ABAP Proxy Interfacing

Applies to:
This document applies to SAP versions ECC 6.0.

Summary
This article contains the guidelines for using the ABAP Proxy interfacing. ABAP proxies are used when ABAP applications needs to send or receive messages. ABAP proxies are generated on the Application server with transaction SPROXY, based on the WSDL representation of the message interface.

Author: Prasanth Prabhakaran
Company: Applexus Software Solutions (P) Ltd
Created on: 27 Jun 2011

Author Bio
Prasanth P is working as SAP ABAP HR with Applexus Software Solutions (P) Ltd. He is having expertise in various Implementation, Support and Rollout Projects across various countries.
## Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>3</td>
</tr>
<tr>
<td>Inbound Proxy</td>
<td>4</td>
</tr>
<tr>
<td>Service Interface</td>
<td>4</td>
</tr>
<tr>
<td>Provider class methods</td>
<td>5</td>
</tr>
<tr>
<td>Service implementation class</td>
<td>7</td>
</tr>
<tr>
<td>Outbound Proxy</td>
<td>8</td>
</tr>
<tr>
<td>Proxy Class</td>
<td>8</td>
</tr>
<tr>
<td>Related Content</td>
<td>10</td>
</tr>
<tr>
<td>Disclaimer and Liability Notice</td>
<td>11</td>
</tr>
</tbody>
</table>
Introduction

ABAP proxies are used when ABAP applications need to send or receive messages. ABAP proxies are generated on the Application server with transaction SPROXY.

Advantages of ABAP Proxies.

1. Proxy communication always bypasses the Adapter Engine and will directly interact with the application system and Integration engine. So it will give us better performance.
2. Proxies communicate with the XI server by means of native SOAP calls over HTTP.
3. Easy to handle messages with ABAP programming.
4. Proxy is good for large volumes of data. We can catch and persist the errors (both system & application fault) which was generated by Proxy setting.

The rest of the document gives you the steps that are necessary for generating and executing ABAP proxies.
Inbound Proxy
Activate the proxy by double clicking the service interface menu
After activating double click the service interface method
And select the provider class by double clicking on it.

Service Interface
Provider class methods
Now the provider class methods are shown

Double click the proxy method of provider class

Note: Before this please create a service implementation class (Service implementation class is a normal class (T.code– se24))
Below is the implementation of above method

In this method first we have to deactivate the xml handling, and then call the service implementation class method (Copy & paste the same code and change the service implementation class name)

```abap
DATA lo_service_class TYPE REF TO ZCL_HR_TIME_DATA_INTSYSMS_IN.

TRY.
  
  deactivate extended xml-handling
  CALL METHOD mo_protocol_payload->set_extended_xml_handling
  EXPORTING
    extended_xml_handling = abap_false.

  SET UPDATE TASK LOCAL.

  lo_service_class = ZCL_HR_TIME_DATA_INTSYSMS_IN->get_instance().

  CALL METHOD lo_service_class->execute_in
  EXPORTING
    is_input = input.

  CATCH cx_sapplico_standard_msg_fault.
    RAISE EXCEPTION TYPE cex_hr_09180.
  ENDTRY.

ENDMETHOD.
```
Service implementation class

Below are the service implementation class methods.

The first 6 methods are for FEH and this is inherited from the interface IF_ECH_ACTION

In our service implementation class we need to create the implementation for these inherited methods (Refer the attached code)

The method flow of service implementation methods of this example are given below

Step1: provider class will call the execute method of service implementation class

Step2: The execute method will call the process_in (service implementation class) method

Step3: process_in method will call input mapping method and backend process method

Step4: Backend process method will call the backend logic methods (Logic wrote for the data updation validation, Etc)
Outbound Proxy

Proxy Class

See above the generated outbound proxy

Double click the proxy class

The proxy class method are shown above
Please select the method `REPLICATE_EMPLOYEE_TIMECALEND` and click the button parameters.

Here we can see the output structure type; in this example the type was `ZSI_HRPT_REPLICATE_EMPLOYEE_T4`.

We want to use a `SE38` program for passing the data to proxy. The example code lines for this above proxy below,

```abap
ls_proxy_out-replicate_employee_time_calend-employee_time_calendar = lt_employee_time_calendar.

TRY.
    CREATE OBJECT lo_hrpt_co_emp_time_roster.
    CALL METHOD lo_hrpt_co_emp_time_roster->replicate_employee_time_calend
        EXPORTING
            output = ls_proxy_out.
    COMMIT WORK.
    CATCH cx_ai_system_fault INTO lo_system_fault.
ENDTRY.
```
Related Content

http://help.sap.com/saphelp_nw04/helpdata/en/14/555f3c482a7331e10000000a114084/content.htm

http://help.sap.com/saphelp_nw04/helpdata/en/ab/585f3c482a7331e10000000a114084/content.htm

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.