How to Consume ES Workplace Enterprise Services in ABAP

Applies to:

ES-Workplace ECC 600 of mySAP 2005 Business Suite with ESA Fast Track Add-on SP2.

NetWeaver 04s ABAP SP8.

Summary

This document shows you how to consume enterprise services exposed through the ES Workplace landscape within ABAP. Though the paper explains the necessary steps to connect to the publicly available systems, this approach also applies to any other ECC 600 system of the mySAP 2005 business suite with the ESA Fast Track Add-On installed.

Author(s): Ingo Sauerzapf

Company: SAP Labs LLC

Created on: 25 January 2007

Author Bio

Ingo Sauerzapf is a member of Market Development Engineering working as Solution Architect. He is responsible for communication and solution provisioning to ISVs and transferring their needs back to SAP development.Before joining SAP Labs he was cofounder of quipus AG and worked for e-SAP.de and SAP SI as a SAP technical consultant.

Table of Contents

Introduction

The word of Enterprise Services has been spread by SAP for some time. Nevertheless, it is still hard to get your feet wet if you do not have an ECC 6.0 system with the ESA Fast Track Add-On installed. As SAP is now providing a landscape which is available on the Internet for users registered for the Enterprise Services Workplace, I would like to show you how to consume the enterprise services of the ES Workplace in ABAP. Contrary to the Visual Composer How-to Guide, this requires a small amount of coding.

This guide was intended for developers who have little experience in ABAP. Since I am a veteran ABAP programmer, and might find things obvious where the ABAP newbie has problems, I'd like to apologize in advance for any omissions.

The guide is split into 3 different parts, described below. Depending on your network infrastructure, you may be able to skip part 1.

Part 1 deals with setting up the proxy settings in NetWeaver 04s if you are behind a firewall. If you have a direct connection to the internet, you can skip this section. If your company uses an http proxy server, this part is essential.

Part 2 explains how to create a proxy class for an Enterprise Service on the ES Workplace and how to configure the proxy class for use.

Part 3 shows you how to write a very small example program in ABAP which calls the generated proxy.

Requirements for this How-to Guide

- A copy of SAP NetWeaver 04s ABAP SP8.

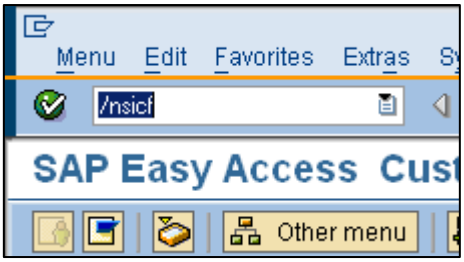
- A user on ES-Workplace in order to consume the Enterprise Services.

Part 1: Configuring Proxy Settings in NetWeaver 2004s ABAP

Before we begin, make sure you know if you sit behind a firewall with the machine on which your NetWeaver 04s ABAP is installed. The easiest way to find out is to look up the settings in Internet Explorer. If you do not have access to it, ask your administrator to provide you with this information.

If you are behind a firewall, continue with this part of the guide. If you are not, you can either skip the section entirely, or read it for purely educational purposes.

Using the OK-Code Field



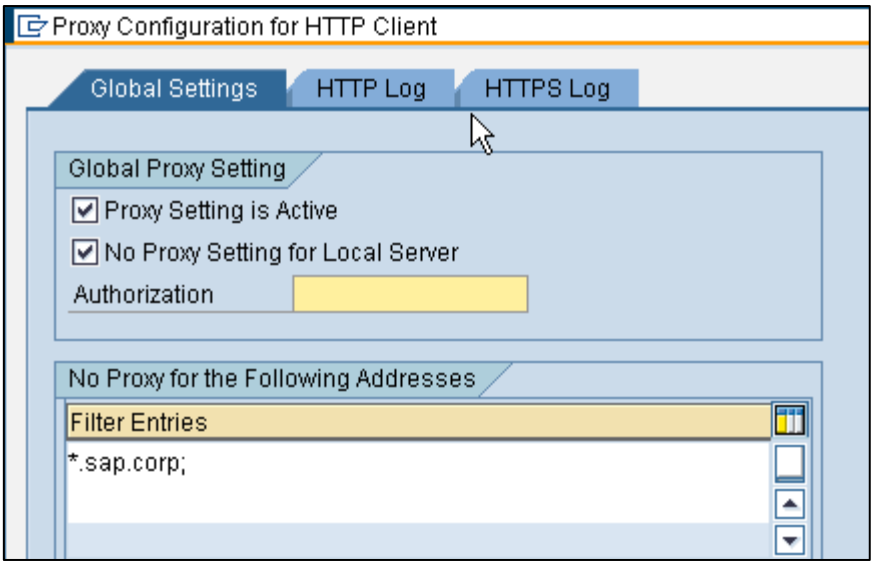
Once you have successfully installed your NetWeaver 04s ABAP System and logged on to it, you will notice a small input field in the upper right hand corner of your SAPGui. The field is called the OK-Code field, and it provides a shortcut to the various screens that will be used in this guide. If you do not see it, then it might be closed. To open it, click on the small arrow next to the green check button. Once the field is open, the arrow icon will face to the left.

In order to make your system use an HTTP proxy, go to the service maintenance screen. To do that, enter "/nsicf" into the OK-Code field and hit the Enter button on your keyboard.

Configuring HTTP and HTTPS Proxy Settings

In the Maintain Services screen, hit the F8 button on your keyboard.

On the next screen, open the "Client" menu and click on "Proxy Settings." This will bring up the Proxy Configuration for HTTP Client dialog, where you can adjust the settings of the HTTP and HTTPS proxies.



In this dialog, specify which domains will not be accessed through the proxy. This is usually the network in which your NetWeaver 04s ABAP resides (*.sap.corp, in this example).

If you are unsure what to enter here, look at your Internet Explorer connection settings or ask your network administrator to provide you with this information.

It is possible to leave this field empty. However, if you leave this empty, you will not be able to see HTTP servers which reside on the same network as your NetWeaver 04s ABAP.

Leave the checkboxes "Proxy Setting is Active" and "No Proxy Setting for Local Server" checked.

Next, move to the “HTTP log” tab. This tab lets you set the host name of the HTTP proxy and the listening port of the HTTP proxy.

Enter the hostname of your HTTP Proxy in the “Host Name” field, and enter the HTTP Proxy listening port in the field “Port”. The information may be available from Internet Explorer – if not, contact your system administrator for the information.

If you also want to use a proxy for your HTTPS connection, repeat this process in the “HTTPS Log” tab.

Now your system is configured to use a proxy for all the addresses which are not specified in the “No Proxy for the Following Addresses” section of the “Global Settings” tab in this dialog. Click the “OK” button to save the entries in the system. After this is done, you are ready for the next step.

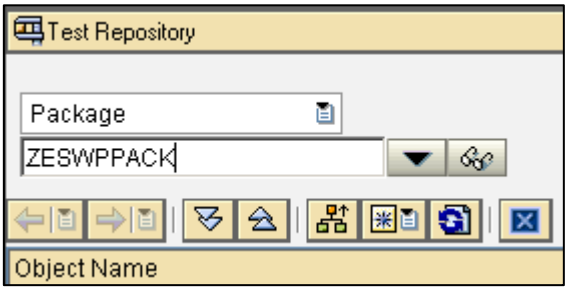
Part 2: Generating and Configuring a Proxy Class for the ES Workplace Enterprise Service

After configuring the proxy settings, you are ready to create a proxy class for the ES Workplace enterprise service we want to use in ABAP. Please note that though this class is also called a proxy, it is completely unrelated to the HTTP proxy settings from part 1.

The proxy we are going to create represents the function we want to invoke on the ES Workplace. In general, it takes care of calling the services and pushing and pulling the right information to and from the enterprise service. As ABAP programmers, we do not have to take care of that functionality ourselves - instead, we use the proxy class to handle this task.

Creating a New Package

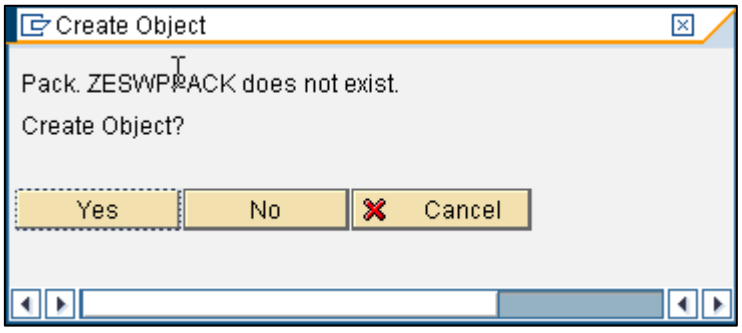
First, access the integrated development environment of SAP NetWeaver 04s ABAP by typing “/nse80” into the OK-Code field. This will bring you to the Object Navigator, the main part of the integrated development environment.



In the Object Navigator screen, you can specify the object you want to see. Since we need to create a new package to store our proxy, start out by selecting “Package” from the drop-down menu above the input field. After that, specify the name you want to give your new package and hit the enter button on your keyboard.

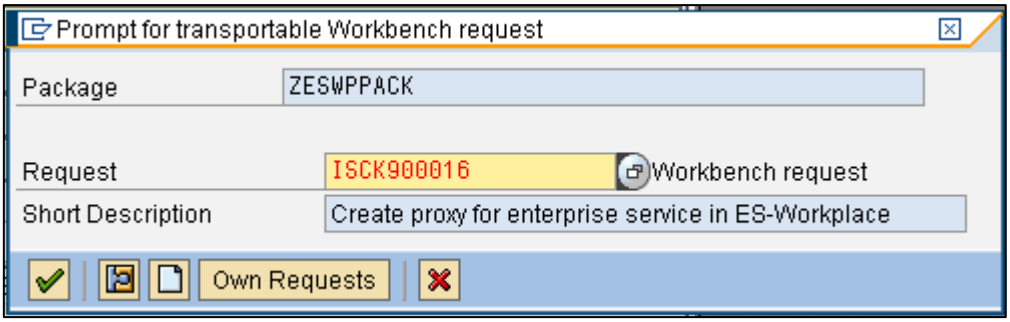
Please note that in order to comply with ABAP namespace regulations, your package and all other objects you will create during this walkthrough have to start with either the letter Y or the letter Z.

After naming your package, a pop-up window will appear to notify you that the object you have specified does not exist. Click on the “Yes” button to create a new object.

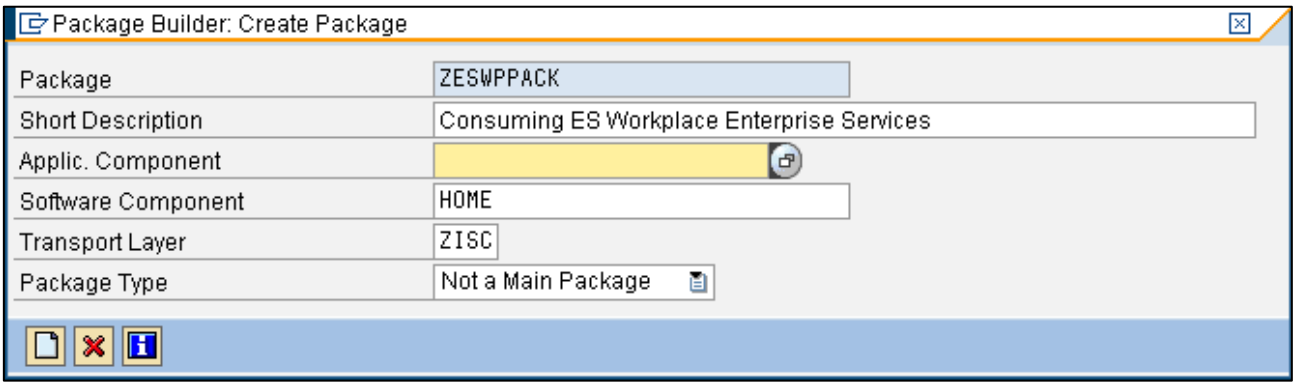


The next pop-up deals with transportation management, which is a functionality you can use to move objects to other SAP NetWeaver 04s systems and is outside the scope of this guide. If you do not already have a transport request, click on the Create

button on the next screen, write a brief description in the next pop-up (not shown here), and click the save button. Once you return to the original pop-up, the newly created transport request will be in the “Request” field. Click the button with the green arrow.

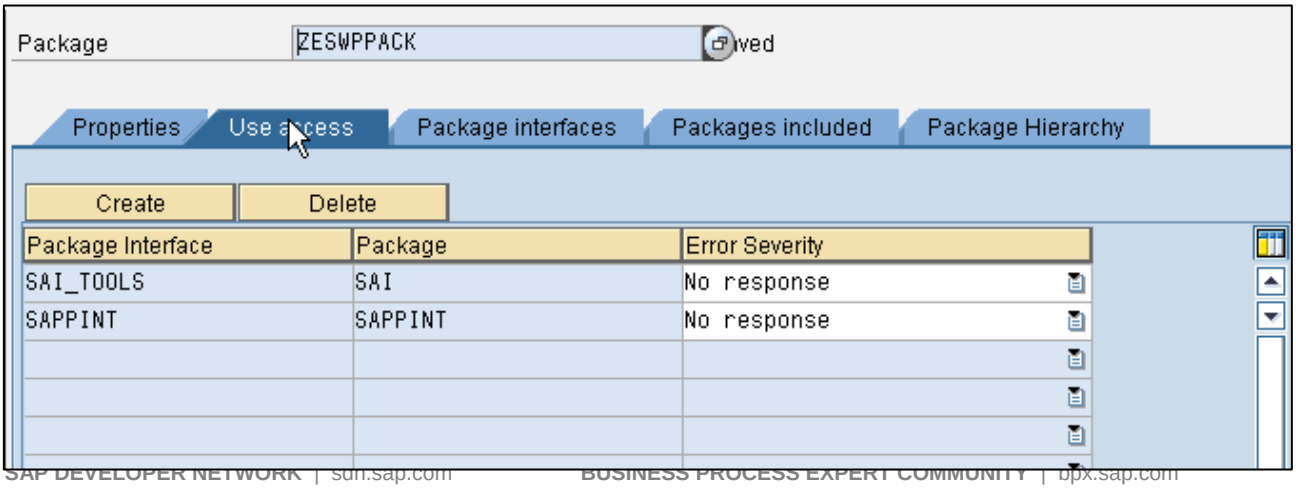


Once you successfully create a new transport request, you will return to the original pop-up. There, you must specify some additional information before you can create the package. Create a simple description in the “Short Description” field and leave the rest of the fields alone. Click the “create” button.



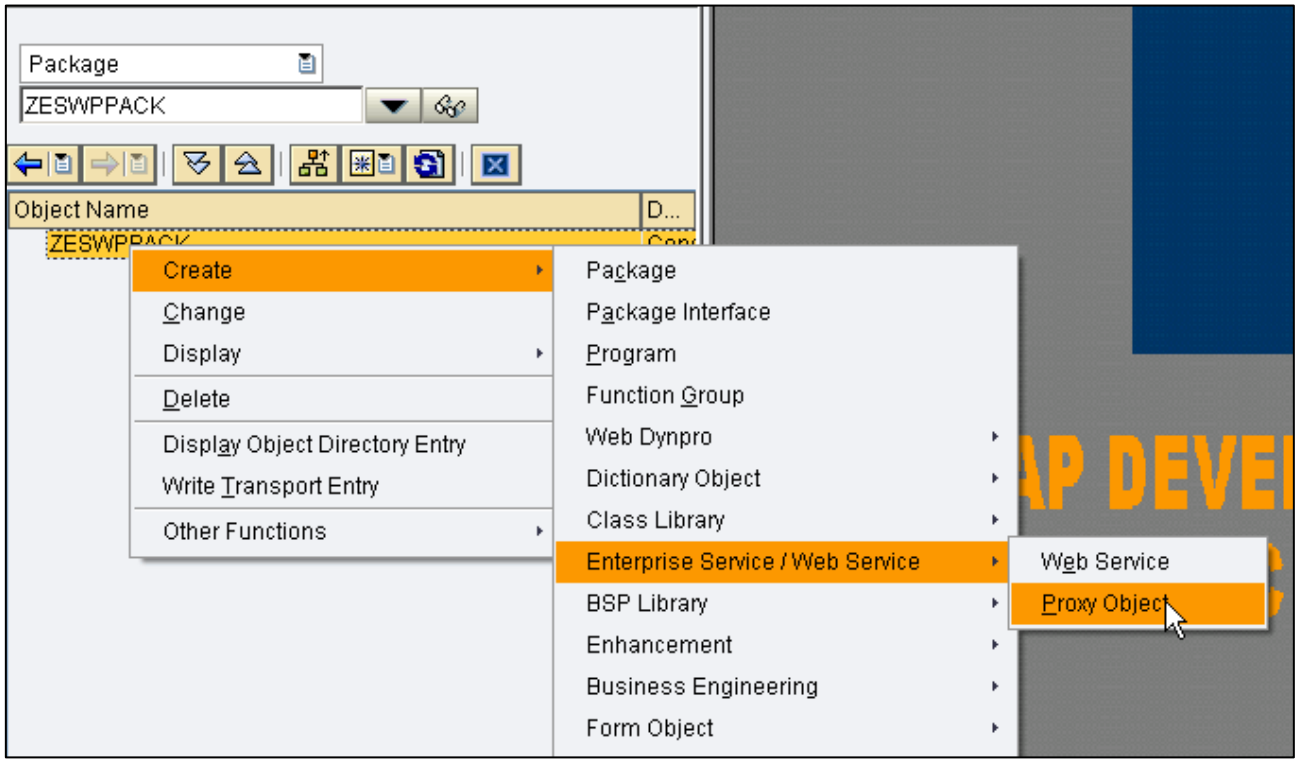
Now that the package is created, double-click on the name of the newly created package in the Object Navigator tree view, which is on the right-hand side below “Object Name.” The properties and attributes of the package itself in the left part of the SAPGui screen. Click on the “Use accesses” tab to specify two package interfaces, which are required by the proxy during runtime. The package interfaces are named “SAI_TOOLS” and “SAPINT.”

From the “Use access” tab, click the “create” button. In the resulting pop-up, enter “SAI_TOOL” into the “Package interface” field and leave the “Error Severity” field set to “No response”. Save, and repeat this process with the Package Interface “SAPINT”. Once you are done, you should see this:

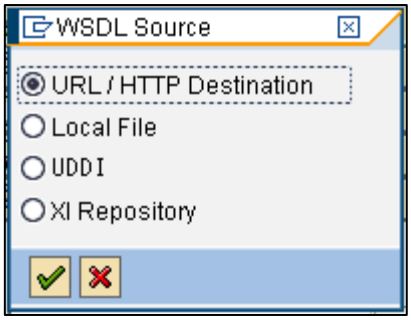


Generating the Proxy Class

We can now start to generate our proxy class to the Enterprise Service on ES Workplace. Right-click on the package name in the Object Navigator to open the context menu. Select “Create” -> “Enterprise Service / Web Service” -> “Proxy Object.”



A tiny pop-up will ask you from where you want to retrieve the definition of the Enterprise Service. Since you have a WSDL definition readily available from the ES Workplace, select “URL/HTTP Destination.” If you do not know how to get the WSDL definition of the Enterprise Service you want to use, refer to the ES Workplace document “How to Use the ES Workplace.”



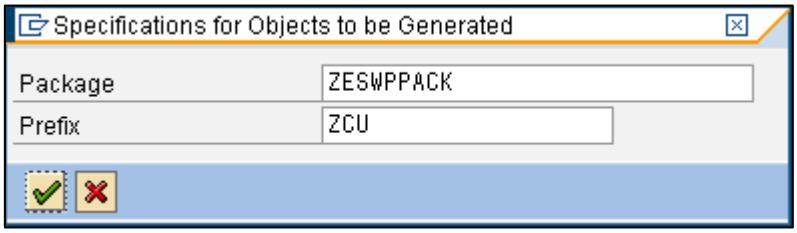
Click OK to continue to the next popup. Enter the URL for the Enterprise Service you want to use and make sure you have the “URL” option checked. For ease of use, use this link to the Customer Query Enterprise Service:

http://erp.esworkplace.sap.com/sap/bc/srt/xip/sap/ECC_CUSTOMER002QR?sap-client=800&wsdl=1.1

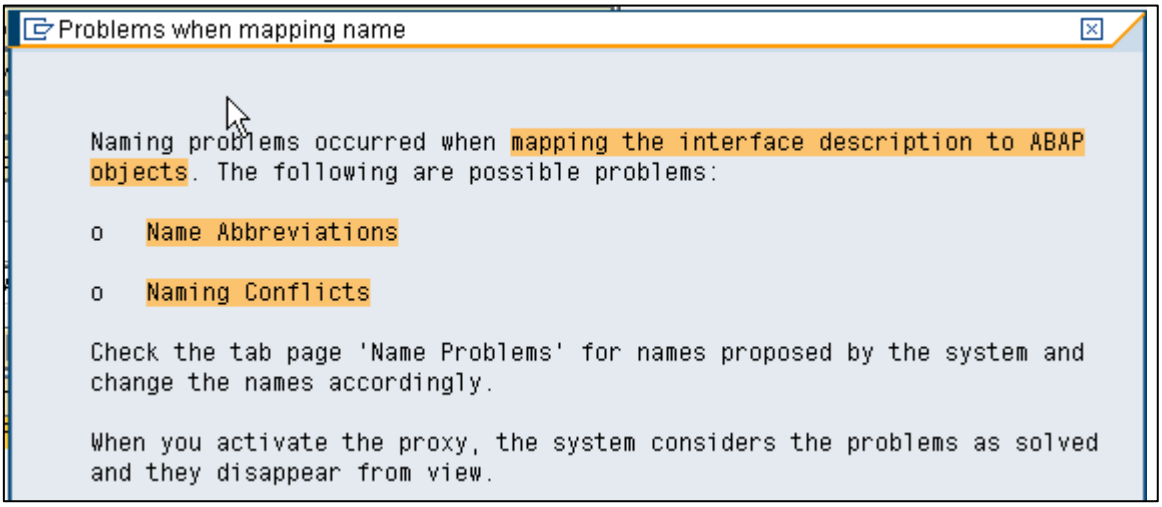
Click the OK button again.

Since the ES Workplace is password protected, enter your username and password in the next pop-up to retrieve the WSDL file from the ES Workplace. After you click on the “OK” button, your system will retrieve the Enterprise Service definition, creating the proxy class and the appropriate input and output structures to work with it.

The next step requires you to specify the package in which you want the proxy class to reside and a prefix for all objects which do get created by the generation of the proxy class. This example uses the package named earlier in the guide, with the prefix “ZCU”. Feel free to use any other prefix, but be aware that you have to stay within the ABAP naming convention, which requires that the name begin with the letter Y or Z.

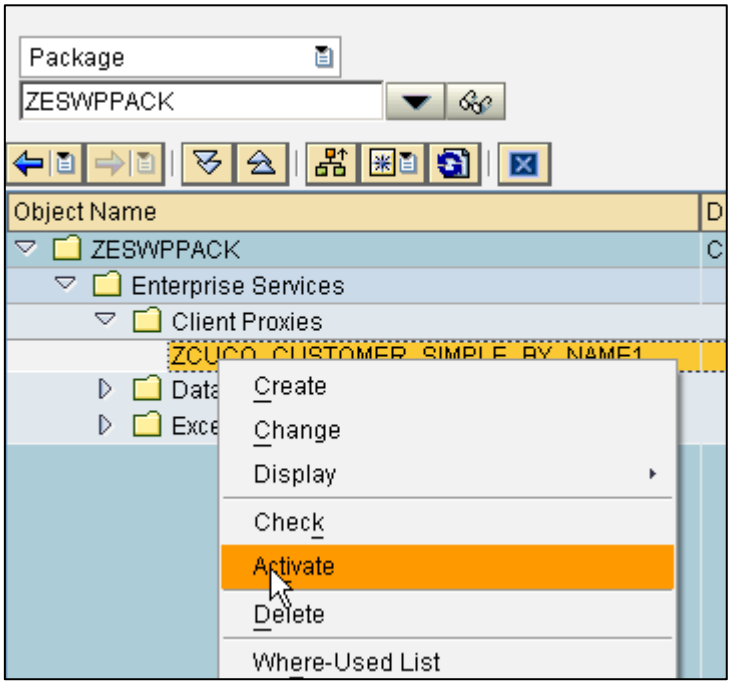


The next screen warns you that some of the names used in the Enterprise Services interface and the operations are either too long or have characters which are not allowed inside ABAP. To prevent problems, the system changes all such names – this might result in some undesirable names, but for now, names suggested by ABAP are fine.



After the system finishes generation, the newly created proxy class must be activated for the newly generated structures to become visible in the object hierarchy.

To do so, first click the “Save” button. Highlight the proxy class in the object hierarchy under your package name (you can find your package name in the object tree located on the left hand side of your SAPGui). Select “Activate” from the context menu.

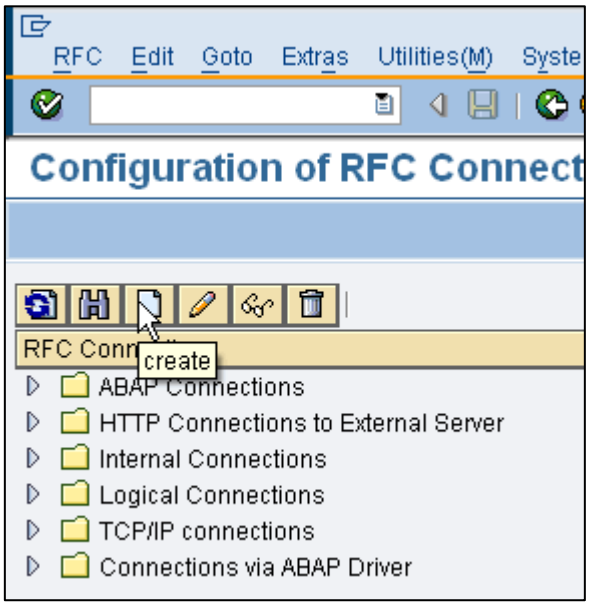


You should see all the generated objects of the proxy class appear underneath the “Enterprise Services” node in the object hierarchy. Before moving on to the next step, please make sure you remember or write down the proxy class’s name. It will be necessary for the next step.

Creating an HTTP Destination

The next task is to create an HTTP Destination so the enterprise service knows where to go to. This is especially important, since this is the only way we can use the basic authentication the ES Workplace requires during runtime.

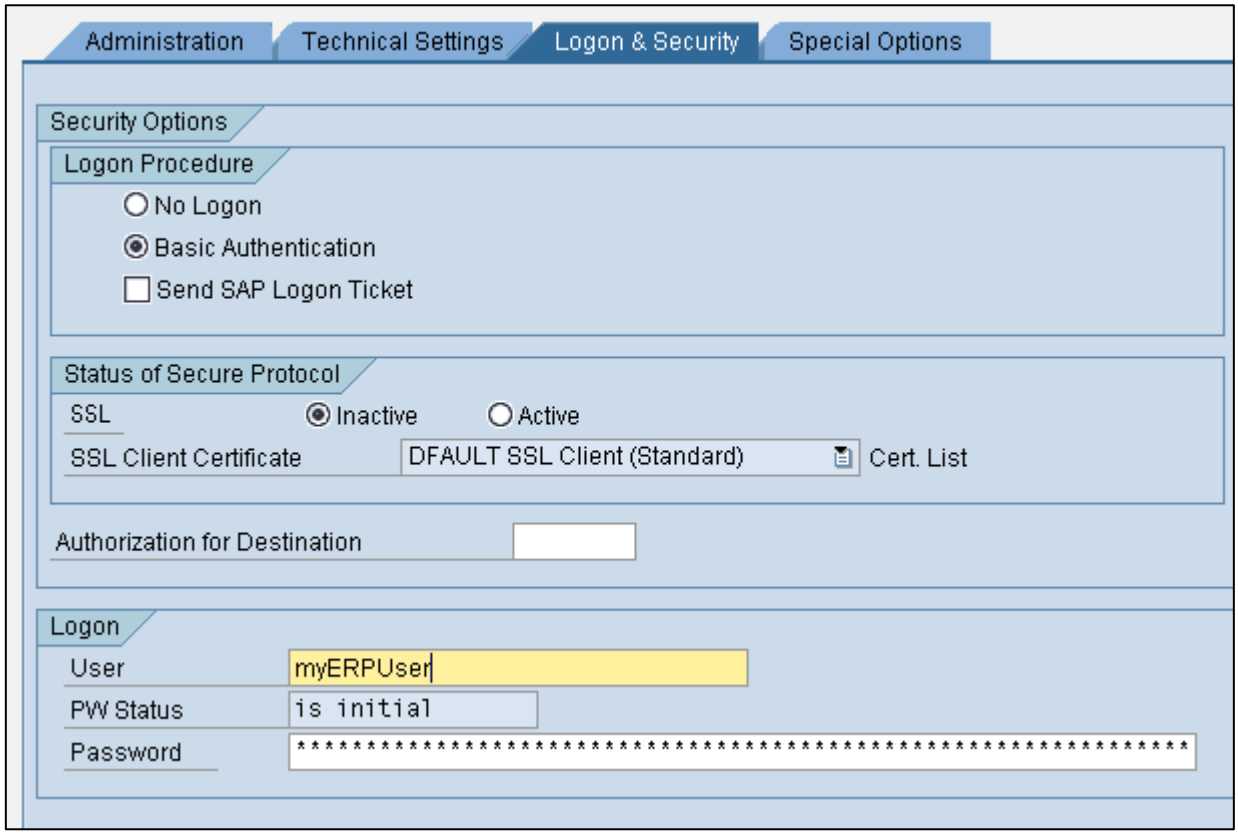
Enter the shortcut “/nsm59” into the OK-Code field to jump to the configuration screen. Click the “create” button to create a new HTTP Destination.



Enter a unique name in the field RFC Destination and select “G” as the connection type in field “Connection Type”. After you enter a short description of where the HTTP Destination leads, you will be prompted with a pop-up to remind you that HTTP connections may not be secure.

In the “Target Host” field, specify the ES Workplace system you want to connect to. For now, we want to connect to the ERP system in the ES Workplace landscape. Use the hostname “erp.esworkplace.sap.com”. Specify 80 as the service number – this is the HTTP port of the ERP system of the ES Workplace. Leave the field “Path Prefix” empty.

Switch to the “Logon & Security” tab and change the “Logon Procedure” option to “Basic Authentication.” A pop-up will appear to inform you that the username and password information you might have provided earlier will be removed if you continue. Since you have not yet specified a username or password, simply continue.



Once this is done, specify your ES Workplace username and password under “Logon” to enable connection to the ES Workplace.

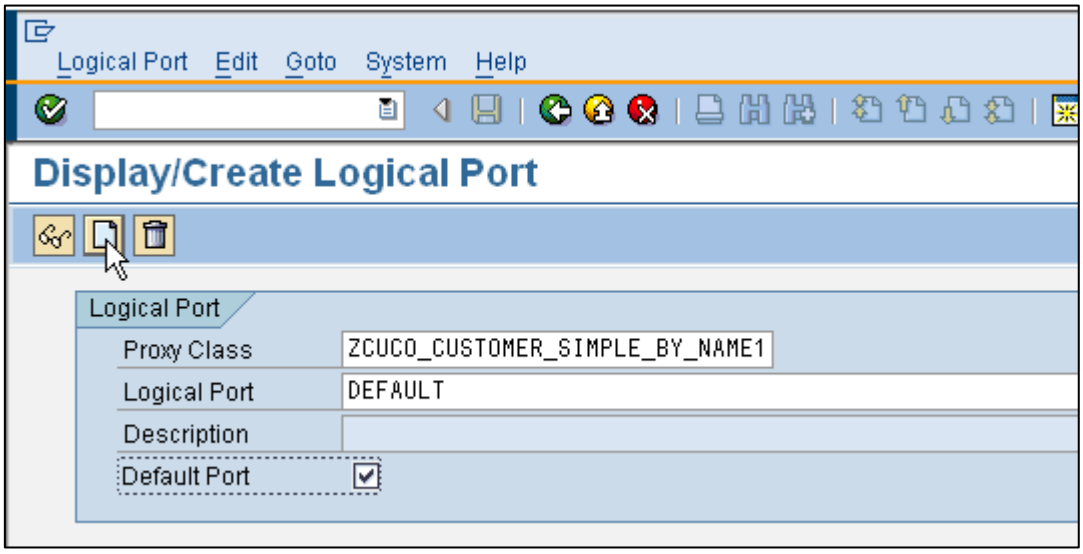
The HTTP Destination is now configured, so be sure to save your work. If you wish, you can use the “Connection Test” button to make sure you the HTTP Destination is configured properly.

Configuring the Endpoint

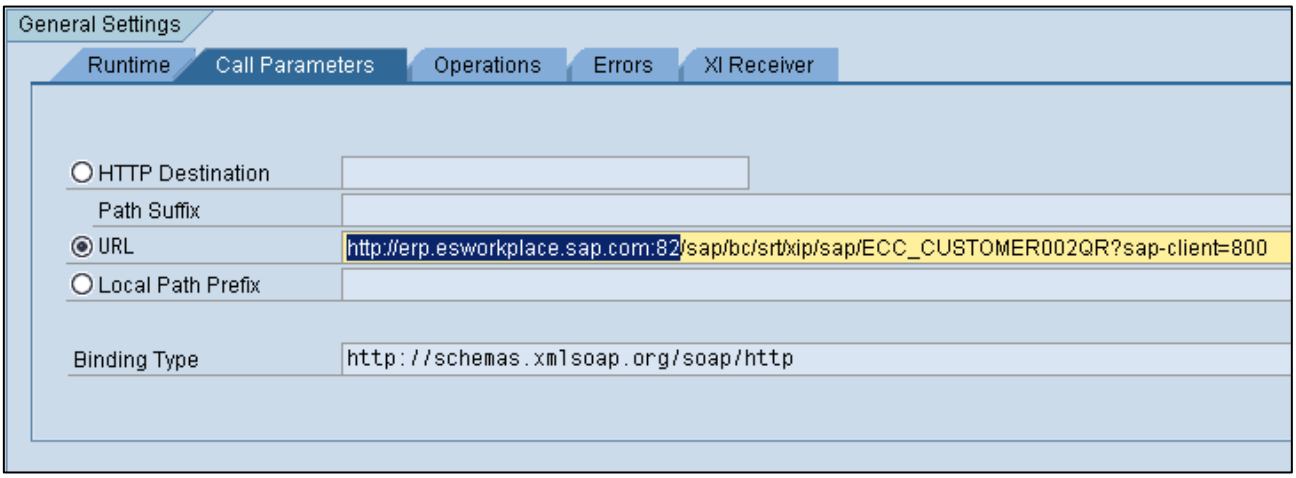
The only thing left before we are finally able to use the proxy in a simple ABAP program is to configure the endpoint for this proxy so it knows where to go during runtime.

Enter “/nplconfig” in the OK-Code field to open the configuration screen.

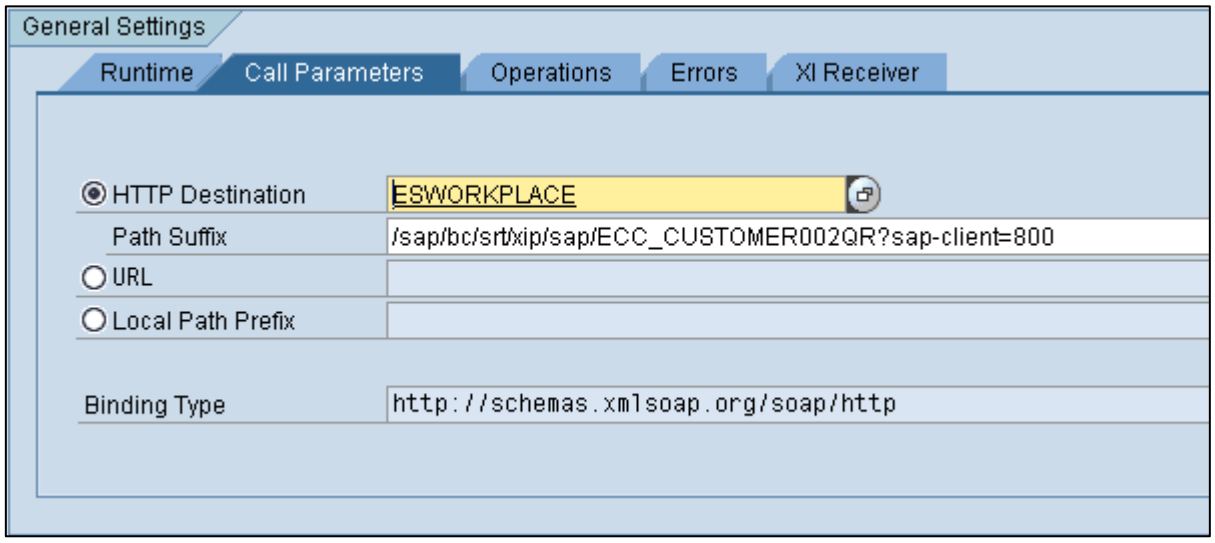
Hopefully, you still remember the name of your proxy class. Enter it in the field “Proxy Class.” In the field “Logical Port,” specify a name for the logical port you want to use. The example uses “DEFAULT,” but any other name can be used as well. Make sure the “Default Port” box is checked. When you’re finished, click the “Create” button.



On the next screen, you will see that the system has retrieved the endpoint of your Enterprise Service from the WSDL. Remove the portion of the URL highlighted below - we will use the HTTP Destination we created in the previous step.



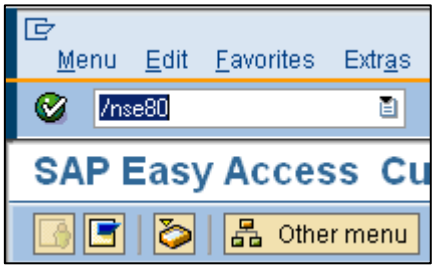
After deleting the highlighted text, save the remaining text to your clipboard. Select the “HTTP Destination” option and paste the saved part of the URL into the “Path Suffix” field. Use the button next to the “HTTP Destination” input field to select the destination you have created before. Your screen should now look similar to the following screenshot:



All there is left to do is give the Logical Destination a short description in the “Description” field. After that, activate the Logical Destination by using the icon with the match on the icon bar. If you do not activate the Logical Destination, it will not be usable and the sample program in Part 3 will throw an exception.

Part 3: Creating a Sample Application in ABAP Which Calls the Enterprise Service

Now that you have created the proxy class and have configured the settings for the proxy class to connect to the ES Workplace, you can create a simple ABAP program which uses this proxy class to call the Enterprise Service on ES Workplace.

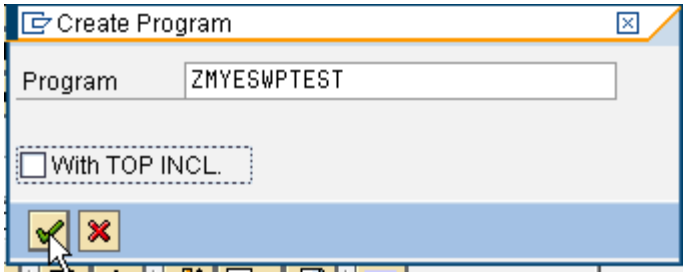


The first thing to do is to return to the integrated development environment by entering the shortcut “/nse80” into the OK-Code field.

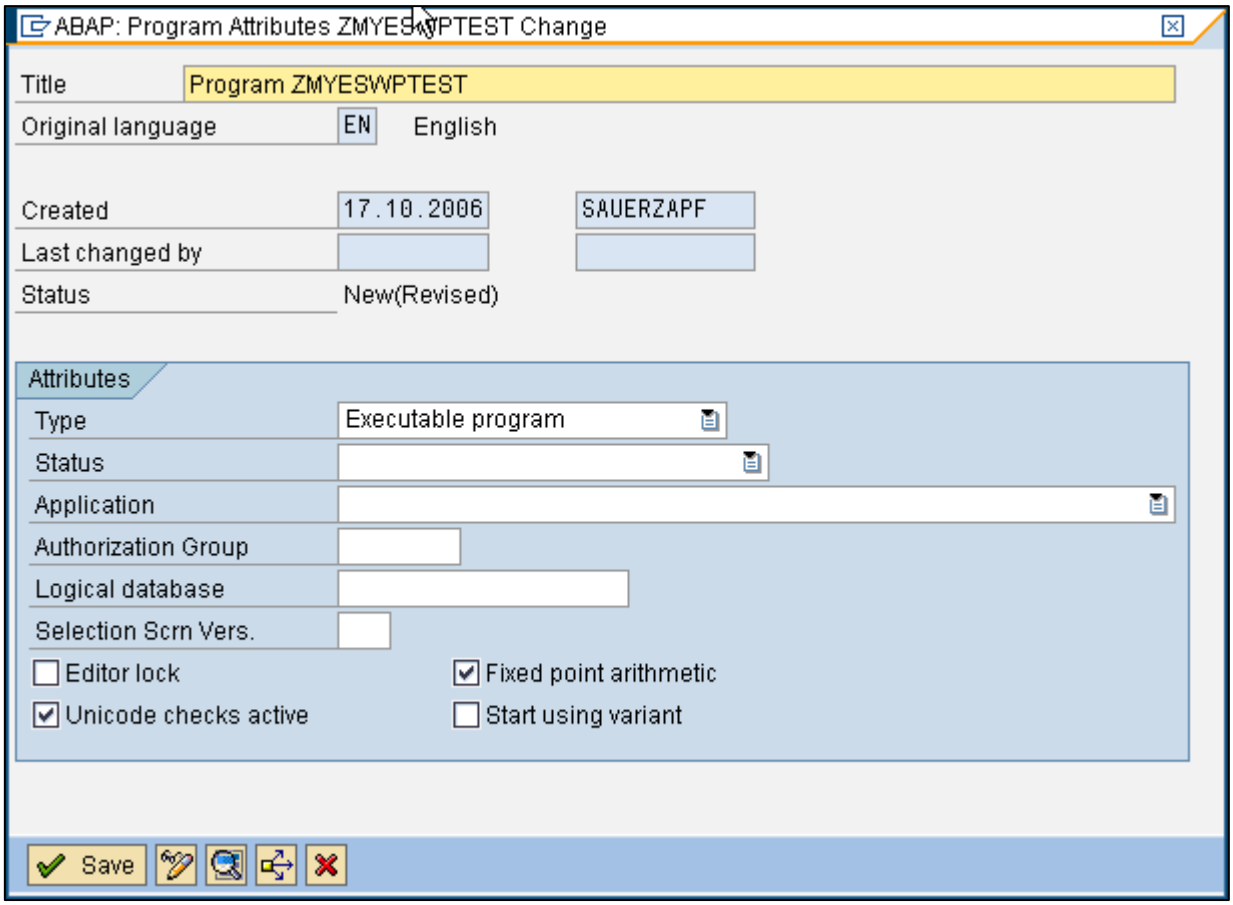
If you did not move to any other object, your package will still be visible once the integrated development environment is shown in SAPGui. If your package is not shown any more select package from the drop down box above the input field and enter your package name in the input field. The package is now visible again on the screen.

Right-click your package and select “Create” -> “Program” from the context menu.

Next, give your program a name within the customer namespace Y or Z. Make sure you have unchecked the “With TOP INCL.” option. The sample program will be too small to justify a top include. Click “OK” and move on to the next pop-up.



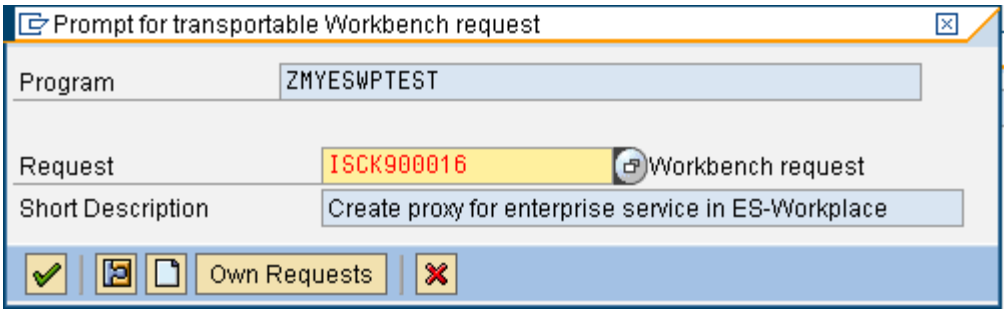
Accept the settings on this screen as they are suggested by the system and click the “Save” button.



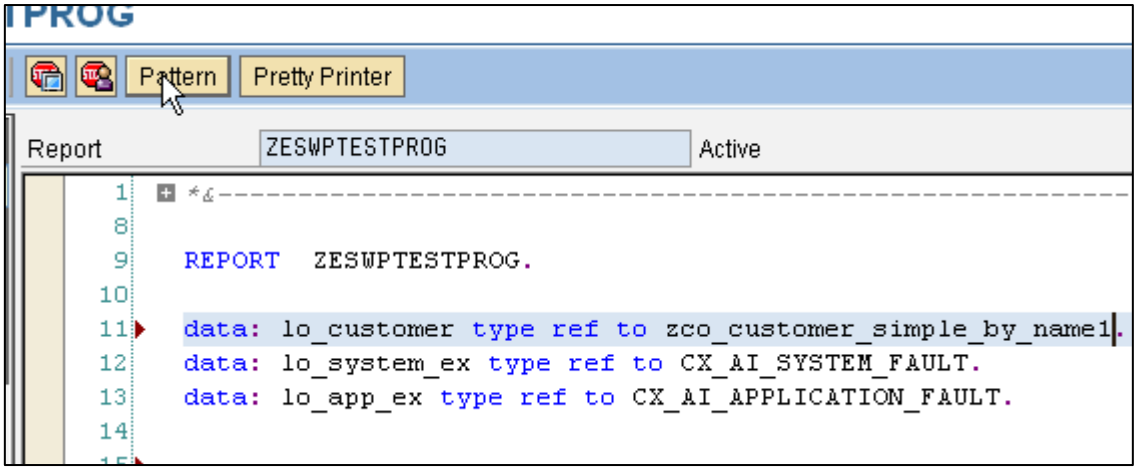
The next pop-up asks you where you want your program to reside. Just accept the settings the system suggests, since this is the package you have created for this guide. Click “Save” to move on to the next pop-up.

The next pop-up asks you what transport request you want to have this program in. Accept the default settings here as well.

Click the “OK” button to create the program.



Once you can see your empty program on the right, use the “Pattern” button to create an object for the proxy class. Do not simply cut and paste code here, as the proxy names will change with each system.



Before you let the system create some of the code we need to use to call the proxy, you have to define an instance variable for the proxy and for the exceptions. Insert the definitions you see in the screenshot above. The first one is for the proxy itself and the subsequent two are for the catch objects. Also remember that the proxy class name might be different on your system.

On the pattern screen, select “ABAP Objects Patterns” and click ”OK.”

Select “Create Object” in the next popup and specify “lo_proxy_customer” or any instance name you like and “Z….. Proxyclassname” in field “Class”. After that, click “OK” to create an instance of that class in your program.

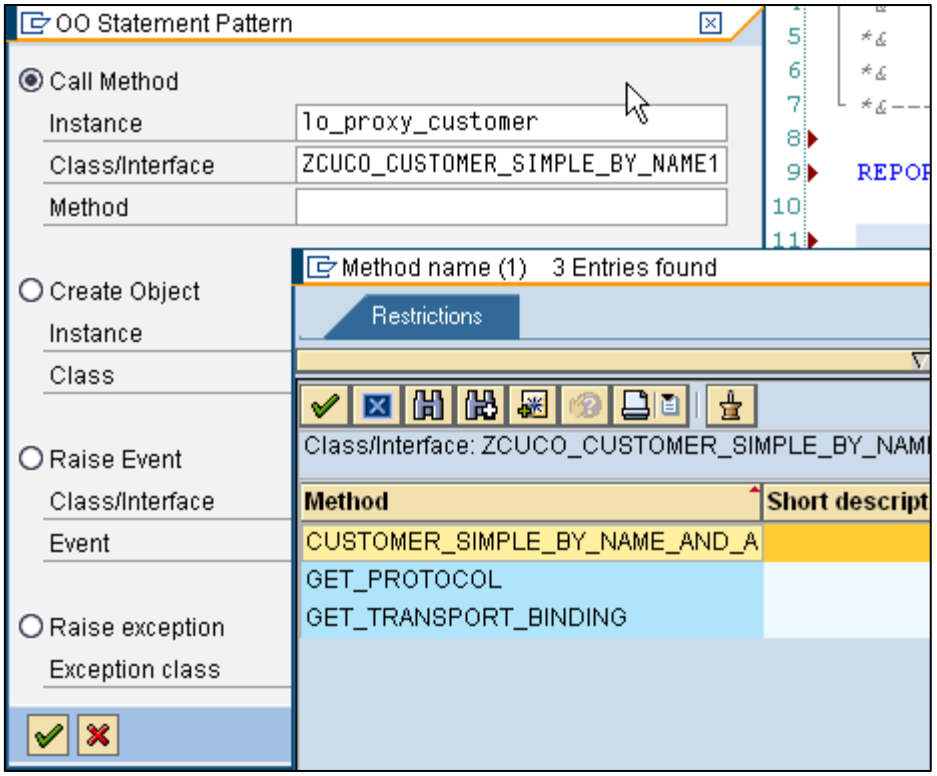
To look up the class name your system created for the proxy you can use the object hierarchy on the left hand side under your package name. Just open up “Enterprise Services” -> “Client Proxies” and under this node you should see the class name of your proxy class.

Next we are going to create the code to call the executing method of the operation in our Enterprise Service. To do so place your cursor right under the line

```
CREATE OBJECT LO_PROXY_CUSTOMER
*  EXPORTING
*    LOGICAL_PORT_NAME  =
    .
```

Use the “Patterns” button to create the code. Select “Call Method” and insert “lo_proxy_customer” in the “Instance” field. In the field “Class/Interface,” insert your proxy class name. Press F4 to bring up the help for choosing the available methods in your proxy class.

The Method we need to select should be “CUSTOMER_SIMPLE_BY_NAME_AND. The name here also depends on the proxy generation and might be slightly different in your system.

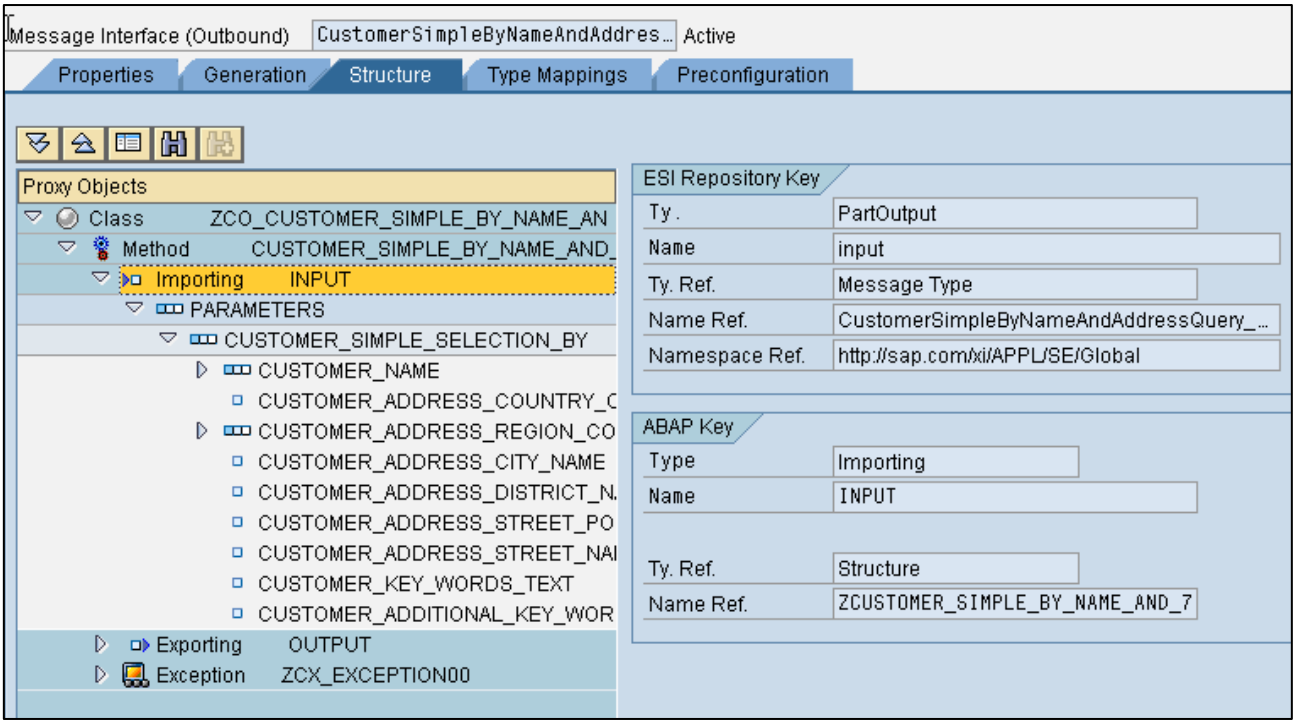


You program should now look similar to the code shown here:

```
REPORT  ZESWPTESTPROG.

data: lo_customer type ref to zco_customer_simple_by_name1.
data: lo_system_ex type ref to CX_AI_SYSTEM_FAULT.
data: lo_app_ex type ref to CX_AI_APPLICATION_FAULT.

TRY.
CREATE OBJECT LO_PROXY_CUSTOMER
*  EXPORTING
*    LOGICAL_PORT_NAME  =
    .
*TRY.
CALL METHOD LO_PROXY_CUSTOMER ->CUSTOMER_SIMPLE_BY_NAME_AND_A
  EXPORTING
    INPUT  =
*  IMPORTING
*    OUTPUT =
    .
* CATCH CX_AI_SYSTEM_FAULT .
* CATCH ZCX_EXCEPTION00 .
* CATCH CX_AI_APPLICATION_FAULT .
*ENDTRY.

CATCH CX_AI_SYSTEM_FAULT .
ENDTRY.
```


You now need to look up some names from the proxy definition. Open another screen so you can see the names and your program at the same time without moving back and forth between them in one screen alone. To do so use the shortcut "/ose80" in the OK-Code field. This opens up a second window.

In this second window, double click on the name of your proxy class on the left hand side navigation tree and select the "Structure" tab on the right hand side. Drill down the Proxy Class until you see the complete input and output structure of the proxy class.



Next to the proxy classes input and output structure, you will find two areas. The one named "ABAP Key" is the important one for now. In the field "Name Ref," you can find the name of the structure you are looking in the hierarchy of the proxy class. Since you need to define variables to input and output something to and from the proxy class, you need to create two objects in your program. One will be for the input and one for the output.

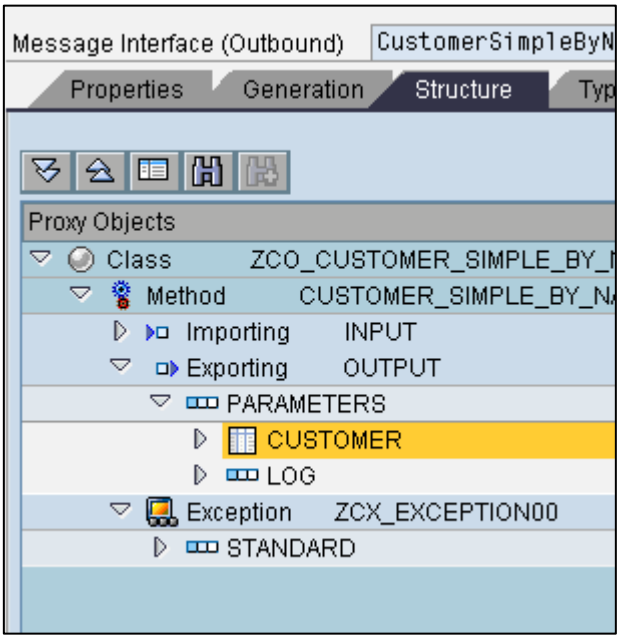
```
data: lo_customer type ref to zco_customer_simple_by_name1.
data: lo_system_ex type ref to CX_AI_SYSTEM_FAULT.
data: lo_app_ex type ref to CX_AI_APPLICATION_FAULT.


* input and output definition.
data: ls_customer_in type ZCUSTOMER_SIMPLE_BY_NAME_AND_7.
data: ls_customer_out type ZCUSTOMER_SIMPLE_BY_NAME_AND_6.

TRY.

CREATE OBJECT LO_CUSTOMER
```


Now that all the definitions are in place for the input and output, assign the input and output variables to the interface of the method called from the proxy class to execute the enterprise service. After streamlining the code a bit and removing the unnecessary comments, your code should now look like this:

```
REPORT  ZESWPTESTPROG.

data: lo_customer type ref to zco_customer_simple_by_name1.
data: lo_system_ex type ref to CX_AI_SYSTEM_FAULT.
data: lo_app_ex type ref to CX_AI_APPLICATION_FAULT.

* input and output definition.
data: ls_customer_in type ZCUSTOMER_SIMPLE_BY_NAME_AND_7.
data: ls_customer_out type ZCUSTOMER_SIMPLE_BY_NAME_AND_6.


TRY.

CREATE OBJECT LO_PROXY_CUSTOMER.

    CALL METHOD LO_PROXY_CUSTOMER ->CUSTOMER_SIMPLE_BY_NAME_AND_A
      EXPORTING
        INPUT  = ls_customer_in
      IMPORTING
        OUTPUT = ls_customer_out.

    CATCH CX_AI_SYSTEM_FAULT .
    CATCH CX_AI_APPLICATION_FAULT.

ENDTRY.
```

The next step is to define some variables for the output. Since the output of the proxy class produces very long names, define two more variables simply for outputting the result on the screen.

To find the name of the generated structure in the proxy for the customer, go to the “Structure” tab of the created proxy class again. Drill down to “Exporting” -> “Parameters” -> “Customer” in the tree and look up the name on the right hand side of the “ABAP Key” area. You can find the name in the field “Name Ref.”



Define the two variables as shown below. One of the variables will hold the customers during runtime, the other one will be used when we loop over the first one to finally output the customer on the screen.

```
REPORT  ZESWPTESTPROG.

data: lo_customer type ref to zco_customer_simple_by_name1.
data: lo_system_ex type ref to CX_AI_SYSTEM_FAULT.
data: lo_app_ex type ref to CX_AI_APPLICATION_FAULT.

* input and output definition.
data: ls_customer_in type ZCUSTOMER_SIMPLE_BY_NAME_AND_7.
data: ls_customer_out type ZCUSTOMER_SIMPLE_BY_NAME_AND_6.

* output evaluation helper
data: lt_customer type ZCUSTOMER_SIMPLE_BY_NAME_A_TAB.
data: lw_customer like line of lt_customer.


TRY.

CREATE OBJECT LO_PROXY_CUSTOMER.
```

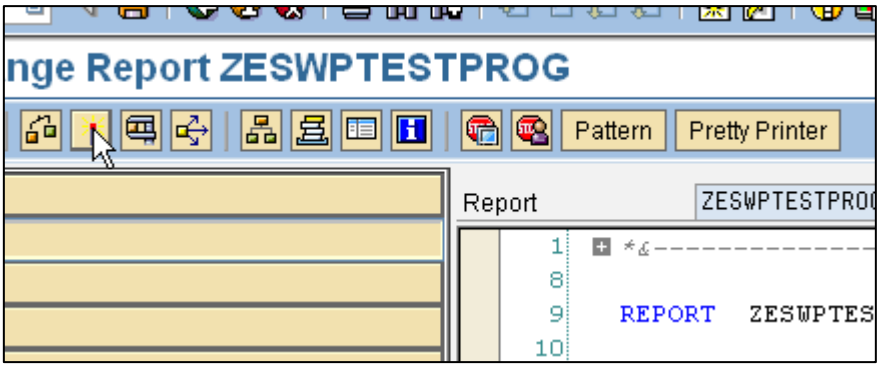
The last thing we have to do is to assign some test data to the input structure and once the proxy got executed move the customers from the output structure of the proxy class to the variables we created before. Once this is done you can loop over the variables and output them on the screen. The coding on the next page shows how to do this.

To find the correct structures and definitions you can again use the “Structure” tab in your proxy class.

```
TRY.
  CREATE OBJECT LO_PROXY_CUSTOMER.

* add some stuff to the input of the web service.
  ls_customer_in-parameters-customer_simple_selection_by-
                      customer_name-first_line_name = 'Maria'.
  ls_customer_in-parameters-customer_simple_selection_by-
                      customer_address_country_code = 'DE'.

  CALL METHOD LO_PROXY_CUSTOMER->CUSTOMER_SIMPLE_BY_NAME_AND_A
    EXPORTING
      INPUT  = ls_customer_in
    IMPORTING
      OUTPUT = ls_customer_out.

* print the output of the web service on the screen
   lt_customer = ls_customer_out-parameters-customer.

   loop at lt_customer into lw_customer.
      write: / lw_customer-id-value,
             lw_customer-basic_data-common-name-first_line_name,
             lw_customer-basic_data-common-natural_person_indicator.
   endloop.

CATCH CX_AI_SYSTEM_FAULT into lo_system_ex.
```


Once the program is done, it should look something like this. Remember that you actually have to look up the name of the structures used. If you simply copy and paste this example, it most likely will not work.

```
REPORT  ZESWPTESTPROG.


data: lo_customer type ref to zco_customer_simple_by_name1.
data: lo_system_ex type ref to CX_AI_SYSTEM_FAULT.
data: lo_app_ex type ref to CX_AI_APPLICATION_FAULT.

* input and output definition.
data: ls_customer_in type ZCUSTOMER_SIMPLE_BY_NAME_AND_7.
data: ls_customer_out type ZCUSTOMER_SIMPLE_BY_NAME_AND_6.

* output helper
data: lt_customer type ZCUSTOMER_SIMPLE_BY_NAME_A_TAB.
data: lw_customer like line of lt_customer.

TRY.
  CREATE OBJECT LO_PROXY_CUSTOMER.

* add some stuff to the input of the web service.
  ls_customer_in-parameters-customer_simple_selection_by-
                      customer_name-first_line_name = 'Maria'.
  ls_customer_in-parameters-customer_simple_selection_by-
                      customer_address_country_code = 'DE'.

  CALL METHOD LO_PROXY_CUSTOMER->CUSTOMER_SIMPLE_BY_NAME_AND_A
    EXPORTING
      INPUT  = ls_customer_in
    IMPORTING
      OUTPUT = ls_customer_out.

* print the output of the web service on the screen
   lt_customer = ls_customer_out-parameters-customer.

   loop at lt_customer into lw_customer.
      write: / lw_customer-id-value,
             lw_customer-basic_data-common-name-first_line_name,
             lw_customer-basic_data-common-natural_person_indicator.
   endloop.

CATCH CX_AI_SYSTEM_FAULT into lo_system_ex.
      write: / 'System fault occurred', lo_system_ex->ERRORTEXT.
CATCH CX_AI_APPLICATION_FAULT into lo_app_ex.
      write: / 'Application fault occurred ',  lo_app_ex->TEXTID.

ENDTRY.
```

For testing your sample program, you have to generate and activate your program and then run it through the "Direct Processing" button. Click the "Activate" button to generate and activate your program. After this is completed, click the "Direct Processing" button to run your program.

If your program executes successfully, you should see something like this:

Congratulations! You have successfully consumed an enterprise service from the ES Workplace in ABAP.

Related Content

Enterprise Services Workplace

Enterprise Services Workplace Registration

SAP NetWeaver 04s ABAP Download

How to Use the ES Workplace

Copyright