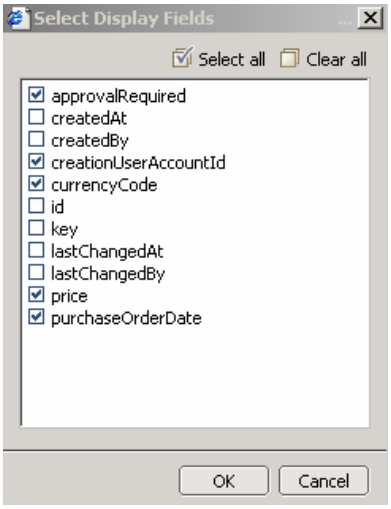
- Click on the “Response” element and drag drop to the right

- Select “Add Form View”



- Select the following fields from the pop-up:

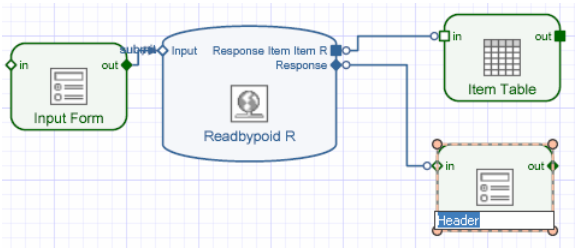
These fields will show up in the display form.



- Enter “Header” as the name

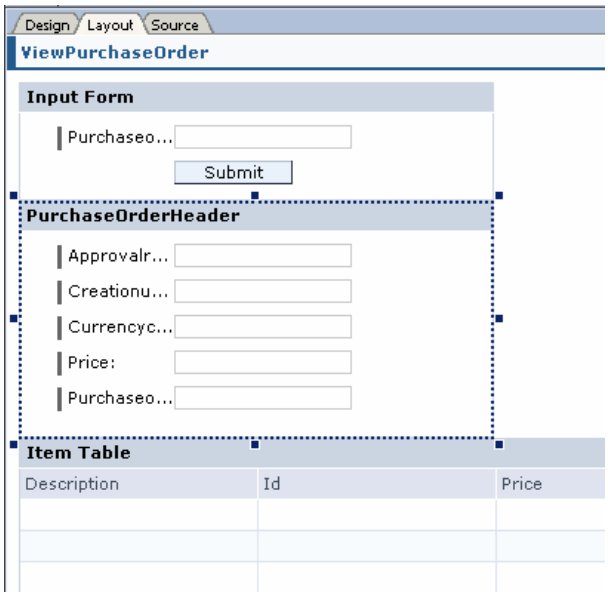
- Press Enter

We are now done with the basic modeling of the UI. Next we will put some minor touches on the UI.



- Select the “Layout” tab

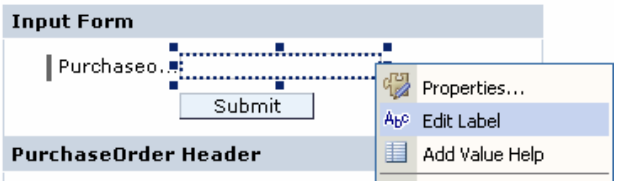
- Resize and drag/drop the UI Components so that they are structured like:



Next we will modify the labels.

- Right-click on the PurchaseOrderId column and select “Edit Label”

- Enter “Id” for the lablel



- Modify the remaining labels:

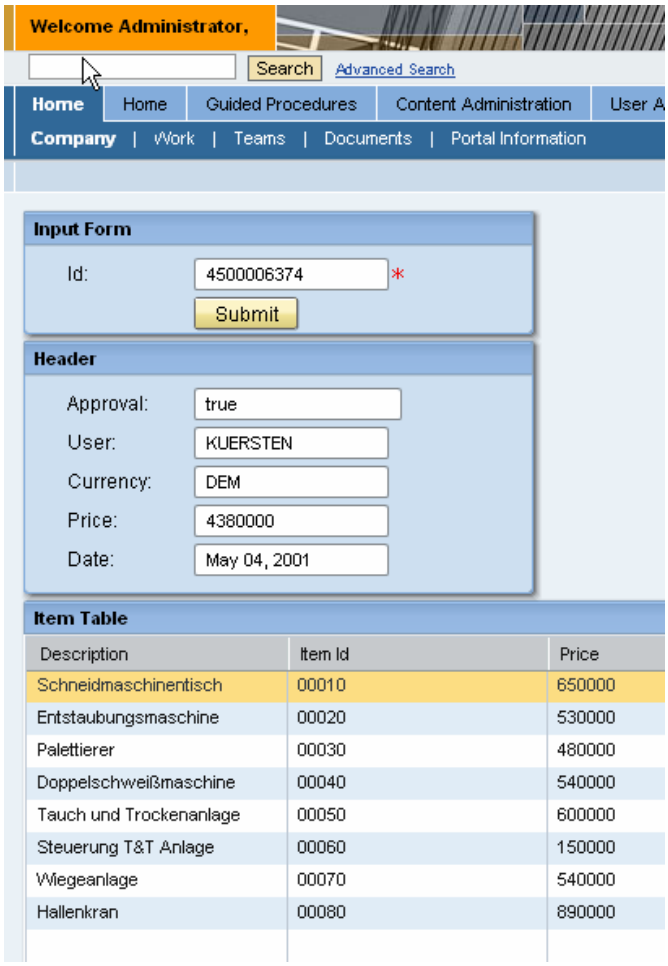
Column	Label
Approvalrequired	Approval
Creationuseraccount	User
Price	Price
PurchaseOrderDate	Date
Item Table-> Id	Item Id

The user interface is now complete! Next we will save, deploy, and run.

- Select  to save the model

- Select  to compile, deploy, and run the model

- Enter a valid PO Id and select “Submit”



Congratulations, you have completed this tutorial!

You have now created a Visual Composer User Interface which consumes the newly composed, added-value composite web service which was built to leverage and enhance the functionality of the SAP Enterprise Service.

This should help you understand some of the benefits of implementing an “open standards” based Enterprise Service Oriented Architecture (ESOA). Some of the most visible benefits we realized in this tutorial was the drastically decreased development time which was enabled by the well defined Enterprise Service and the model driven architecture (MDA) tools (CAF and VC). An enormous amount of efficiency can be realized when the BPX can model the business process using MDA tools and then generate the majority of the composite application. This reduces the amount of knowledge transfer required between the BPX and the programmer. Finally, the BPX can own the implementation of the business process, not just the specification. The MDA tools, including CAF and Visual Composer also produce a highly flexible and maintainable product. Since business process changes can be implemented directly by the BPX with SAP’s ESOA modeling tools rather than a programmer implementing them with low-level code changes, the programmer error risk is virtually eliminated. We also saw that a non-developer BPX could easily perform the majority of the tasks in this development effort (except for the 3 lines of java code)!

The end result is quicker time to market, lower TCO, and the ability to focus on creating innovative business processes. Implementing an ESOA will help adapt IT-supported business models and processes to new market requirements rapidly and flexibly. ESOA’s help to optimize business processes and make them more flexible, support the implementation of new applications and services, and allow innovative business models to be introduced.

Related Content

Service Marketplace download for the ERP Enterprise Service Business Package (registered users only)

Enterprise Services Workplace homepage

Apply for access to the ES Workplace

Installation and configuration of Visual Composer

Visual Composer homepage

Composite Application Framework homepage

Copyright