Consumer and Provider ABAP Proxy Service to Multiple SAP NW PI Systems Using Process Integration 7.1

Applies to:
SAP NetWeaver Process Integration 7.1
SAP Application Server ABAP 7.1
SAP Application Server ABAP 7.0 SP 14
For more information, visit the Service Bus-based Integration homepage.

Summary
In SAP PI 7.0 SP13 and below, ABAP proxies can only communicate with one SAP PI system. SXMB_ADM configuration change will be required when the ABAP proxy needs to integrate with another SAP PI system.

With SAP NW PI 7.1, an alternative configuration, using WS adapter, is now available to create multiple configurations using the same ABAP proxy, based on Application Server ABAP 7.0 SP14+ or Application Server ABAP 7.1, to communicate with multiple SAP NW PI 7.1 systems.

In the article, we will provide a step-by-step configuration required for both consumer and provider ABAP proxies.

Author: William Li
Company: SAP Labs LLC
Created on: 7 November 2008

Author Bio
William Li is a member of the SAP NetWeaver Integration Team in SAP NetWeaver RIG Americas focusing on Process Integration. He has been with SAP since 1998 - originally as a SAP America consultant then as a developer at SAP Labs. He joined SAP NetWeaver RIG Americas in January of 2003.
Table of Contents

Introduction .........................................................................................................................................................3
  Restrictions .....................................................................................................................................................3
  Requirements ................................................................................................................................................3
  What is New? ................................................................................................................................................4

Business Scenario ..............................................................................................................................................5

Design and Configuration Steps .........................................................................................................................6
  1. Design Inbound and Outbound Interfaces and mappings for consumer and provider .......................7
  2. Export and import the interface design as external definitions .............................................................10
  3. Create the interfaces on all the IR and ESR systems using the external definitions ...........................12
  4. Create mappings and all required design objects on the PI 7.1 systems ...........................................17
  5. Generate the ABAP proxies on the consumer and provider systems .................................................18
  6. Use tx: SOAMANAGER to configure the endpoints for the provider proxy ......................................21
  7. Configure the objects in the ID of the PI 7.1 systems ........................................................................26
  8. Use tx: SOAMANAGER to configure the ports/bindings for the consumer proxy .............................35
  9. Test the consumer proxy using an ABAP program ..............................................................................39

Appendix ...........................................................................................................................................................41
  ABAP Program to Test the Consumer Proxy ............................................................................................41

Related Content ................................................................................................................................................43

Copyright...........................................................................................................................................................44
Introduction

Traditionally, in SAP PI 7.0 and SAP XI 3.0, when working with ABAP proxies, the SAP system where the ABAP proxy resides has to be configured as an application system connecting to only one Integration Server. When there is a need to integrate to another Integration Server, the server configuration will have to be made using tx: SXMB_ADM to point to this other Integration Server.

With the introduction of WebService runtime in the ABAP stack of Application Server 7.1 and Application Server 7.0 SP14, another option becomes available for us to work with ABAP proxy such that connections to multiple Integration Servers of SAP NW PI 7.1 are now possible.

Restrictions

WS-runtime, which enables this new configuration, is available with all Application Server ABAP 7.1 and Application Server ABAP 7.0 SP14+ systems. This means the consuming and provider proxies can be on either of these systems.

In addition, the Integration Server will have to be on a SAP NW PI 7.1 system. Only SAP NW PI 7.1 provides the WS adapter, which is required during the configuration of the end-to-end scenarios.

Requirements

Even though there may only be one consumer and one provider proxy involved, the configurations for the end-to-end scenarios will still need to be configured on every connected Integration Server. This means the designs for interface definitions, mappings, etc. will still have to be done for every involved Integration Server.

In addition, the proxy will still have to be generated on the perspective consuming or provider systems.
What is New?

In Application Server ABAP 7.1 and Application Server ABAP 7.0 SP14+, the ABAP proxy, specifically the consumer proxy, determines the endpoint (or which Integration Server to send the message) with two different methods:

1. Based on the application server’s configuration, which is done using tx: SXMB_ADM. As a result, when we wish to send the message to a different Integration Server, we will have to use this transaction to enter a new server address. This does not provide a lot flexibility and transparency. We would not know, through our application, which Integration Server will receive the message.

2. Based on webservice configuration, which is done using a new tx: SOAMANAGER. With this transaction, we can assign multiple ports and bindings for a single proxy. Each port and binding will point us to a different Integration Server. In the application, we will specify which port will be used for the integration. This method provides both flexibility and transparency. Therefore, if integration with multiple SAP PI 7.1 systems is required, using the new tx SOAMANAGER is the recommended option for configuration.
**Business Scenario**

To demonstrate the design and configuration requirement, I will use the following scenario.

The consumer proxy will be on a SAP 7.0 SP15 system. The provider proxy will be on a SAP 7.1 SP5 system. Two SAP NW PI 7.1 systems will be used to integrate the systems.

A simple synchronous interface will be used, so that a request/response can be shown easily during testing.

Below is a high-level view of our scenarios:
Design and Configuration Steps

The basic steps required for the complete design and configuration process are as follow:

1. Design the inbound and outbound interfaces and mappings in the Enterprise Services Repository (ESR). The design will be done on one of the SAP NW PI 7.1 systems.

2. In order to create the interfaces, export and import the design as external definitions to the ESR of another SAP NW PI 7.1 system, the IR of the PI 7.0 which the SAP backend application server 7.0 is connected to, and the ESR of the PI 7.1 which the SAP backend application server 7.1 is connected to.

   (Note: the design components, e.g. interfaces and mappings, in the ESR can also be transported to the different systems. But, this will require that the SWCVs are identical. In our example, each system has a different SWCV.)

3. Create the interfaces on all the IR and ESR systems using the external definitions.

4. Create mappings and all required design objects on the SAP NW PI 7.1 systems.

5. Generate the ABAP proxies on the consumer and provider systems.

6. Use tx: SOAMANAGER to configure the endpoints for the provider proxy.

7. Configure the objects in the ID of the SAP NW PI 7.1 systems, e.g. communication channel, receiver determinations, etc.

8. Use tx: SOAMANAGER to configure the ports/bindings for the consumer proxy.

9. Test the consumer proxy using an ABAP program.
1. Design Inbound and Outbound Interfaces and mappings for consumer and provider

A. Design the interfaces. Only the interfaces are shown here. The data type and message type are not shown here. The namespace used is http://test.com/ws/abap.

Consumer interface: consumer_Sync_Out

Provider interface: provider_Sync_Out
B. Design the message mappings. Develop request and response message mappings to concatenate the messages with the PI server id, e.g. XLI or M01. This way, in our test, we can determine which PI server received the consumer proxy request.

Request message mapping:

![Request message mapping diagram]

Change the value to reflect the PI system id where the mapping is being executed.
Response message mapping:

![Diagram of message mapping](image-url)
2. Export and import the interface design as external definitions

A. For each interface, export the WSDL to a local file:

B. Export the message mapping to an external file:

Since the configurations will have to be duplicated on all PI servers, we will also need to export the mappings to avoid efforts in re-design the mapping.

The message mapping can be export in the following steps:

1) Go to the message mapping tool and the “Test” tab in the tool.
Then, hit the keys, cntl-shift-0 (zero), concurrently. You should have the following screen.

Export the message mapping to a file.

The file exported is of type ".xim".

2) Perform the above for all the mappings.

3. Create the interfaces on all the IR and ESR systems using the external definitions

Import the WSDL files as external definitions on the perspective systems (PI and backend):

A. On XR9, for the consumer proxy, create a namespace identical to the one used in the previous step. The namespace can be created in any SWCV.

1) Create an external definition for the consumer, using the exported file from the previous step:
External Definition Name: consumer (any name can be used)
2) Select Category "wsdl"

Click on “Import External Definitions”

Select the file

Click on “Open”

Save the External Definition

3) Create the outbound interface:
Use the same name as the original interface, consumer_Sync_Out:
4) Select the mode as synchronous

Drag the external definition, consumer, to the “hand” and select the appropriate file depending on request or response message type.

5) The following contains the results:

6) Activate the objects.

B. On M01 (the other PI server), perform the same steps for both consumer and provider interfaces (consumer_Sync_Out and provider_Sync_In); and, on M62 (the provider system), perform the same steps for the provider interface (provider_Sync_In). Please make sure to use the namespace: http://test.com/ws/abap
4. Create mappings and all required design objects on the PI 7.1 systems

A. Import the exported message mapping files into the message mapping tool of the PI systems:

On M01, the other PI server’s ESR, create a new message mapping and use the same message mapping name as XLI.

1) Create a message mapping using the same name as the XLI PI server.
2) Drag-n-drop the external definition to the message type and select the appropriate request or response message type from the external definition.
3) Hit the keys, cntl-shift-0 (zero), concurrently. Enter the exported mapping file.

B. Create Operations Mapping using the interfaces and message mappings created above.

(Unfortunately, there are no import/export functions for this step.)
5. Generate the ABAP proxies on the consumer and provider systems

A. Generate the ABAP proxy for the provider on M62. Logon to the 002 client; this is the application server.

1) Enter transaction SPROXY.
3) Right-click on the provider interface and select “Generate”.
4) In the dialog box that followed:
   a) select “Local Object”
   b) enter the prefix, which can be anything: ZTST_
   c) click “Finish”
5) The following ABAP proxy will be created:

![Service Interface](image)

6) Save and activate the proxy.
7) Enter code for the provider proxy:
   a) Double-click on the provider class

![Interface](image)
b) Double-click on the method

![Method Interface](image)

**ZST II PROVIDER_SYNC_IN**

b) Double-click on the method

```plaintext
```

c) Insert the following code:

```
```

d) Save and activate the proxy.

B. Generate the ABAP proxy for the consumer on XR9. Logon to the 002 client; this is the application server.

Follow similar steps as generating the provider proxy. Except this time, it is the for consumer interface, consumer_Sync_Out. When the proxy is generated and activated, we will have the following:
6. **Use tx: SOAMANAGER to configure the endpoints for the provider proxy**

A. Configure the webservice for the provider proxy:

1) Logon to the provider system: M62, client 002.

2) Enter transaction: SOAMANAGER

3) The following web page will be display in the web browser.
   a) Click on the “Business Administration” tab.
   b) Click on “Web Service Administration” link.

4) The following web page will be display in the web browser.
   a) Select Search by: Service
   b) Enter Search Pattern: provider_Sync_In (this name is from the SPROXY display of the “External Key” of the provider interface.
   c) Select Field: External Name
   d) Clock “Go”.
   e) With the Search Result, select ZPROVIDER_SYNC_IN
   f) Click “Apply Selection”.

---

**Web Service Administration**

- **Search Design Time object for Web Service Configuration**
  - Search by: Service
  - Search Pattern: provider_Sync_In
  - Field: External Name
  - In System: M62/002
  - Clock “Go”.

**Search Results**

- ZPROVIDER_SYNC_IN
  - External Name: provider_Sync_In
  - Namespace: provider_Sync_In
  - Type: Service Interface
  - Description: http://test.com/soap
  - Direction: Inbound
  - Used Objects:
    - External Key
    - Service Interface
    - Namespace: provider_Sync_In
    - State
  - Properties:
    - Name
    - Namespace: provider_Sync_In
  - Description:
    - Inbound
  - Direction:
    - Inbound
  - Used Objects:
    - External Key
    - Service Interface
    - Namespace: provider_Sync_In
    - State
  - Properties:
    - Name
    - Namespace: provider_Sync_In
  - Description:
    - Inbound
  - Direction:
    - Inbound
  - Used Objects:
    - External Key
    - Service Interface
    - Namespace: provider_Sync_In
    - State
  - Properties:
    - Name
    - Namespace: provider_Sync_In
  - Description:
    - Inbound
  - Direction:
    - Inbound
  - Used Objects:
5) On the lower half of the screen, click on: Create Service

6) Enter any service and binding names for the webservice:
7) Again, on the lower half of the screen:
   a) select “User ID/Password” for Authentication.
   b) Save the configuration.
8) We should now have the following:

![Web Service Administration](image1)

9) When we go to the “Overview” tab, we can test the webservice using WSNavigator:

![Web Service Administration](image2)
10) The test using the WSNavigator will produce the following result:

![Service Information](image)

```
<V:Envelope xmlns:V="http://schemas.xmlsoap.org/soap/envelope/">
  <V:Body>
    ...
  </V:Body>
</V:Envelope>
```

11) Very importantly, we must note the WSDL URL for the webservice. We can copy-n-paste it to a temporary location. This URL will be needed in the Integration Directory configuration. The URL is:

```
```

- **Host**: iwdf4334.wdf.sap.corp
- **Port**: 50000
- **URL Path**:

```
```
7. Configure the objects in the ID of the PI 7.1 systems

The reason we are configuring the ID objects before configuring the consumer proxy webservice is because the WS sender communication channel contains information that will be needed during the consumer proxy configuration.

A. Create the following configuration objects in the Integration Directory on M01:

1) Consumer business component:

Add the outbound interfaces to the Sender.

![WS_Consumer: Display Communication Component](image)
2) Provider business component:

Add the inbound interfaces to the Receiver.

![WS_Provider: Edit Communication Component](image)

- **Name**: provider_Sync_In
- **Namespace**: http://test.com/wsabap
- **Software Component Version**: TRAIN_PI_STUDY_SYS1_0 of sap
3) **WS sender communication channel**: (consumer)

Enter system related information of the PI server, M01 (when configuring XLI, the XLI server information must be used):

![WS sender communication channel setup](image)
4) **WS receiver communication channel: (provider)**

![Diagram of WS receiver communication channel settings]

- **Adapter Type**: WS
- **Message Protocol**: HTTP 1.0
- **Adapter Engine**: Integration Server
- **Security Settings**: User Password (Authentication at HTTP Level, Changed to SAML Authentication if Required by Interface Operations)
- **Technical Transport Settings**: HTTPS (Changed to HTTPS if Required by Interface Operations)
- **Transport Protocol**: https://ws44334.wdf.sap.corp:443
- **Service Name/Port**: wsdl11/AllInOne/WS_PolicyDocument?soap-client=002
- **URL Access Path**: ?soap-client=002

*From the WSDL URL in step 6.3.11*
5) Sender agreement:

Obtain the WSDL URL for the Sender Agreement. This URL will be used in the configuration of the consumer proxy on XR9.

Menu: Sender Agreement → Display WSDL
Copy-n-paste the URL to a temporary location:

WSDL-URL:


6) Receiver agreement:
7) Interface determination:

![Interface determination diagram]

- **Sender**
  - Communication Party: WS_Consumer
  - Communication Component: consumer_Sync_Out
  - Interface: consumer_Sync_Out
  - Namespace: http://test.com/ws/abap

- **Receiver**
  - Communication Party: WS_Provider
  - Communication Component: consumer_Sync_Out
  - Description: Software Component Version of Sender Interface TRAIN_PL_Studio_SYS031.0 of sap
  - Maintain Order At Runtime

**Receiver Interfaces**:
- 2 interfaces:
  - **consumer_provider**
    - Condition: consumer_provider
    - Operation Mapping: provider_Sync_In
    - Namespace: http://test.com/ws/abap
    - Multiplicity: 1
  - 1 other interface
8) Receiver determination:

a)

<table>
<thead>
<tr>
<th>Receiver Determination Table</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sender</strong></td>
</tr>
<tr>
<td>Communication Party</td>
</tr>
<tr>
<td>Communication Component</td>
</tr>
<tr>
<td>Interface</td>
</tr>
<tr>
<td>Namespace</td>
</tr>
<tr>
<td><strong>Receiver</strong></td>
</tr>
<tr>
<td>Communication Party</td>
</tr>
<tr>
<td>Communication Component</td>
</tr>
<tr>
<td>Description</td>
</tr>
</tbody>
</table>

Software Component Version: TRAIN_PI_STUDXX_SYS3 1.0 of sap

Type of Receiver Determination: Standard

Configured Receivers

- Rule: LAW
- Condition: Communication Party
- Communication Component: WS_Provider

If no Receiver is found, proceed as follows:
- Error Message
- Ignore
- Select the following Receiver

C. Create the same configuration objects in the Integration Directory on XLI by following the same steps above.

The WSDL-URL for the Sender Agreement on XLI: (this URL will be used in the consumer proxy configuration on XR9)

8. **Use tx: SOAMANAGER to configure the ports/bindings for the consumer proxy**

   A. Configure the webservice for the consumer proxy:

      1) Logon to the consumer system, XR9 client 200, and execute transaction: SOAMANAGER

      2) Click on tab “Business Administration” and, then, “Web Service Administration”.

      3) Select the consumer proxy:
4) On the bottom half of the browser page, click on the “Configurations” tab and then “Create Logical Port”:

![Web Service Administration](image)

**Search Design Time object for Web Service Configuration**

<table>
<thead>
<tr>
<th>Search by: Consumer Proxy</th>
<th>Search Pattern: ZTST*</th>
<th>Field: Internal Name</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Internal Name</th>
<th>External Name</th>
<th>Namespace</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZTST_CO_CONSUMER_SYNC_OUT</td>
<td>consumer.Sync_Out</td>
<td><a href="http://test.com">http://test.com</a></td>
</tr>
</tbody>
</table>

**Details of Proxy Definition: ZTST_CO_CONSUMER_SYNC_OUT**

- [Create Logical Port](#)
- [Delete Logical Ports](#)
- [Edit](#)
- [Display](#)
5) Create the service and port for the PI systems:
   a) For XLI:
Save the configuration:

b) For M01: Repeat the above steps for creating the port for M01. Make sure the WSDL-URL is from step 7.A.5.
c) When complete, the following shows the ports:

![Web Service Administration](image)

9. Test the consumer proxy using an ABAP program

1) Logon to the consumer system, XR9 client 200.

2) Transaction: SE38

Create an ABAP program (can be any name): ZTST_WS_CONSUMER

Copy-n-paste the program in the appendix.

3) This program is very similar to any other ABAP proxy program. The primary difference is that the WS APIs are used to send the message. In this program, I have 2 input parameters: the message and which PI server to send the message.

![Test consumer for WS](image)

To determine which PI server to send the message, the following code is used:

```java
if XLI = 'X'.
  CREATE OBJECT CONSUMER
  EXPORTING
  logical_port_name = 'XLI_PORT'.
else.
  CREATE OBJECT CONSUMER
  EXPORTING
  logical_port_name = 'M01_PORT'.
endif.
```

As you can see, XLI_PORT and M01_PORT are what we configured in the SOAMANAGER for the consumer proxy in step 8.E.
4) When the executed the program using:

**With XLI:**

![Test consumer for WS](image)

We get the following result:

**With M01:**

![Test consumer for WS](image)

We get the following result:

5) The messages can also be monitored on the XLI and M01 integration server.
Appendix

ABAP Program to Test the Consumer Proxy

Below is the listing of the ABAP program:

Please note the lines which checks for the parameter which will instruct the program to send the message to the desired PI server. XLI_Port and M01_Port are the different port names used to determine which PI server will receive the request.

```abap
REPORT ZTST_WS_CONSUMER.
DATA: CONSUMER TYPE REF TO ZTST_CO_CONSUMER_SYNC_OUT,
     m_seq_prot TYPE ref to IF_WS_PROTOCOL_SEQUENCE,
     m_seq TYPE ref to IF_WS_CLIENT_SEQUENCE,
     l_wsprot type ref to if_wsprotocol,
     lv_seq TYPE SRT_SEQ_ID.
data: l_request type ZTST_REQ_MT.
data: l_response type ZTST_RESP_MT.

parameters: MESSAGE(30) type c.
parameters: XLI radiobutton group RAD1 default 'X',
            M01 radiobutton group RAD1.

l_request-REQ_MT-MSG = MESSAGE.

try.
  if XLI = 'X'.
    CREATE OBJECT CONSUMER
    EXPORTING
      logical_port_name = 'XLI_PORT'.
  else.
    CREATE OBJECT CONSUMER
    EXPORTING
      logical_port_name = 'M01_PORT'.
  endif.

  generation of the sequence protocol and the sequence
  m_seq_prot ?= CONSUMER->get_protocol( if_wsprotocol=>sequence ).
  m_seq = m_seq_prot->create_persistent_sequence( ).

  start sequencing and get id
  m_seq->begin( ).
  m_seq_prot->set_client_sequence( m_seq ).
  lv_seq = m_seq->get_id( ).

  CALL METHOD CONSUMER->CONSUMER_SYNC_OUT
  EXPORTING
    OUTPUT = l_request
```
IMPORTING
  INPUT = l_response.

  end sequencing and commit work
  m_seq->end().
  cl_soap_tx_factory=>commit_work().

  write:/ 'Msg send:'.
  write:/ 'Response: ', l_response-RESP_MT-MSG.
CATCH CX_AI_SYSTEM_FAULT
  CX_AI_APPLICATION_FAULT .
  write : 'Error during proxy call'.
  exit.
ENDTRY.
Related Content

Demo for Design and Configuration of Direct Connection in SAP NetWeaver PI 7.1
SAP NetWeaver Process Integration 7.1 - Overview
SAP NetWeaver Process Integration 7.1 - Overview of New Capabilities

For more information, visit the Service Bus-based Integration homepage.