

Java Connector CookBook

Version 1.1

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION.....	6
1.1 WHAT IS A COOKBOOK?.....	6
1.2 WHO CAN USE THIS COOKBOOK?.....	6
1.3 PRE-REQUISITES	6
1.4 HOW TO USE THIS COOKBOOK.....	7
CHAPTER 2 DEPLOYMENT OF SAP GENERIC CONNECTORS.....	8
2.1 STEPS FOR DEPLOYING SAP GENERIC CONNECTOR.....	8
CHAPTER 3 DEPLOYMENT FOR SP GENERIC CONNECTORS.....	12
3.1 STEPS FOR DEPLOYING SP GENERIC CONNECTOR.....	13
CHAPTER 4 IMPLEMENTATION & DEPLOYMENT OF SAP APPLICATION CONNECTOR	16
4.1 STEPS FOR DEPLOYING SAP APPLICATION SPECIFIC CONNECTOR.....	16
CHAPTER 5 IMPLEMENTATION & DEPLOYMENT OF SP APPLICATION CONNECTOR	21
5.1 STEPS FOR DEPLOYING SP APPLICATION SPECIFIC CONNECTOR.....	22
CHAPTER 6 CONNECTORS IN SSDG ARCHITECTURE	28
6.1 REAL WORLD APPLICATION-MAHARASHTRA INCOME CERTIFICATE	28
6.2 SCHEMA FOR REGISTRATION REQUEST AND RESPONSE FOR SYNCHRONOUS REQUEST	30
6.3 SCHEMA FOR CERTIFICATE REQUEST AND RESPONSE FOR ASYNCHRONOUS REQUEST	34
6.4 EXECUTING SYNCHRONOUS SUBMIT REQUEST AT SAP END	38
<i>Recipe 1 Synchronous Submit Request Execution.....</i>	<i>38</i>
6.5 PROCESSING OF E-SERVICES REQUEST AT SP END SENT IN SYNCHRONOUS MODE	44
<i>Recipe 2 Retrieve Request from SSDG in case of Synchronous Submit Request</i>	<i>44</i>
6.6 RETRIEVING THE RESULT SENT BY SP APPLICATION TO SAP	52
<i>Recipe 3 Retrieve Result from SSDG in case of Synchronous Submit Request</i>	<i>52</i>
6.7 EXECUTING ASYNCHRONOUS SUBMIT REQUEST.....	56
<i>Recipe 4 Asynchronous Submit Request Execution.....</i>	<i>56</i>
6.8 RETRIEVING ACKNOWLEDGEMENT FROM SSDG.....	64
<i>Recipe 5 Retrieve the Acknowledgement from SSDG.....</i>	<i>64</i>
6.9 PROCESSING OF E-SERVICES REQUEST AT SP END SENT IN ASYNCHRONOUS MODE.....	68
<i>Recipe 6 Retrieve Request from SSDG in case of Asynchronous Submit Request</i>	<i>68</i>
6.10 SUBMITTING THE RESPONSE FROM SP APPLICATION TO SSDG	76
<i>Recipe 7 Submit Response from SP Application to SSDG for Asynchronous Request.....</i>	<i>76</i>
6.11 POLLING THE SSDG TO RETRIEVE THE RESULT	83
<i>Recipe 8 Poll the SSDG to retrieve the result submitted by SP Application</i>	<i>83</i>
6.12 RETRIEVE THE RESULT OF SUBMIT POLL METHOD.....	90
<i>Recipe 9 Retrieve the result of Submit Poll method</i>	<i>90</i>
6.13 DELETING REQUEST SUBMITTED TO SSDG	106
<i>Recipe 10 Executing Delete Request.....</i>	<i>106</i>
6.14 LISTING REQUESTS SUBMITTED TO SSDG	109
<i>Recipe 11 Executing List Request for Submit Response.....</i>	<i>109</i>
<i>Recipe 12 Executing List Response for Submit Request.....</i>	<i>111</i>
ADDITIONAL RESOURCES	114

Table Of Figures

<i>Figure 1 : Creating a new Project for implementing SAP Application Connector</i>	<i>8</i>
<i>Figure 2 : Creating the new project as a Java Project.....</i>	<i>9</i>
<i>Figure 3 : Naming the new project as e Services.....</i>	<i>10</i>
<i>Figure 4 : Deploying the Response Component</i>	<i>13</i>
<i>Figure 5 : Deploying the Response Component</i>	<i>14</i>
<i>Figure 6 : Testing the deployment in browser</i>	<i>15</i>
<i>Figure 7 : Creating a new Project for deploying SAP Application Connector</i>	<i>16</i>
<i>Figure 8 : Creating the new project as a Java Project.....</i>	<i>17</i>
<i>Figure 9 : Naming the new project as e Services.....</i>	<i>18</i>
<i>Figure 10 : SAP Application Specific Connector Code</i>	<i>20</i>
<i>Figure 11 : Creation of New Java Project.....</i>	<i>22</i>
<i>Figure 12 : Naming the Java Project as SPeServices.....</i>	<i>23</i>
<i>Figure 13 : Importing from File provided by CDAC.....</i>	<i>24</i>
<i>Figure 14 : Importing file system.....</i>	<i>25</i>
<i>Figure 15 : Importing for writing SP Application Specific Code</i>	<i>26</i>
<i>Figure 16 : Implement SPConnectorClientBean</i>	<i>27</i>
<i>Figure 17 : SAP calling e-services application in synchronous mode</i>	<i>39</i>
<i>Figure 18 : SAP asynchronously submits request to e-services application through SSDG..</i>	<i>57</i>
<i>Figure 19 : E-services application submitting response to SSDG for Asynchronous Request</i>	<i>82</i>
<i>Figure 20 : SAP Polling SSDG for response</i>	<i>83</i>
<i>Figure 21 : SAP calling delete request for previously submitted requests.</i>	<i>108</i>
<i>Figure 22 : List Request to track the status of all requests and response from SSDG</i>	<i>111</i>

Table of Code Snippets

<i>Snippet 1 : Schema for Registration Request</i>	32
<i>Snippet 2 : Schema for Registration Response</i>	33
<i>Snippet 3 : Schema for Income Certificate Request</i>	35
<i>Snippet 4 : Schema for Income Certificate Response</i>	36
<i>Snippet 5 : AynchronousSubmitRequest.xsd</i>	37
<i>Snippet 6 : HTML code of Registration Form for one time registration</i>	39
<i>Snippet 7 : Code for RegistrationServlet.java</i>	42
<i>Snippet 8 : Code for Synchronous Submit Request</i>	44
<i>Snippet 9 : Code for SPApplicationConnector for Synchronous Submit Request</i>	51
<i>Snippet 10 : Code for retrieving response for Synchronous Submit Request</i>	53
<i>Snippet 11 : Code for RegistrationServlet.java</i>	56
<i>Snippet 12 : Income Certificate .html</i>	58
<i>Snippet 13 : IncomeCertApplication</i>	61
<i>Snippet 14 : ASynchronous Submit Request</i>	63
<i>Snippet 15 : Retrieving Acknowledgement for Asynchronous Submit Request</i>	67
<i>Snippet 16 : Code for SPConnector Client.java</i>	76
<i>Snippet 17 : Sending Response from SP Side for Asynchronous Submit Request</i>	79
<i>Snippet 18 : Code for submitting Response from SP to SSDG</i>	82
<i>Snippet 19 : PollInfo.html</i>	84
<i>Snippet 20 : Code for CheckStatus.java</i>	88
<i>Snippet 21 : Submit Poll Operation</i>	90
<i>Snippet 22 : Asynchronous Poll Response Schema</i>	100
<i>Snippet 23 : Code for CheckStatus.java</i>	105
<i>Snippet 24 : Code for Delete Request</i>	108
<i>Snippet 25 : List Request for Submit Response</i>	111
<i>Snippet 26 : List Request for Submit Request</i>	113

Version details:

Date	Changes	Revision	Revised by
11Jan2012	Technology and version details are added. In discussion Response status points are added.	1.1	NSDG Team CDAC Mumbai
10Mar2010	Not applicable	1.0	NSDG Team CDAC Mumbai

Chapter 1 Introduction

NOTE: The Cook Book document for NSDG and SSDG are same and the term Gateway refers to both NSDG as well as SSDG.

1.1 What is a CookBook?

Typically, a cookbook is a collection of recipes or instructions that explain how to perform a certain task and what is needed to execute it. The basic objective of this Cookbook is to provide an essential problem-solving resource. Just as cookbooks for meals contain step-by-step directions for creating various dishes, this book provides recipes for using SAP and SP generic connector. Recipes are provided for quick reference about invoking the functionalities provided by generic connectors and use of these functionalities by application specific connector.

While deploying the SAP and SP application connectors, the developer is likely to face certain queries or problems. This cookbook tries to span all such problems that the developer may face during the phase of implementing the application connectors along with the solutions for these problems. This cookbook is a collection of code solutions to common and uncommon problems encountered while implementing the application specific connectors in J2EE framework.

1.2 Who can use this CookBook?

Developers intending to build and deploy the application connectors in J2EE framework may refer to the cookbook for finding solutions to commonly encountered problems. This cookbook is not meant to guide / instruct a developer about building a EJB specific application connector. Developers may seek reference from the Java Connector Manual for instructions on developing the application connectors in J2EE framework.

1.3 Pre-requisites

This book is intended for developers with at least some development experience in EJB programming and J2EE framework. It is assumed that the reader understands the concepts of enterprise development and the basics of J2EE framework and has gone through the Java Connector Development manual. The format used here specifically references problems and issues, avoiding the use / explanation of Java/EJB keywords This manual can be read up at any point.

Technology Details:

Technology	version
jdk	1.6
SAP Generic Connector	SapConnectorVer2.jar
SP Generic Connector	SPGenericConnectorVer2.aar
Async Response Component	AsyncResponseComponentVer2.jar

Application Server - jboss4.3.

1.4 How to use this CookBook

This cookbook spans a total of 6 chapters each exploring and resolving the issues faced during implementing and working of the application specific connectors.

NOTE : This cook book contains only sample code for understanding purpose. User needs to change the request and response schema as well as the business logic as per their requirements.

Chapter 1: Provides an introduction to what a CookBook essentially is and who can use this CookBook. It gives an overview about how the cookbook can be useful in solving some common issues faced by developers in the process of using application connectors with the generic connectors.

Chapter 2: Extensively covers the deployment of SAP generic connectors in a step-by-step approach.

Chapter 3: Explains the deployment of SP generic connectors in a sequential manner. Also testing of SP generic connector is conducted to ensure that it is up and running by deploying it in JBoss Application Server.

Chapter 4: Demonstrates the implementation and deployment of SAP application connector.

Chapter 5: Demonstrates the implementation and deployment of SP application connector

Chapter 6: Provides a real world application example of how request and its subsequent response is communicated between the SAP and SP generic and application specific connectors. Elaborates the flow of request and response on both SAP and SP side for synchronous and asynchronous mode. Also discusses various issues encountered while handling these requests and response and provides solutions to these issues in the form of recipes.

Chapter 2 Deployment of SAP Generic Connectors

SAP generic connector is a Java Archive (jar) file and needs to be added in the project in which SAP implements application specific connector. Deployment of application specific connector will be as per requirements of the SAP application. The following are the steps for deploying SAP generic connector.

2.1 Steps for deploying SAP Generic Connector

- i) Create a new Java Project for implementing SAP application specific connector as shown in *Figure 1*.

Select File->New->Project

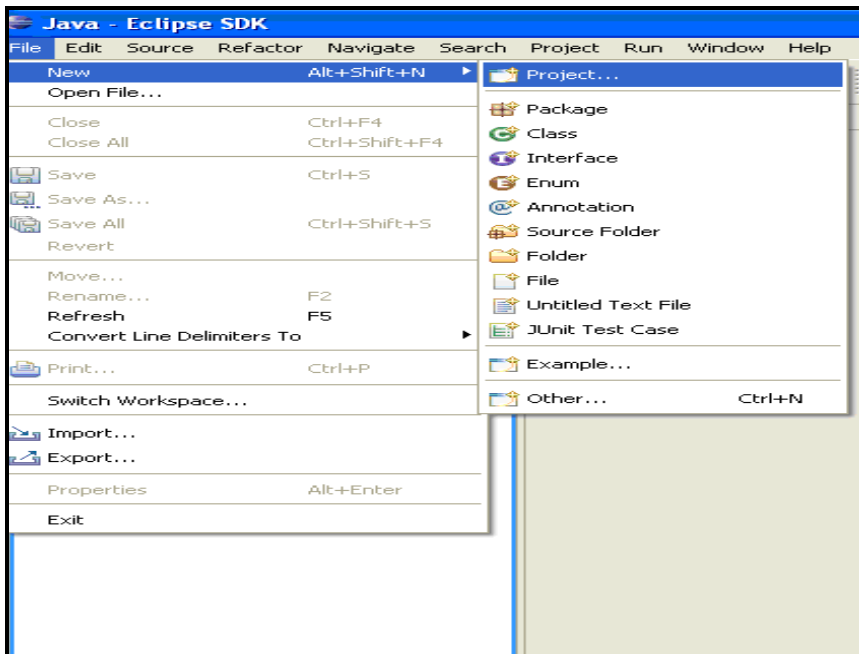


Figure 1 : Creating a new Project for implementing SAP Application Connector

Select Java Project as shown in *Figure 2*.

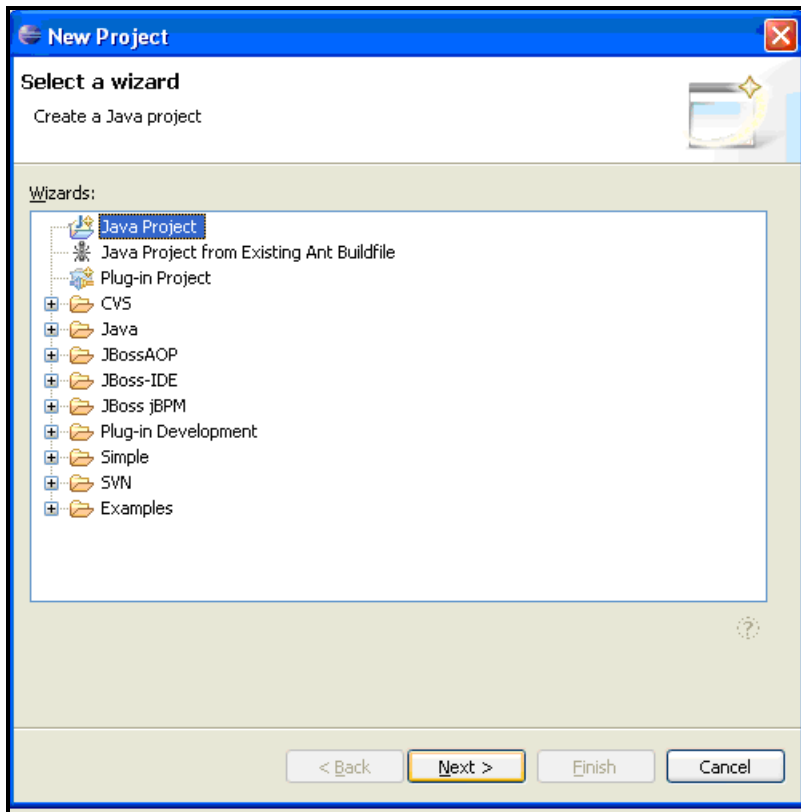


Figure 2 : Creating the new project as a Java Project

ii) Name the Project as eServices as shown in *Figure 3*.

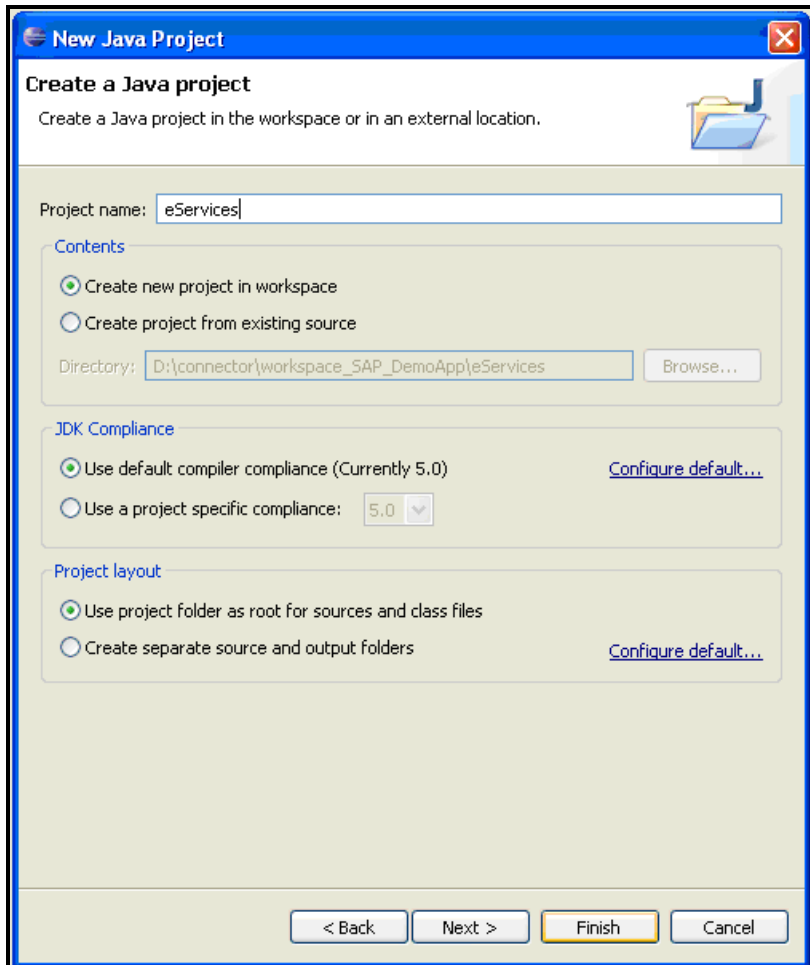


Figure 3 : Naming the new project as e Services

- iii) In the Build Path of this java project, add the following jar files which will be provided by CDAC ;
- SapConnectorVer2.jar
 - Libraries in the form of jar files of Axis2 1.5 and XmlBeans 2.2 onwards
- iv) Compile this project after adding the above files.

Chapter 3 Deployment for SP Generic Connectors

The SP end has the following 2 components that take care of the incoming requests and serve the appropriate response.

SP End Components

- SP Core Component – for processing synchronous and asynchronous submit requests
- SP Response Component – for sending asynchronous submit response

This section explains in a stepwise manner how SP generic connector is deployed in JBoss application server. The snapshots of the deployment are provided for a clear understanding.

3.1 Steps for deploying SP Generic Connector

- i) Copy the axis2.war and SP Response Component (named as AsyncResponseComponentVer2.jar) provided by CDAC in %JBoss installation path%default\deploy folder as shown in Figure 4.

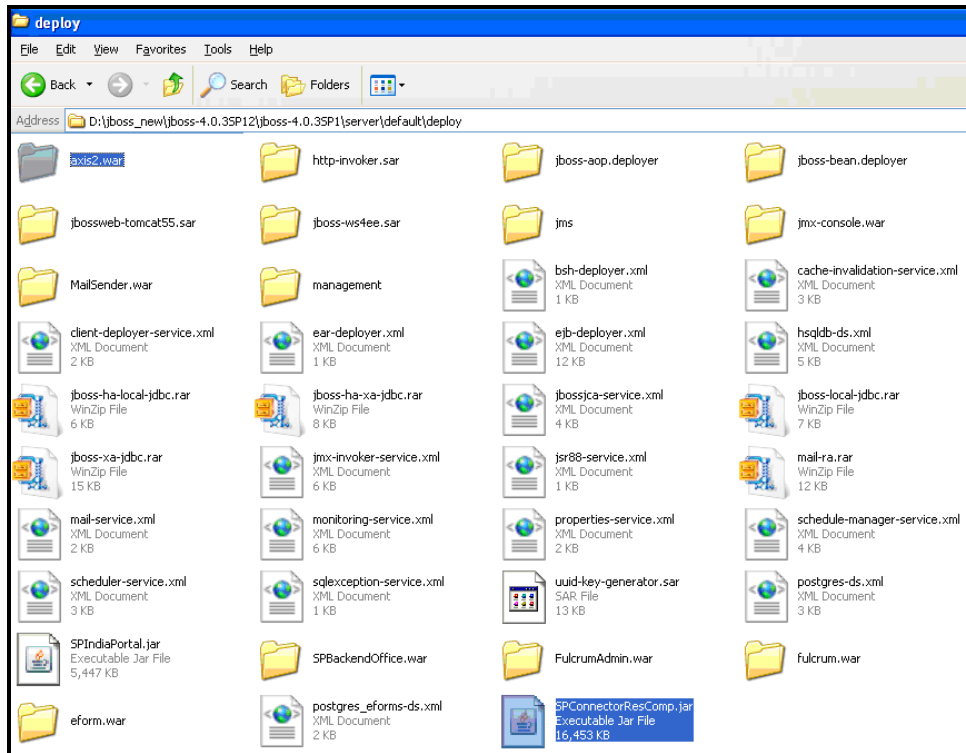


Figure 4 : Deploying the Response Component

- ii) SP Core Component is in the form of web service named as SPGenericConnectorVer2.aar. Copy this in **%JBoss installation path%default\deploy\axis2.war\WEB-INF\services folder** as shown in *Figure 5*

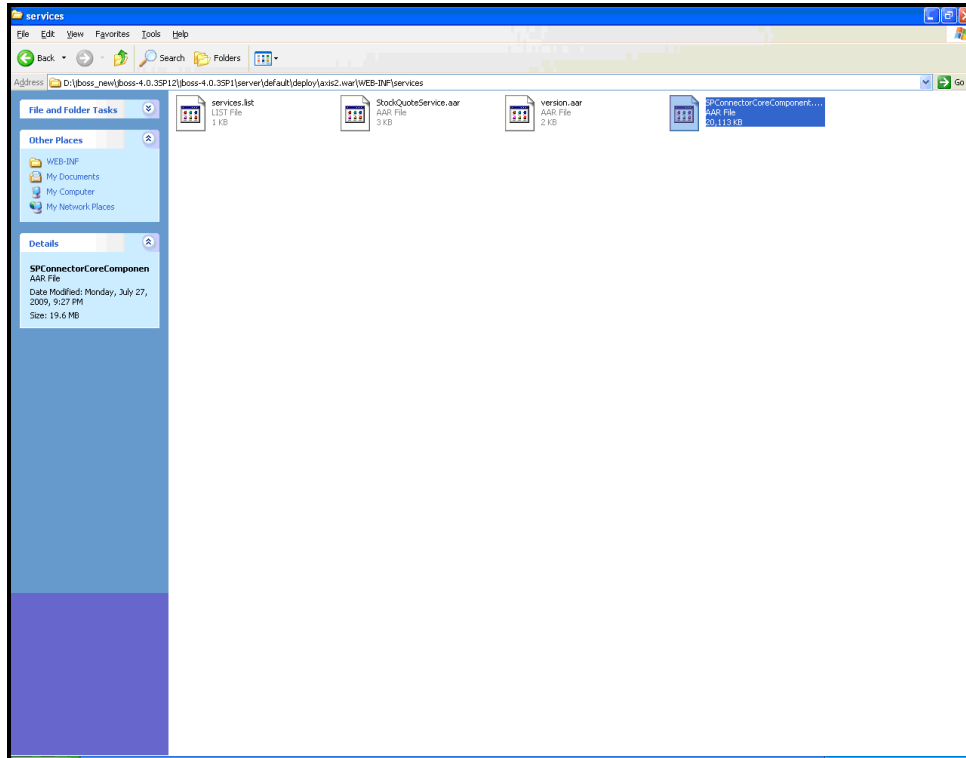


Figure 5 : Deploying the Response Component

- iii) Run JBoss.
- iv) Test the deployment in the browser as shown in *Figure6*

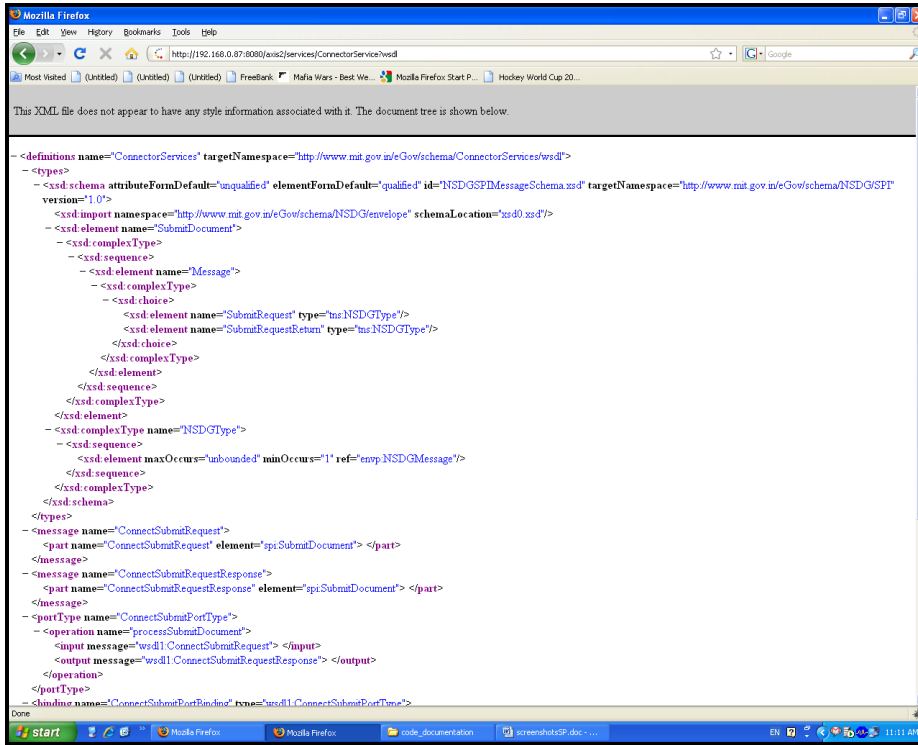


Figure 6 : Testing the deployment in browser

Chapter 4 Implementation & Deployment of SAP Application Connector

The SAP application specific connector will use the jar file of SAP generic connector.

4.1 Steps for deploying SAP Application Specific Connector

- i) Create a new Java Project for implementing SAP application specific connector as shown in *Figure 6*.

Select File->New->Project

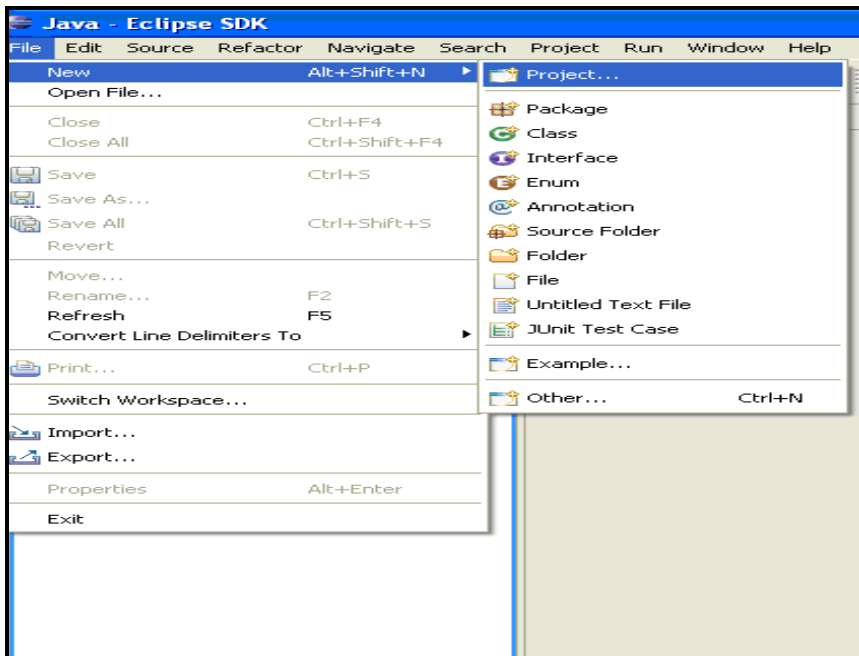


Figure 7 : Creating a new Project for deploying SAP Application Connector

Select Java Project as shown in *Figure 7*.

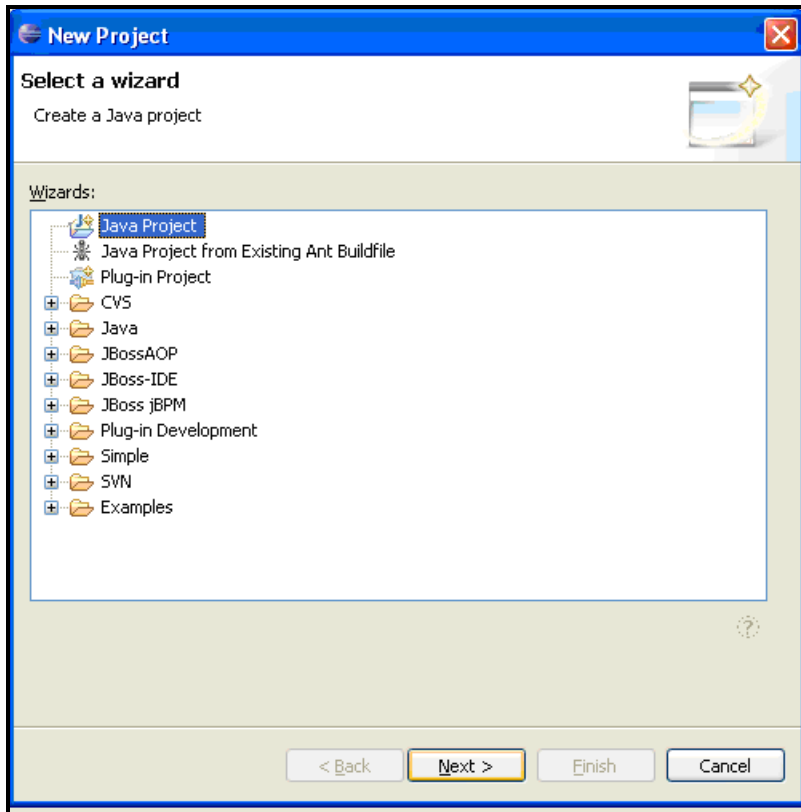


Figure 8 : Creating the new project as a Java Project

ii) Name the Project as eServices as shown in *Figure 8*.

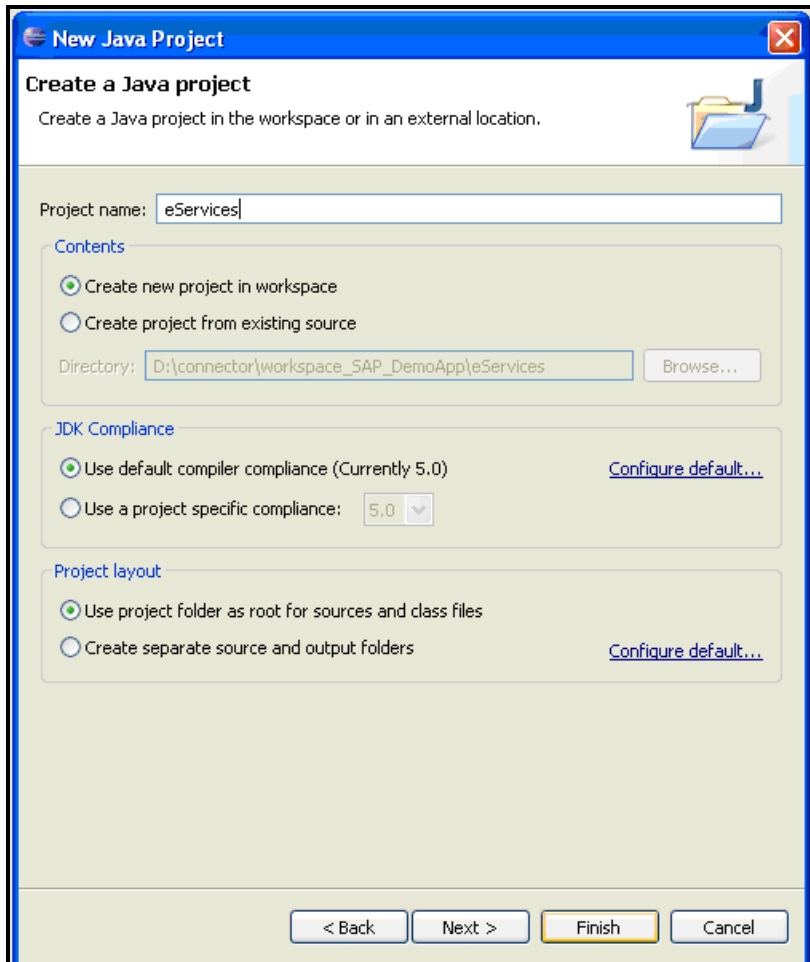


Figure 9 : Naming the new project as e Services

- iii) Right click on java project-> Select Build Path. In the Build Path of this java project, add the following jar files which will be provided by CDAC ;

- Libraries in the form of jar files of Axis2 1.5 and XmlBeans 2.2.
 - SapConnectorVer2.jar
- iv) Compile this project after adding the above files.
- v) As shown in Figure 9 below, SAP application specific connector code can be written in java package *in\cdacmumbai\eservices\client* .Depending on the requirements of SAP application, one can refer the recipes given in Chapter 6.

```
Java - RegistrationServlet.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer Hierarchy
  Services
    src
      in.cdacmumbai.eservices.client
    JRE System Library [jre6]
    J2EE 1.4 Libraries (JBoss-IDE)
    archive
    lib
    pages
    RegistrationForm.html
    WEB-INF
      classes
      lib
      web.xml
    packaging-build.xml

RegistrationServlet.java RegistrationBean.java

package in.cdacmumbai.eservices.client;

import java.io.IOException;

public class RegistrationServlet extends HttpServlet {

    @Override
    public void init(ServletConfig config) throws ServletException {
        super.init(config);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

    }

    public void destroy() {

    }
}
```

Figure 10 : SAP Application Specific Connector Code

Chapter 5 Implementation & Deployment of SP Application Connector

The SP application specific connector will use SP generic connector. For implementing SP Application Specific connector follow the steps given in section 3.1 and check that SP generic connector is deployed.

Deployment of SP Application Specific Connector

This section discusses how the SP application specific connector is implemented. SP generic connector calls the application specific connector on SP side with prescribed two interfaces namely *callSyncSubmitRequest()* and *callAsyncSubmiRrequest()*. SP application specific connector should be present in the form of EJB jar. EJB name should be *SPConnectorClient Remote* and should contain the above 2 methods. The implementation of these methods needs to be done in the bean class named *SPConnectorClientBean* whereas the remote class will contain only the method definition. This section explains the deployment of the SP application specific connector in a step by step manner along with the screen shots for every step.

5.1 Steps for deploying SP Application Specific Connector

Following are the steps for importing the SP Specific Connector Project.

- i) Open your eclipse and go to File => New => Project

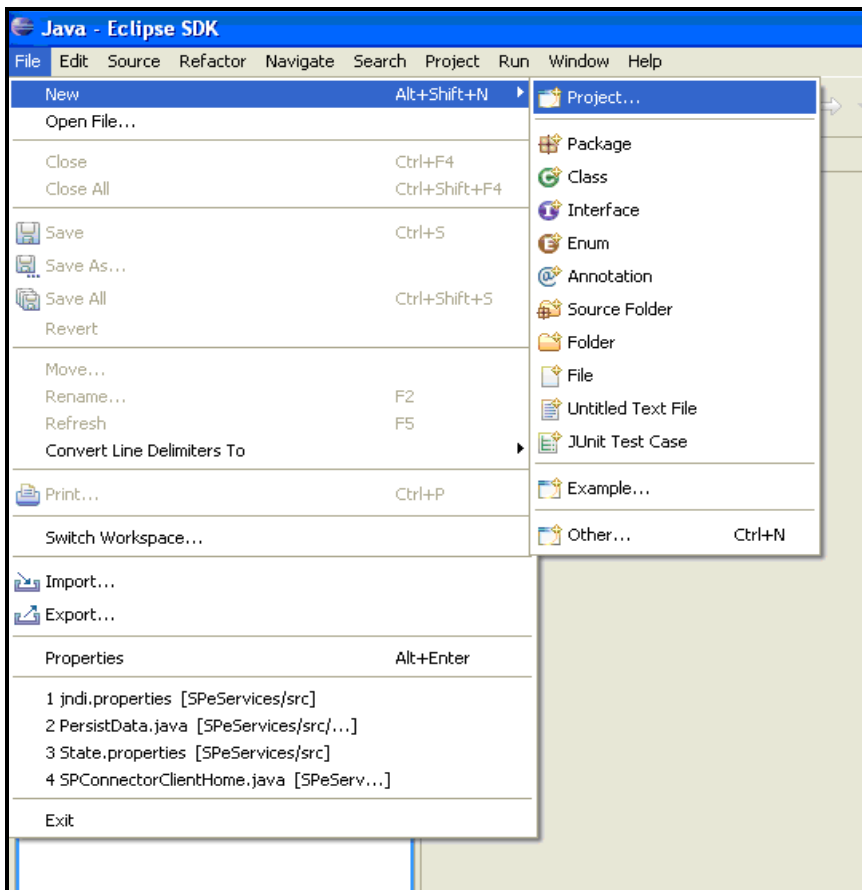


Figure 11 : Creation of New Java Project

- ii) Give the Project Name and select the option “Create new project in workspace”
Then click on finish

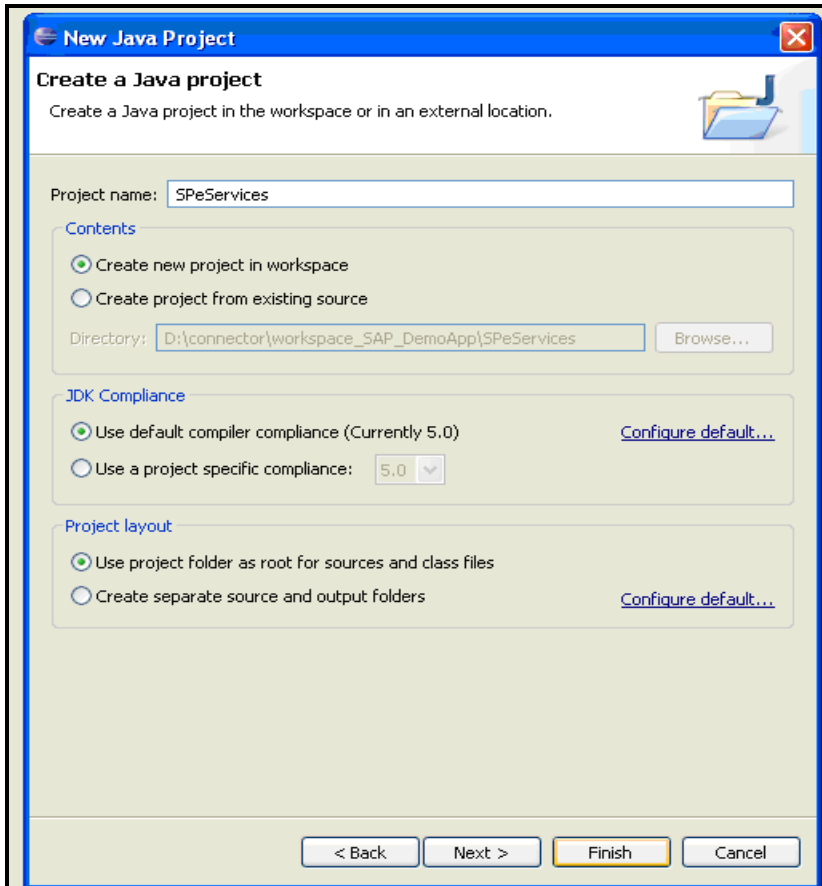


Figure 12 : Naming the Java Project as SPeServices

- iii) Right Click on the project in your Project Explorer window and select “Import”

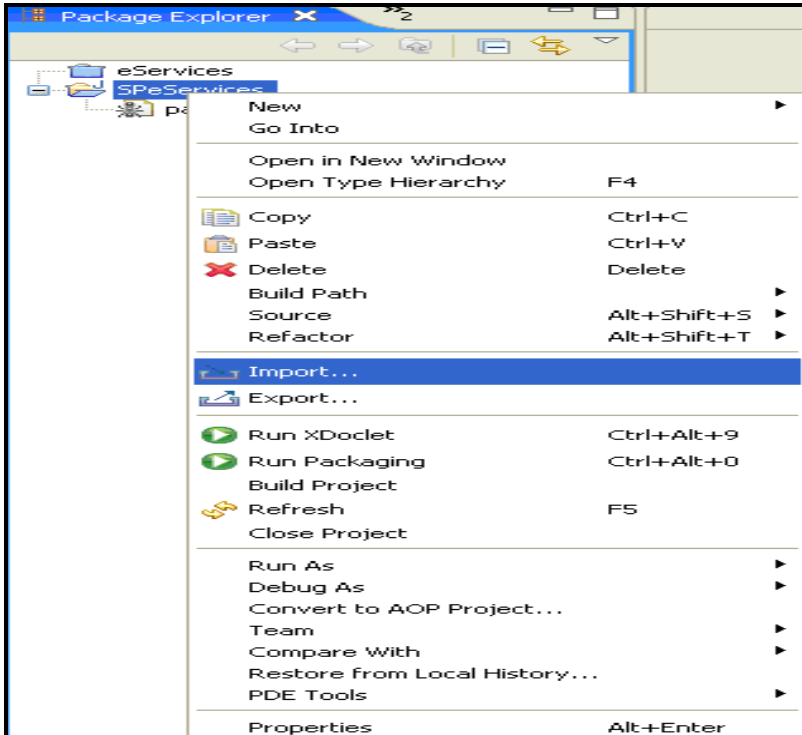


Figure 13 : Importing from File provided by CDAC

- iv) Select “FileSystem” from your Import window and then click “Next”

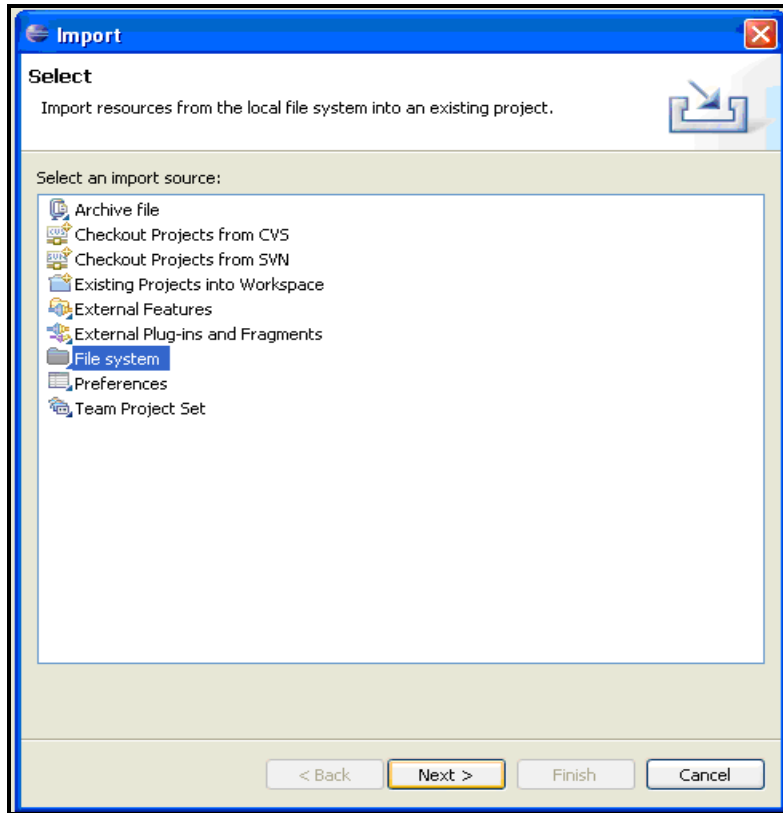


Figure 14 : Importing file system

- v) Select the folder from your filesystem of SP Specific Connector project given by CDAC

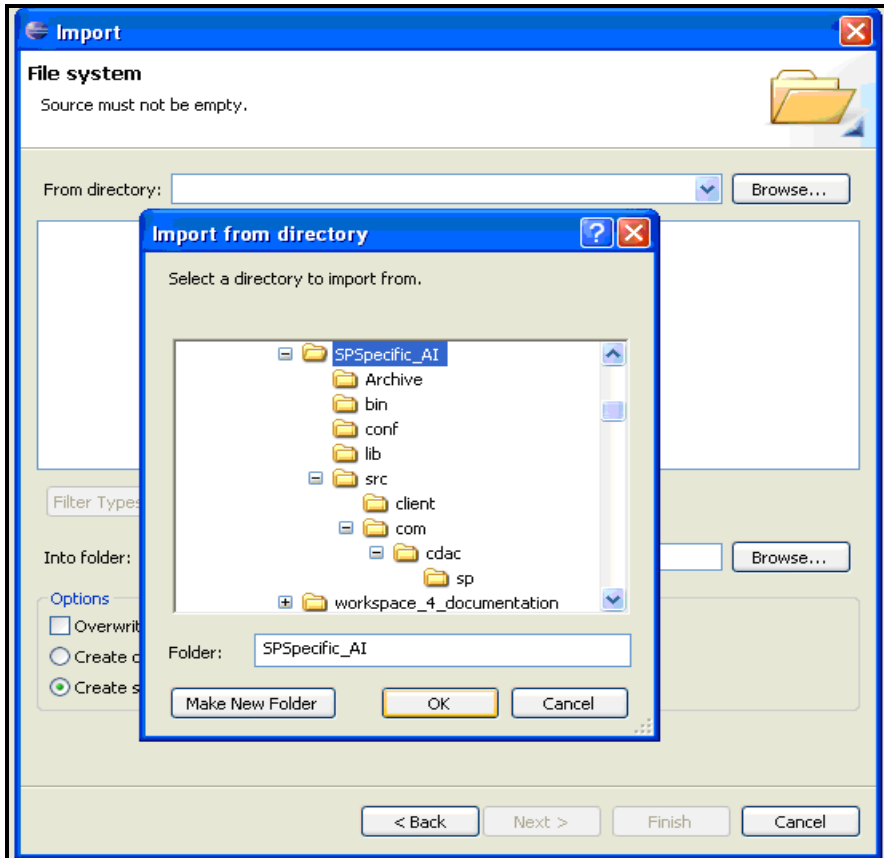


Figure 15 : Importing for writing SP Application Specific Code

- vi) This will be what you get after successfully importing the project.

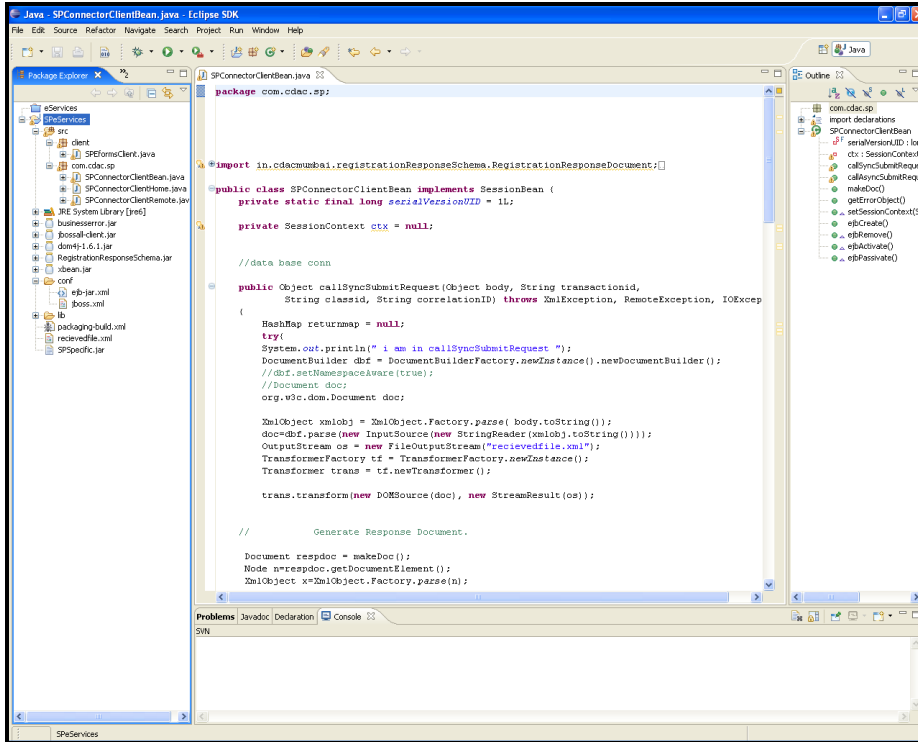


Figure 16 : Implement SPConnectorClientBean

Note: Code used specific to body are dummy, it can vary from application to application and implementation as per business logic of SP Application.

Chapter 6 Connectors in SSDG Architecture

This section elaborately discusses how the SAP and SP connectors interact with SSDG to exchange requests and relevant response. The classes and methods pertaining to the SAP and SP connectors are extensively explained along with the code. Also, there is a comprehensive discussion as to how the request travels to the SSDG through SAP side connectors finally reaching the SP application, and how response for the same is submitted to SAP through SSDG in case of synchronous and asynchronous requests.

A real world example is discussed in the following part where all scenarios pertaining to the connectors and their functioning are discussed in detail along with code snippets.

6.1 Real World Application-Maharashtra Income Certificate

In this cookbook, different scenarios pertaining to connector implementation for generating Maharashtra income certificate are discussed.

The Maharashtra Income Certificate application provides the certificate of income to all citizens who register for it.

The citizen registers for the various e-services provided by the government, the Maharashtra Income Certificate being one of them. After registration, the citizen receives a unique Registration Id. using which the citizen will apply for the income certificate. After due processing the citizen will receive the income certificate.

The citizen registers himself using the registration form of SAP. This registration form contains all details of the citizen. This form is submitted to SSDG, viz the SAP application and generic connectors. SSDG will then pass the request to relevant SP through SP generic and application connectors. SP will process the form by validating details of the citizen and will immediately generate a unique Registration Id. and send the same to SAP through SSDG. Here the exchange of request and response is synchronous.

Once the citizen receives the unique Registration Id., he can avail all any service. In this cookbook the scenario for obtaining Maharashtra Income Certificate is discussed. The citizen fills up the registration form for obtaining the certificate. The citizen also fills up the Registration Id. along with other details for obtaining the income certificate. This form is then submitted to SSDG using the SAP application and generic connectors. Upon submission, the citizen will receive a unique form ID. As this exchange of request and response is asynchronous, the citizen will not receive the income certificate immediately. The citizen uses

the unique form ID to track the income certificate. When the relevant income certificate is generated by SP, it is submitted to SSDG using SP generic and application connectors. Meanwhile, the citizen keeps polling the SSDG intermittently for the income certificate. Once the certificate has been submitted by SP to SSDG, the certificate is served to the citizen when he next polls for it.

Communication scenarios for Synchronous Request, Asynchronous Request and Inter-Gateway request submission are discussed. Also, how the response for these requests is received is detailed. Various gateway operations like Poll Request, Delete Request and List Request are also explained. SAP in this case is portal from where user/citizen keys in the data which is converted as per the request schema. Whereas the SP is an backend application processing the request and generating the response.

6.2 Schema for Registration Request and Response for Synchronous Request

Schema for Registration Request

XML Generated using this schema will be set in the Body Element of Submit Request document generated by SAP generic connector.

```
<?xml version="1.0" encoding="UTF-8" ?>
<schema id="RegistrationForm" version="1.0"
  targetNamespace="http://eservices.cdacmumbai.in/RegistrationForm"
  elementFormDefault="qualified" xmlns="http://www.w3.org/2001/XMLSchema">
  <element name="Form"
  <complexType>
  <sequence>
  <element name="Content" maxOccurs="1" minOccurs="1">
  <complexType>
  <sequence>
  <element name="Name">
  <simpleType>
  <restriction base="string">
  <minLength value="1" />
  <maxLength value="99" />
  <pattern value="([a-zA-Z .])*" />
  </restriction>
  </simpleType>
  </element>
  <element name="ShortName">
  <simpleType>
  <restriction base="string">
  <minLength value="1" />
  <maxLength value="99" />
  <pattern value="([a-zA-Z .])*" />
  </restriction>
  </simpleType>
  </element>
  <element name="GuardianType">
  <simpleType>
```

```
<restriction base="string">
<minLength value="1" />
<maxLength value="50" />
</restriction>
</simpleType>
</element>
<element name="GuardianName">
<simpleType>
<restriction base="string">
<minLength value="0" />
<maxLength value="99" />
<pattern value="([a-zA-Z.])*" />
</restriction>
</simpleType>
</element>
<element name="DateOfBirth">
<simpleType>
<restriction base="string">
<minLength value="8" />
<maxLength value="10" />
</restriction>
</simpleType>
</element>
<element name="Gender">
<simpleType>
<restriction base="string">
<minLength value="1" />
<maxLength value="2" />
</restriction>
</simpleType>
</element>
<element name="Nationality">
<simpleType>
<restriction base="string">
<minLength value="1" />
<maxLength value="50" />
<pattern value="([a-zA-Z ])*" />
</restriction></simpleType>
</element>
<element name="PermanentCountry">
<simpleType>
<restriction base="string">
<minLength value="0" />
<maxLength value="30" />
```

```
</restriction>
```

```
</simpleType>
</element>
<element name="PermanentState">
<simpleType>
<restriction base="string">
<minLength value="1" />
<maxLength value="70" />
</restriction>
</simpleType>
</element>
<element name="PermanentDistrict">
<simpleType>
<restriction base="string" />
</simpleType>
</element>
<element name="PermanentSubDistrict">
<simpleType>
<restriction base="string" />
</simpleType>
</element>
<element name="PermanentCity">
<simpleType>
<restriction base="string">
<minLength value="1" />
<maxLength value="50" />
</restriction>
</simpleType>
</element>
<element name="PermanentPinCode">
<simpleType>
<restriction base="string">
<length value="6" />
<pattern value="[0-9]{6}" />
</restriction>
</simpleType>
</element>
</sequence>
</complexType>
</element>
</sequence>
<attribute name="FormName" type="string" />
<attribute name="ParentFormName" type="string" /> </complexType>
</element>
</schema>
```

Snippet 1 : Schema for Registration Request

Schema for Registration Response



XML Generated using this schema will be set in the Body Element of Submit Response document generated by SP generic connector.

```
<?xml version="1.0" encoding="UTF-8" ?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.cdacmumbai.in/RegistrationResponseSchema"
xmlns:tns="http://www.cdacmumbai.in/RegistrationResponseSchema"
elementFormDefault="qualified" version="1.0" attributeFormDefault="unqualified"
id="RegistrationResponseSchema">
<element name="RegistrationResponse">
<complexType>
<sequence>
<element name="applicationId" type="string" />
<element name="applicantName" type="string" />
<element name="responseStatus" type="string" />
<element name="responseDetails" type="string" />
</sequence>
</complexType>
</element>
</schema>
```

Snippet 2 : Schema for Registration Response

6.3 Schema for Certificate Request and Response for Asynchronous Request

Schema for Income Certificate Request

XML Generated using this schema will be set in the Body Element of Submit Request document generated by SAP generic connector.

```
<?xml version="1.0" encoding="UTF-8" ?><schema id="IncomeCertificate"
  version="1.0"targetNamespace="http://eservices.cdacmumbai.in/IncomeCertificate" elementFormDefault="qualified"
  xmlns="http://www.w3.org/2001/XMLSchema">
<element name="Form">
<complexType>
<sequence>
<element name="Content" maxOccurs="1" minOccurs="1">
<complexType>
<sequence>
<element name="RegistrationID">
<simpleType>
<restriction base="string">
<length value="11" />
<pattern value="[0-9]*" />
</restriction>
</simpleType>
</element>
<element name="NumberOfFamilyMembers">
<simpleType>
<restriction base="string">
<minLength value="1" />
<maxLength value="50" />
</restriction>
</simpleType>
</element>
<element name="MonthlyIncome">
<simpleType>
<restriction base="string">
<minLength value="1" />
<maxLength value="50" />
</restriction>
</simpleType>
</element>
```

```
<element name="SourceOfIncome">
  <simpleType>
    <restriction base="string">
      <minLength value="1" />
      <maxLength value="300" />
    </restriction>
  </simpleType>
</element>
<element name="PurposeForApplying">
  <simpleType>
    <restriction base="string">
      <minLength value="1" />
      <maxLength value="300" />
    </restriction>
  </simpleType>
</element>
</sequence>
</complexType>
</element>
</sequence>
<attribute name="FormName" type="string" />
<attribute name="ParentFormName" type="string" />
</complexType></element></schema>
```

Snippet 3 : Schema for Income Certificate Request

Schema for Income Certificate Response

XML Generated using this schema will be set in the Body Element of Submit Response document generated by SP generic connector.

```
<?xml version="1.0" encoding="UTF-8" ?> <schema
xmlns=http://www.w3.org/2001/XMLSchema
targetNamespace=http://www.cdacmumbai.in/RegistrationResponseSchema
xmlns:tns=http://www.cdacmumbai.in/RegistrationResponseSchema
elementFormDefault="qualified" version="1.0" attributeFormDefault="unqualified"
id="RegistrationResponseSchema">
<element name="RegistrationResponse">
<complexType><sequence><element name="applicationId" type="string" />
<element name="applicantName" type="string" />
<element name="responseStatus" type="string" />
<element name="responseDetails" type="string" />
</sequence>
</complexType>
</element>
</schema>
```

Snippet 4 : Schema for Income Certificate Response

Schema for Income CertificateAsynchronous Request

XML Generated using this schema will be set in the Body Element of Submit Request document generated by SAP generic connector

```
<?xml version="1.0" encoding="UTF-8" ?> <schema
xmlns=http://www.w3.org/2001/XMLSchema
targetNamespace=http://eservices.cdacmumbai.in/AsynSubmitReq
xmlns:tns=http://eservices.cdacmumbai.in/AsynSubmitReq version="1.0"
id="AsynSubmitReq" elementFormDefault="qualified">
<element name="Form">
<complexType>
<sequence><element name="Content" maxOccurs="1" minOccurs="1">
<complexType>
<sequence>
<element name="ServiceID">
<simpleType>
<restriction base="string">
```

```
<pattern value="[0-9]{11}" /></restriction>
```

```
</simpleType>  
</element>  
</sequence>  
</complexType>  
</element>  
</sequence>  
</complexType>  
</element>  
</schema>
```

Snippet 5 : AynchronousSubmitRequest.xsd

6.4 Executing Synchronous Submit Request at SAP end

Recipe 1 Synchronous Submit Request Execution

The citizen will fill up registration form to avail of all e-services once he receives the unique Registration Id. eServices should give one unique Registration Id to the citizen as soon as he fills up Registration form. This requirement can be fulfilled by sending a Synchronous Submit Request to the SP, in which SP responds immediately with Synchronous Submit Response.

Solution:

The following steps are involved in executing the synchronous submit request. Each of these step are explained in detail along with the code snippets.

1. The communication between SAP-Gateway & Gateway-SP follows IIP-IIS protocol standards so all the requests to & from the gateway are in XML format. The data from SAP to SP is set in <Body> element of the submit request and is in XML format. So the data must be first converted to XmlObject ().
2. There must be some understanding between SAP and SP regarding the structure of data sent from SAP to SP. So a schema of the same is recommended.
3. Follow the guidelines under [Chapter 4.1](#) to build a web application.
4. Create a schema as per code given in *Snippet 1*.
5. Build a jar (Java classes) from the above schema (using XMLBEANS or any other tool) which could be used for converting the simple text data to XmlObject ().
6. Call the setter methods of class SubmitRequest and call makeSynchronousSubmitRequest () in application connector.

Flow of Synchronous Submit Request from SAP to SP

Registration.html → RegistrationServlet.java → BusinessClass.java → SAPGeneric Connector → SSDG → SP Generic → SP Application Specific

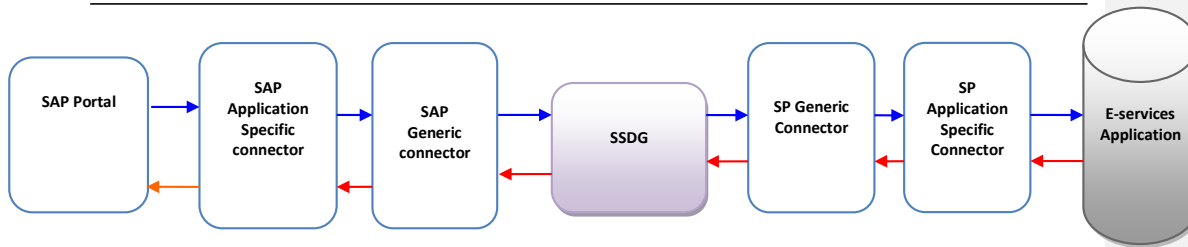


Figure 17 : SAP calling e-services application in synchronous mode

```
<html>
<head>
<title>Registration Form</title>
</head>
<body>
<form name="frmMinPadivam" method="POST" action="RegistrationServlet">
</form>
</body>
</html>
```

Snippet 6 : HTML code of Registration Form for one time registration

Discussion:

Registration Form collects the data entered by Citizen and forwards it to RegistrationServlet class.

```
package in.cdacmumbai.eservices.client;
import in.cdacmumbai.eservices.registrationForm.FormDocument;
```

```
import in.cdacmumbai.eservices.registrationForm.FormDocument.Form;
import in.cdacmumbai.eservices.registrationForm.FormDocument.Form.Content;
import
in.cdacmumbai.registrationResponseSchema.RegistrationResponseDocument.RegistrationRes
ponse;

import java.io.IOException;
import java.io.PrintWriter;
import connector.messages.*;
import connector.messages.parameters.*;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.xmlbeans.XmlException;
import org.apache.xmlbeans.XmlObject;
import org.w3c.dom.Node;

public class RegistrationServlet extends HttpServlet {

    @Override
    public void init(ServletConfig config) throws ServletException {
        super.init(config);
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {

        //Get Data From HttpServletRequest.

        FormDocument formDoc=FormDocument.Factory.newInstance();
        Form form=formDoc.addNewForm();
        Content content= form.addNewContent();
```

```
System.out.println("REQUESTED PARAM"+(String)request.getParameter("Name"));
```

```

content.setName((String)request.getParameter("Name"));
content.setShortName((String)request.getParameter("ShortName"));
content.setGuardianType((String)request.getParameter("GuardianType"));
content.setGuardianName((String)request.getParameter("GuardianName"));
content.setDateOfBirth((String)request.getParameter("DateOfBirth"));
content.setGender((String)request.getParameter("Gender"));
content.setNationality((String)request.getParameter("Nationality"));
content.setPermanentCountry((String)request.getParameter("PermanentCountry"));
content.setPermanentState((String)request.getParameter("PermanentState"));
content.setPermanentDistrict((String)request.getParameter("PermanentDistrict"));
content.setPermanentSubDistrict((String)request.getParameter("PermanentSubDistrict"));
content.setPermanentCity((String)request.getParameter("PermanentCity"));
content.setPermanentPinCode((String)request.getParameter("PermanentPinCode"));

boolean flag=true;
    SubmitResponse submitResponse=new SubmitResponse();
try
{
//Convert this Data into XMLObject.
    XmlObject xmlObj=XmlObject.Factory.parse(formDoc.toString());
//Call SAP Connector
    BusinessClass bb= new BusinessClass();
//Call Synchronous Service of Application Specific Connector.
    submitResponse =bb.makeSyncCall(xmlObj);
    flag=false;
}
catch(XmlException e)
{
    flag=true; //raise error condition.
}

if(!flag) //if NO error
{
if(submitResponse.getStatus().equals(RequestStatus.SUCCESS)
{
    XmlObject body=(XmlObject) submitResponse.getResponse() ;
    Node node=body.getDomNode();
    Node pNode=node.getChildNodes().item(0);
//System.out.println("PNODE"+pNode.getNodeName());
    Node appId=pNode.getChildNodes().item(0).getNextSibling();
    System.out.println("appId"+appId.getNodeName());
    Node n=appId.getChildNodes().item(0);
    String registration_id=n.getNodeValue();

```

```

        RequestDispatcher rd=request.getRequestDispatcher("/success.jsp");
        request.setAttribute("FieldName","RegistrationID");
        request.setAttribute("RegistrationID",registration_id);
        rd.forward(request,response);
    }
    else
    {
        RequestDispatcher rd=request.getRequestDispatcher("/error.jsp");
        request.setAttribute("error","There is some problem in processing yuor request.
<br/> Please Try again later.");
        rd.forward(request,response);
    }
}
}

}
public void destroy() {
}
}
}

```

Snippet 7 : Code for RegistrationServlet.java

Discussion:

- i) FormDocument class is class generated from RegistrationForm Schema and is used to convert data received from Registration form to XmlObject.
- ii) Call ***makeSyncCall*** () method and pass this XmlObject as parameter as highlighted in blue in *Snippet 7*.

```

package in.cdacmumbai.eservices.client;
import org.apache.xmlbeans.XmlException;
import org.apache.xmlbeans.XmlObject;
import connector.sap.SAPConnector;
import connector.sap.SAPConnectorImpl;
import connector.messages.*;
import connector.messages.parameters.*;
import java.util.Calendar;
import java.util.Date;
import java.util.Enumeration;
import java.util.Properties;
import java.util.ResourceBundle;

```

```

public class BusinessClass {

    public SubmitResponse makeSyncCall(XmlObject xmlObject){
        SubmitResponse submitResponse=new SubmitResponse();
        String targetEndPoint =
"http://nsdgstaging.cdacmumbai.in/gateway/services/NSDGService";

        String classId="http://202.141.151.140/77";//Sync1
        SubmitRequest submitRequest=new SubmitRequest();
        submitRequest.setTargetEndPointURL(targetEndPoint);
        submitRequest.setTransactionId(transactionId);
        submitRequest.setClassId(classId);
        submitRequest.setResponseMode(ResponseMode.SYNCHRONOUS);
        submitRequest.setNsdgTest(NSDGTTest.NO);
        submitRequest.setSpAuthentication(SPAAuthentication.NO);
        submitRequest.setAuthenticationType(AuthenticationType.CLEAR);
        submitRequest.setMessageBody(xmlObject);
        SAPConnector connector = new SAPConnectorImpl();
        System.out.println("Making request for class : " + classId);
        try{
            submitResponse = connector.makeSynchronousSubmitRequest(submitRequest);
        }catch(Exception e){
            e.printStackTrace();
            System.out.println("Exception occurred!");
        }

        if(submitResponse.getStatus().equals(RequestStatus.RESUBMIT))
        {
            System.out.println("Submit Request is not valid.");
            System.out.println("Please RESUBMIT the request. Reason :
"+submitResponse.getReason());
        }
        else if(submitResponse.getStatus().equals(RequestStatus.FAILED))
        {
            System.out.println("Request FAILED.");
            String error="";
            String [] errors=submitResponse.getErrors();
            for(int i=0;i<errors.length;i++)
            {
                System.out.println(errors[i]);
                error=error+"\n"+errors[i];
            }
        }
    }
}

```

```
        else if(submitResponse.getStatus().equals(RequestStatus.SUCCESS))
        {
            System.out.println("Submitted Request Successfully.");
            System.out.println("Response
Document"+submitResponse.getResponse().toString());
        }
        return submitResponse;
    }
```

Snippet 8 : Code for Synchronous Submit Request

Discussion:

- i) Call setter methods of `SubmitRequest` for paasing values for parameters like `targetEndPoint`, `CLASSID`, `RESPONSEMODE`, `BODY` etc. as highlighted in blue in *Snippet 8*.
- ii) Create an instance of `SAPConnector` class (Generic Connector class) and call method `makeSynchronousSubmitRequest` (`SubmitRequest`) as highlighted in pink in *Snippet 8*.
- iii) `makeSynchronousSubmitRequest` returns `Response` and we check the status in the returned response it can be `RESUBMIT`, `FAILED` or `SUCCESS`.

6.5 Processing of eServices Request at SP end sent in Synchronous Mode

The request sent by SAP is retrieved by SP generic connector from SSDG and is sent to SP application specific connector by invoking the method `callSyncSubmitRequest ()` of SP application specific connector. This method will process the data sent by SAP.

The steps given below discuss the request processing at SP end. Deployment steps for SP generic connector and SP application specific connector is already discussed in [Section 3.1](#) and [Section 5.1](#) .

Recipe 2 Retrieve Request from SSDG in case of Synchronous Submit Request.

eService Application needs to retrieve the request sent by SAP as described in *Recipe 1*.

Solution:

- i) SP Generic connector receives the registration form through SSDG in IIP/IIS format.
- ii) The SP Generic Connector extracts the body part generated by SAP and identifies if it is a synchronous call or asynchronous call.
- iii) Accordingly the method *callSyncSubmitRequest()* of SP application connector will be called for synchronous Submit Request.

```
package com.cdac.sp;
import in.cdacmumbai.eservices.client.Database;
import in.cdacmumbai.eservices.registrationForm.FormDocument;
import in.cdacmumbai.eservices.registrationForm.FormDocument.Form;
import in.cdacmumbai.registrationResponseSchema.RegistrationResponseDocument;
import
in.cdacmumbai.registrationResponseSchema.RegistrationResponseDocument.RegistrationRes
ponse;
import in.gov.mit.eGov.indiaOne.schema.errorresponse.ErrorResponseDocument;
import
in.gov.mit.eGov.indiaOne.schema.errorresponse.ErrorResponseDocument.ErrorResponse;
import
in.gov.mit.eGov.indiaOne.schema.errorresponse.ErrorResponseDocument.ErrorResponse.Err
or;
import javax.ejb.CreateException;
import javax.ejb.EJBException;
import javax.ejb.SessionBean;
import javax.ejb.SessionContext;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.xml.crypto.dsig.XMLObject;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import org.apache.xmlbeans.XmlException;
import org.apache.xmlbeans.XmlObject;
import org.w3c.dom.DOMImplementation;
import org.w3c.dom.Document;
import org.w3c.dom.DocumentType;
import org.w3c.dom.Node;
```

```
import org.w3c.dom.Text;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import sun.misc.BASE64Decoder;
import com.sun.org.apache.bcel.internal.generic.GETSTATIC;
//import com.cdac.ReportTest;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.io.StringReader;
import java.math.BigInteger;
import java.rmi.RemoteException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import java.text.DateFormat;
import java.text.ParseException;

import java.text.SimpleDateFormat;
import java.util.*;

public class SPConnectorClientBean implements SessionBean {
    private static final long serialVersionUID = 1L;

    private SessionContext ctx = null;

    //data base conn

    public Object callSyncSubmitRequest(Object body, String classid,
        String transactionid, String correlationID) throws XmlException, RemoteException,
        IOException
    {
        try{

System.out.println("In callSyncSubmitRequest of SP Application Specific");
            System.out.println("Body ::::: "+body.toString());
            XmlObject respObj;
            XmlObject xmlobj = XmlObject.Factory.parse( body.toString());

            if(classid.equals("http://192.168.0.140/150"))
```

```

{
//If Registration Service.

FormDocument formDoc=FormDocument.Factory.parse(xmlobj.toString());
Form rForm=formDoc.getForm();

String personname="" + rForm.getContent().getName();
String shortname = "" + rForm.getContent().getShortName();
String guardiantype = "" + rForm.getContent().getGuardianType();
String guardianname="" + rForm.getContent().getGuardianName() ;
String dateofbirth = "" + rForm.getContent().getDateOfBirth();
String gender = "" + rForm.getContent().getGender();
String nationality ="" + rForm.getContent().getNationality();
String permanentstate="" + rForm.getContent().getPermanentState();
String permanentdistrict="" + rForm.getContent().getPermanentDistrict();
String permanentsubdistrict="" + rForm.getContent().getPermanentSubDistrict();
String permanentcity="" + rForm.getContent().getPermanentCity();
String permanentPinCode="" + rForm.getContent().getPermanentPinCode();

SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
Date dateOfBirth=sdf.parse(dateofbirth);

String query = "INSERT INTO commonform(applicantid, \"name\", \" +
"shortname, gurdiantype, gurdianname,\" +
" dateofbirth,gender,\" +
" nationality, \" +
"permanentstate, permanentdistrict, permanentsubdistrict, permanentcity, \" +
" permanentpincode)" +
" VALUES (nextval('uid_seq'),"+ personname+ ",\""+ shortname+ "\",\""+
guardiantype + "\",\""+ guardianname + "\",\""+
+dateOfBirth+ "\",\""+ gender+ "\",\""+ nationality+ "\",\""+ permanentstate+ "\",\""+
permanentdistrict+ "\",\""+ permanentsubdistrict+ "\",\""+
+ permanentcity+ "\",\""+ permanentPinCode+ ")";

Connection con=Database.getConnection();
PreparedStatement pstmt = con.prepareStatement(query);
Statement stmt=con.createStatement();
int i=stmt.executeUpdate(query);
stmt.close();
pstmt.close();
con.close();
ResultSet rs= null;
long id=0;

```

```

        if (i==1)
        {
            //Success Condition.
            //Extract Registration Id.
            String queryId="select curval('uid_seq');";
            rs=stmt.executeQuery(queryId);
            if(rs.next())
        {
            id=rs.getLong(1);
            }
            System.out.println("Registration ID"+id);
        }

        //Prepare Response XML.
        RegistrationResponseDocument
        rd=RegistrationResponseDocument.Factory.newInstance();
        RegistrationResponse resp= rd.addNewRegistrationResponse();
        resp.setApplicationId(""+id);
        resp.setApplicantName("" + personname);
        resp.setResponseStatus("SUCCESS");
        resp.setResponseDetails("SUCCESS");

        respObj=XmlObject.Factory.parse(rd.toString());
    }
    else
    //if(classid.equals("http://192.168.0.140/151"))
    {
        // Insert into Income cetificate table.
        in.cdacmumbai.eservices.incomeCertificate.FormDocument
        form=in.cdacmumbai.eservices.incomeCertificate.FormDocument.Factory.parse(xmlobj.toString());
        String registrationId=form.getForm().getContent().getRegistrationID();
        String
        noOfFamilyMembers=form.getForm().getContent().getNumberOfFamilyMembers();
        String mIncome=form.getForm().getContent().getMonthlyIncome();
        String sourceOfIncome=form.getForm().getContent().getSourceOfIncome();
        String purpose=form.getForm().getContent().getPurposeForApplying();

        String query="Insert into incomecert
        values('"+registrationId+"','"+noOfFamilyMembers+"','"+mIncome+"','"+sourceOfIncome
        +"'','"+purpose+"')";
        Connection con=Database.getConnection();
        PreparedStatement pstmt = con.prepareStatement(query);
        Statement stmt=con.createStatement();

```

```
int i=stmt.executeUpdate(query);
stmt.close();
pstmt.close();
con.close();
ResultSet rs= null;
long id=0;
if (i==1)
{
    //Success Condition.
    //Extract Application Id.
    String queryId="select currval('uid_seq')";

    rs=stmt.executeQuery(queryId);
    if(rs.next())
    {
        id=rs.getLong(1);
    }
    System.out.println("Application ID"+id);
}

//Create Response.

RegistrationResponseDocument
rd=RegistrationResponseDocument.Factory.newInstance();
RegistrationResponse resp= rd.addNewRegistrationResponse();
resp.setApplicationId(""+id);
resp.setResponseStatus("SUCCESS");
resp.setResponseDetails("SUCCESS");

    respObj=XmlObject.Factory.parse(rd.toString());
}

// Generate Response Document

SPResponse response=new SPResponse();
//If no error set Response Type as RESPONSE.
response.setResponseType(ResponseType.RESPONSE);
response.setResponse(resObj);

}catch (Exception E) {
E.printStackTrace();
//Else if error Prepare Error object.
```

```
        response.setResponseType(ResponseType.ERROR);
        response.setResponse(obj);
    }

    return response;
}

public Object getErrorObject()
{
    ErrorResponseDocument errordoc = null;
    errordoc = ErrorResponseDocument.Factory
        .newInstance();
    ErrorResponse newerrorresponse = errordoc
        .addNewErrorResponse();
    Error error = newerrorresponse.addNewError();
    error.setRaisedBy("SP Application");
    error.setNumber(new BigInteger("7000"));
    error.setType("Application Error");
    String[] str = new String[1];
    str[0] = "There was problem processing your request, Please retry after some time.";
    error.setTextArray(str);
    error.setLocation("SP");
    XmlObject returnerror = errordoc;
    Object obj=returnerror;
    System.out.println("in error obj"+ returnerror.xmlText());
    return obj;
}

public void setSessionContext(SessionContext ctx) throws EJBException,
    RemoteException {
    System.out.println("setSessionContext() method called");
    this.ctx = ctx;
}

public void ejbCreate() throws RemoteException,CreateException {
    System.out.println("ejbCreate()");
}

public void ejbRemove() throws EJBException, RemoteException {
    System.out.println("ejbRemove()");
}

}

public void ejbActivate() throws EJBException, RemoteException {
```

```
System.out.println("ejbActivate()");
}

public void ejbPassivate() throws EJBException, RemoteException {
    System.out.println("ejbPassivate()");
}
}
```

Snippet 9 : Code for SPApplicationConnector for Synchronous Submit Request

Discussion:

- i) Method *callSyncSubmitRequest* () is called by SP Generic Connector.
- ii) Code in red checks for Registration Form Request.
- iii) The Xml data sent from SAP in Body element of submit request ids then converted into text format using class FormDocument which is generated using RegistrationForm Schema shown in *Snippet 1*.
- iv) Code in blue inserts data into database and retrieves a Registration Id to be sent back to the Citizen.
- v) This response data is converted to XmlObject () using class RegistrationResponseDocument generated using RegistrationResponseSchema given in *Snippet 2*.
- vi) These values are then set in return SPResponse object and given back to the SSDG through SP Generic Connector.

6.6 Retrieving the Result sent by SP Application to SAP

SAP will retrieve the values received from eServices SP application as per RegistrationResponseSchema in *Snippet 2* against the synchronous request sent by SAP.

Recipe 3 Retrieve Result from SSDG in case of Synchronous Submit Request.

```
package in.cdacmumbai.eservices.client;
import org.apache.xmlbeans.XmlException;
import org.apache.xmlbeans.XmlObject;
import connector.sap.SAPConnector;
import connector.sap.SAPConnectorImpl;
import connector.messages.*;
import connector.messages.parameters.*;
import java.util.Calendar;
import java.util.Date;
import java.util.Enumeration;
import java.util.Properties;
import java.util.ResourceBundle;

public class BusinessClass {

    public SubmitResponse makeSyncCall(XmlObject xmlObject){
        SubmitResponse submitResponse=new SubmitResponse();
        String targetEndPoint =
"http://nsdgstaging.cdacmumbai.in/gateway/services/NSDGService";

        String classId="http://202.141.151.140/77";//Sync1
        SubmitRequest submitRequest=new SubmitRequest();
        submitRequest.setTargetEndPointURL(targetEndPoint);
        submitRequest.setTransactionId(transactionId);
        submitRequest.setClassId(classId);
        submitRequest.setResponseMode(ResponseMode.SYNCHRONOUS);
        submitRequest.setNsdgTest(NSDGTTest.NO);
        submitRequest.setSpAuthentication(SPAuthentication.NO);
        submitRequest.setAuthenticationType(AuthenticationType.CLEAR);
        submitRequest.setMessageBody(xmlObject);
        SAPConnector connector = new SAPConnectorImpl();
        System.out.println("Making request for class : " + classId);
        try{
            submitResponse = connector.makeSynchronousSubmitRequest(submitRequest);
```

```
    }catch(Exception e){
        e.printStackTrace();
        System.out.println("Exception occured!");
    }

    if(submitResponse.getStatus().equals(RequestStatus.RESUBMIT))
    {
        System.out.println("Submit Request is not valid.");
        System.out.println("Please RESUBMIT the request. Reason :
"+submitResponse.getReason());
    }
    else if(submitResponse.getStatus().equals(RequestStatus.FAILED))
    {
        System.out.println("Request FAILED.");
        String error="";
        String [] errors=submitResponse.getErrors();
        for(int i=0;i<errors.length;i++)
        {
            System.out.println(errors[i]);
            error=error+"\n"+errors[i];
        }
    }
    else if(submitResponse.getStatus().equals(RequestStatus.SUCCESS))
    {
        System.out.println("Submitted Request Successfully.");
        System.out.println("Response
Document"+submitResponse.getResponse().toString());
    }
    return submitResponse;
}
}
```

Snippet 10 : Code for retrieving response for Synchronous Submit Request

Discussion:

- i) ***makeSynchronousSubmitRequest*** () method of Generic connector will return SubmitResponse which contains the 'status'(using getStatus() of SubmitResponse class) of the response i.e. either SUCCESS/FAILED and also the Response Data sent

by the SP as highlighted in blue in *Snippet 10*. This SubmitResponse will be returned to RegistrationServlet class as shown in *Snippet*

```
package in.cdacmumbai.eservices.client;
import in.cdacmumbai.eservices.registrationForm.FormDocument;
import in.cdacmumbai.eservices.registrationForm.FormDocument.Form;
import in.cdacmumbai.eservices.registrationForm.FormDocument.Form.Content;
import
in.cdacmumbai.registrationResponseSchema.RegistrationResponseDocument.RegistrationRes
ponse;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.xmlbeans.XmlException;
import org.apache.xmlbeans.XmlObject;
import org.w3c.dom.Node;
import connector.messages.*;
import connector.messages.parameters.*;

public class RegistrationServlet extends HttpServlet {

    @Override
    public void init(ServletConfig config) throws ServletException {
        super.init(config);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {

        //Get Data From HttpServletRequest.

        FormDocument formDoc=FormDocument.Factory.newInstance();
        Form form=formDoc.addNewForm();
        Content content= form.addNewContent();
        System.out.println("REQUESTED PARAM"+(String)request.getParameter("Name"));

        content.setName((String)request.getParameter("Name"));
        content.setShortName((String)request.getParameter("ShortName"));
        content.setGuardianType((String)request.getParameter("GuardianType"));
        content.setGuardianName((String)request.getParameter("GuardianName"));
    }
}
```

```

content.setDateOfBirth((String)request.getParameter("DateOfBirth"));
content.setGender((String)request.getParameter("Gender"));
content.setNationality((String)request.getParameter("Nationality"));
content.setPermanentCountry((String)request.getParameter("PermanentCountry"));
content.setPermanentState((String)request.getParameter("PermanentState"));
content.setPermanentDistrict((String)request.getParameter("PermanentDistrict"));
content.setPermanentSubDistrict((String)request.getParameter("PermanentSubDistrict"));
content.setPermanentCity((String)request.getParameter("PermanentCity"));
content.setPermanentPinCode((String)request.getParameter("PermanentPinCode"));

boolean flag=true;
    SubmitResponse submitResponse=new SubmitResponse();
try
{
//Convert this Data into XMLObject.
    XmlObject xmlObj=XmlObject.Factory.parse(formDoc.toString());
//Call SAP Connector
    BusinessClass bb= new BusinessClass();
//Call Synchronous Service of Connector.
    submitResponse =bb.makeSyncCall(xmlObj);
    flag=false;
}
catch(XmlException e)
{
    flag=true; //raise error condition.
}
catch(Exception e)
{
    flag=true; //raise error condition.
}

if(!flag) //if NO error
{
if(submitResponse.getStatus().equals(RequestStatus.SUCCESS))    {

        XmlObject body=(XmlObject) submitResponse.getResponse() ;
        Node node=body.getDomNode();
        Node pNode=node.getChildNodes().item(0);
        //System.out.println("PNODE"+pNode.getNodeName());
        Node appId=pNode.getChildNodes().item(0).getNextSibling();
        System.out.println("appId"+appId.getNodeName());
        Node n=appId.getChildNodes().item(0);
        String registration_id=n.getNodeValue();
    }
}

```

```
RequestDispatcher rd=request.getRequestDispatcher("/success.jsp");
request.setAttribute("FieldName","RegistrationID");
request.setAttribute("RegistrationID",registration_id);
rd.forward(request,response);
}
else
{
RequestDispatcher rd=request.getRequestDispatcher("/error.jsp");
request.setAttribute("error","There is some problem in processing yuor request.
<br/> Please Try again later.");
rd.forward(request,response);
}
}
}
public void destroy() { }
}
```

Snippet 11 : Code for RegistrationServlet.java

Discussion:

- i) Code highlighted in pink in *Snippet 11* retrieves XmlObject from submitResponse.getResponse() .
- ii) Next Line uses RegistrationResponseDocument class which is generated from RegistrationResponseSchema in *Snippet 2* and is used here for retrieving the data from XmlObject.
- iii) The registrationId is then displayed to the Citizen.

6.7 Executing Asynchronous Submit Request

In this section we will discuss the Income Tax Certification issue request and response are discussed. The mode of this communication is asynchronous.

Recipe 4 Asynchronous Submit Request Execution

SAP raises a request for generation of Income Certificate by filling in necessary details along with the unique Registration Id. The SP is registered with SSDG in asynchronous mode of communication so it provides the response in the form of Income Certificate to SSDG after processing at some later point of time.

Solution:

- i) The communication between SAP-Gateway & Gateway-SP follows IIP-IIS protocol standards so all the requests to & from the gateway are in XML format. The data from SAP to SP is set in <Body> element of the submit request and is in XML format. So the data must be first converted to XmlObject ().
- ii) There must be some understanding between SAP and SP regarding the structure of data sent from SAP to SP. So a schema of the same is recommended.
- iii) Create a schema as per code *Snippet 3*.
- iv) Build a jar (Java classes) from the above schema (using XMLBEANS or any other tool) which could be used for converting the simple text data to XmlObject ().
- v) Call `makeAsynchronousSubmitRequest()` of `SubmitRequest` by setting values using setter methods in application specific connector.

Flow of Asynchronous Submit Request from SAP to SP:

IncomeCertificate.html → IncomeCertApplication.java → BusinessClass.java → Generic Connector → SSDG → SP Generic Connector → SP Application specific Connector → SP Application.

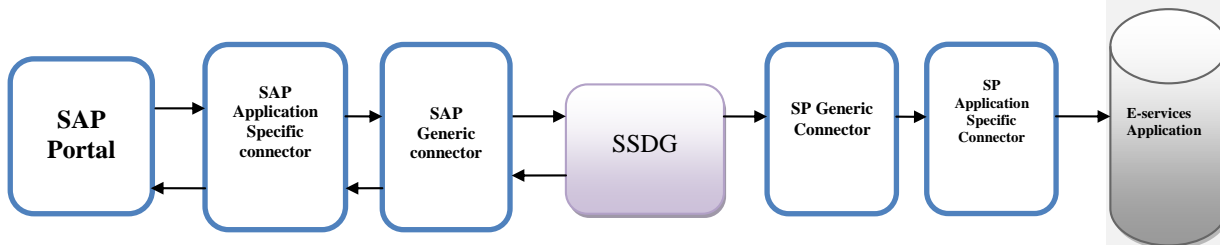


Figure 18 : SAP asynchronously submits request to e-services application through SSDG.

```
<html>
<head>
<title>Income Certificate Application Form</title>
</head>
<body>
<form name="frmMinPativam" method="POST" action="IncomeCert">
</form>
</body>
</html>
```

Snippet 12 : Income Certificate .html

Discussion:

- i) This form collects the data entered by Citizen and forwards it to IncomeCertApplication class

```
package in.cdacmumbai.eservices.client;

import in.cdacmumbai.eservices.incomeCertificate.FormDocument;
import in.cdacmumbai.eservices.incomeCertificate.FormDocument.Form;
import in.cdacmumbai.eservices.incomeCertificate.FormDocument.Form.Content;
import java.io.IOException;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.Statement;
import connector.messages.*;
import connector.messages.parameters.*;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.xmlbeans.XmlException;
import org.apache.xmlbeans.XmlObject;
import org.w3c.dom.Node;

public class IncomeCertApplication extends HttpServlet {
```

```

@Override
public void init(ServletConfig config) throws ServletException {
    super.init(config);
}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

    //Get Data From HttpServletRequest.

    FormDocument formDoc=FormDocument.Factory.newInstance();
    Form form=formDoc.addNewForm();
    Content content= form.addNewContent();

    content.setRegistrationID(request.getParameter("RegistrationID"));
    content.setNumberOfFamilyMembers(request.getParameter("NumberOfFamilyMembers")
);
    content.setMonthlyIncome((String)request.getParameter("MonthlyIncome"));
    content.setSourceOfIncome(request.getParameter("SourceOfIncome"));
    content.setPurposeForApplying(request.getParameter("PurposeForApplying"));
    boolean flag=true;

    SubmitResponse submitResponse=new SubmitResponse();
    BusinessClass bb;
    try
    {
        //Convert this Data into XMLObject.
        XmlObject xmlObj=XmlObject.Factory.parse(formDoc.toString());
        //Call SAP Connector
        bb= new BusinessClass();
        //Call Synchronous Service of Connector.
        submitResponse =bb.makeSyncCall(xmlObj);
        flag=false;

    if(submitResponse.getStatus().equals(RequestStatus.SUCCESS))
    {
        XmlObject body=(XmlObject) submitResponse.getResponse();
        Node node=body.getDomNode();
        Node pNode=node.getChildNodes().item(0);
        //System.out.println("PNODE"+pNode.getNodeName());
        Node appId=pNode.getChildNodes().item(0).getNextSibling();
        System.out.println("appId"+appId.getNodeName());
        Node n=appId.getChildNodes().item(0);
        String fileNo=n.getNodeValue();
    }
}

```

```

//Apply For Income Certificate Against this File Number.
in.cdacmumbai.eservices.asynSubmitReq.FormDocument
asyn_doc=in.cdacmumbai.eservices.asynSubmitReq.FormDocument.Factory.newInstance();
in.cdacmumbai.eservices.asynSubmitReq.FormDocument.Form
frm=asyn_doc.addNewForm();
in.cdacmumbai.eservices.asynSubmitReq.FormDocument.Form.Content
cont=frm.addNewContent();
cont.setServiceID(fileNo);

//Prepare XmlObject for Async Request.
System.out.println("FORMDOC"+asyn_doc.toString());
XmlObject body1=XmlObject.Factory.parse(asyn_doc.toString());
System.out.println("FormDoc "+asyn_doc.toString());
System.out.println("make Async service call");
bb= new BusinessClass();
SubmitResponse submitAsyncResponse=new SubmitResponse();
submitAsyncResponse =bb.makeAsyncCall(body1);

if(submitAsyncResponse.getStatus().equals(RequestStatus.SUCCESS))
{
//Successful Acknowledement of Asynchronous Request.
//Retrive Correlation Id.
String corrId= submitAsyncResponse .getCorrelationId ();

//Keep this Correlation Id against FileNumber for polling purpose.
String sqlquery="INSERT INTO sappoll_map (file_no, correlation_id) VALUES
("+fileNo.trim()+",""+corrId.trim()+")";

Connection con=Database.getConnection();
PreparedStatement pstmt = con.prepareStatement(sqlquery);
Statement stmt=con.createStatement();
int i=stmt.executeUpdate(sqlquery);
stmt.close();
pstmt.close();
con.close();

//Send the file Number to Citizen.
RequestDispatcher rd=request.getRequestDispatcher("/success.jsp");
request.setAttribute("FieldName", "File Number");
request.setAttribute("File Number",fileNo);
rd.forward(request,response);
}
else

```

```

        if(submitAsyncResponse.getStatus().equals(RequestStatus.FAILED))    {
            //Error in Submitting Asynchronous Request.
            flag=true;
        }
    }
    else
    {
        flag=true;
    }

}
catch(XmlException e)
{
flag=true; //raise error condition.
}
catch(Exception e)
{
    flag=true; //raise error condition.
}
if(!flag) //if NO error
{

}
else
{
    RequestDispatcher rd=request.getRequestDispatcher("/error.jsp");
    request.setAttribute("error","There is some problem in processing yuor request. <br/>
Please Try again later.");
    rd.forward(request,response);
}
}

public void destroy() { }
}

```

*Snippet 13 : IncomeCertApplication***Discussion:**

- i) Here the class FormDocument collects data from HttpServletRequest and converts it into XmlObject format. This FormDocument class is generated from *Snippet 3*.
- ii) Make Synchronous Submit Request first to Get File Number from SP. Retrieve File Number using RegistrationResponseDocument schema given in *Snippet 4*.

Comment [a1]: Add point for response status(failed, resubmit or success)

- iii) Upon successful synchronous call, prepare XmlObject for **Asynchronous Method call** using XmlDocument class which is generated using schema in *Snippet 5*
- iv) Call **makeAsyncCall ()** method of SAP Application Specific Connector and pass the above mentioned XmlObject to it.
- v) makeAsyncCall returns submitResponse, after receiving response we are checking the response status.

```
package in.cdacmumbai.eservices.client;
import connector.messages.*;
import connector.messages.parameters.*;

import org.apache.xmlbeans.XmlException;
import org.apache.xmlbeans.XmlObject;

import connector.sap.SAPConnector;
import connector.sap.SAPConnectorImpl;

import java.util.Calendar;
import java.util.Date;
import java.util.Enumeration;
import java.util.Properties;
import java.util.ResourceBundle;
public class BusinessClass {

    public SubmitResponse makeAsyncCall(XmlObject xmlObject){
        System.out.println("BusinessClass::makeCall() method starts");
        String targetEndPoint="http://192.168.0.37:8080/gateway/services/NSDGService";
        String classId="http://192.168.0.140/130";
        SubmitRequest submitRequest=new SubmitRequest();
        submitRequest.setTargetEndPointURL(targetEndPoint);
        submitRequest.setTransactionId(transactionId);
        submitRequest.setClassId(classId);
        submitRequest.setResponseMode(ResponseMode.ASYNCHRONOUS);
        submitRequest.setNsdgTest(NSDGTTest.NO);
        submitRequest.setSpAuthentication(SPAAuthentication.NO);
        submitRequest.setAuthenticationType(AuthenticationType.CLEAR);

        xmlObject=XmlObject.Factory.parse(xmlBody);
        submitRequest.setMessageBody(xmlObject);
    }
}
```

```
System.out.println("Submit Request Parameters have been set!");

SAPConnector connector = new SAPConnectorImpl();
System.out.println("Making submit request for class : " + classId);

    submitResponse = connector.makeAsynchronousSubmitRequest(submitRequest);

    if(submitResponse.getStatus().equals(RequestStatus.RESUBMIT))
    {
        System.out.println("Submit Request is not valid.");
        System.out.println("Please RESUBMIT the request. Reason :
"+submitResponse.getReason());
    }
    else if(submitResponse.getStatus().equals(RequestStatus.FAILED))
    {
        System.out.println("Request FAILED.");
        String error="";
        String [] errors=submitResponse.getErrors();
        for(int i=0;i<errors.length;i++)
        {
            System.out.println(errors[i]);
            error=error+"\n"+errors[i];
        }
    }
    else if(submitResponse.getStatus().equals(RequestStatus.SUCCESS))
    {
        System.out.println("Submitted Request Successfully.");
        System.out.println("Response
Document"+submitResponse.getResponse().toString());
    }
return submitResponse;
}
```

Snippet 14 : ASynchronous Submit Request

Discussion:

- i) Code in red sets different parameters like targetEndPoint, CLASSID, RESPONSEMODE, BODY etc.using setter methods of class SubmitRequest.
- ii) Create an instance of SAPConnector class (Generic Connector class) and call method makeAsynchronousSubmitRequest (SubmitRequest) as shown in *Snippet 14*.

- iii) `makeAsynchronousSubmitRequest` returns response and the code in pink is checking the response status.

6.8 Retrieving Acknowledgement from SSDG

Recipe 5 Retrieve the Acknowledgement from SSDG

The asynchronous request sent by SAP is routed to e-services application through SSDG. Now, as the request is asynchronous, the response will not be returned instantaneously to SAP. Meanwhile when the request is submitted to SSDG, the SSDG validates the correctness of request message and if the request is found to be valid, SSDG issues submit acknowledgement to SAP. This recipe shows how this acknowledgement is retrieved by SAP which is submitted by SSDG.

Solution:

As the mode of communication is asynchronous the response from e-services application will be submitted to SSDG. SAP will retrieve the values received in the submitted acknowledgement and use the retrieved values to poll SSDG for retrieving the response submitted by SP to SSDG.

- i) `SubmitResponse` returned from SAP Generic connector contains either `ACKNOWLEDGEMENT` or `ERROR` values in it.
- ii) So check the 'status' returned in `RequestStatus`, if it is `SUCCESS` i.e. successful request has been made and acknowledgement has been returned then retrieve the `Correlation Id` from `HashMap` which will be useful for Polling purpose.

```
package in.cdacmumbai.eservices.client;

import in.cdacmumbai.eservices.incomeCertificate.FormDocument;
import in.cdacmumbai.eservices.incomeCertificate.FormDocument.Form;
import in.cdacmumbai.eservices.incomeCertificate.FormDocument.Form.Content;

import connector.messages.SubmitPoll;
import connector.messages.SubmitPollResponse;
import connector.messages.SubmitRequest;
import connector.messages.SubmitResponse;
import connector.messages.parameters.AuthenticationType;
import connector.messages.parameters.NSDGTest;
```

```
import connector.messages.parameters.RequestStatus;
import connector.messages.parameters.ResponseDocumentType;
import connector.messages.parameters.ResponseMode;
import connector.messages.parameters.SPAuthentication;
import connector.sap.SAPConnector;
import connector.sap.SAPConnectorImpl;
import java.io.IOException;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.Statement;
import connector.messages.*;
import connector.messages.parameters.*;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.xmlbeans.XmlException;
import org.apache.xmlbeans.XmlObject;
import org.w3c.dom.Node;

public class IncomeCertApplication extends HttpServlet {

    @Override
    public void init(ServletConfig config) throws ServletException {
        super.init(config);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {

        //Get Data From HttpServletRequest.

        FormDocument formDoc=FormDocument.Factory.newInstance();
        Form form=formDoc.addNewForm();
        Content content= form.addNewContent();

        content.setRegistrationID(request.getParameter("RegistrationID"));
        content.setNumberOfFamilyMembers(request.getParameter("NumberOfFamilyMembers")
);
        content.setMonthlyIncome((String)request.getParameter("MonthlyIncome"));
    }
}
```

```

content.setSourceOfIncome(request.getParameter("SourceOfIncome"));
content.setPurposeForApplying(request.getParameter("PurposeForApplying"));
boolean flag=true;
BusinessClass bb;
    SubmitResponse asyncsubmitResponse=new SubmitResponse();
try
{
    //Call SAP Connector
    //Prepare XmlObject for Async Request.
    System.out.println("FORMDOC"+asyn_doc.toString());
    XmlObject body1=XmlObject.Factory.parse(asyn_doc.toString());
    System.out.println("FormDoc "+asyn_doc.toString());
    System.out.println("make Async service call");
    bb= new BusinessClass();
    asyncsubmitResponse =bb.makeAsyncCall(body1);

    if(asyncsubmitResponse.getStatus().equals(RequestStatus.SUCCESS))
    {
        //Successful Acknowledement of Asynchronous Request.
        //Retrive Correlation Id.
        String corrId= asyncsubmitResponse.getCorrelationId();

        //Keep this Correlation Id against FileNumber for polling purpose.
        String sqlquery="INSERT INTO sappoll_map (file_no, correlation_id) VALUES
        ("'+fileNo.trim()+"','"+corrId.trim()+"");

        Connection con=Database.getConnection();
        PreparedStatement pstmt = con.prepareStatement(sqlquery);
        Statement stmt=con.createStatement();
        int i=stmt.executeUpdate(sqlquery);
        stmt.close();
        pstmt.close();
        con.close();

        //Send the file Number to Citizen.
        RequestDispatcher rd=request.getRequestDispatcher("/success.jsp");
        request.setAttribute("FieldName","File Number");
        request.setAttribute("File Number",fileNo);
        rd.forward(request,response);
    }
    else
    if(asyncsubmitResponse. getStatus().equals(RequestStatus.FAILED))
    {
        //Error in Submitting Asynchronous Request.

```

```
        flag=true;
    }

    }
    else
    {
        flag=true;
    }

    }

    catch(XmlException e)
    {
        flag=true; //raise error condition.
    }

    catch(Exception e)
    {
        flag=true; //raise error condition.
    }
    if(!flag) //if NO error
    {

    }
    else
    {
        RequestDispatcher rd=request.getRequestDispatcher("/error.jsp");
        request.setAttribute("error","There is some problem in processing yuor request. <br/>
Please Try again later.");
        rd.forward(request,response);
    }

}

public void destroy() { }
}
```

Snippet 15 : Retrieving Acknowledgement for Asynchronous Submit Request

Discussion:

- i) Code in blue shows that the *makeAsyncCall()* method returns SubmitResponse object *asynsubmitResponse*.
- ii) The subsequent line checks for Successful Acknowledgement of asynchronous request.

- iii) As the Correlation Id works as receipt number, it is necessary to retrieve it and store it as highlighted in pink.

6.9 Processing of E-services Request at SP end sent in Asynchronous Mode

In the above procedures request is send to SSDG through SAP Generic connector which will be forwarded to SP Generic connector through SSDG. When request from SAP is received by SSDG, SSDG will determine if the request is valid. If valid, SSDG sends Acknowledgement to SAP. The request is then retrieved by SP generic connector from SSDG and is sent to SP application specific connector by invoking the web method *callAsyncSubmitRequest ()* of SP application connector. This method will process the data sent by SAP.

Recipe 6 Retrieve Request from SSDG in case of Asynchronous Submit Request

eServices Application needs to retrieve the request sent by SAP as described in *Recipe 4*

Solution:

- i) SP Generic connector will be receiving the registration form through gateway in the IIP/IIS format.
- ii) The SP Generic Connector will extract the body part generated by SAP and will identify whether it is a synchronous call or asynchronous call.
- iii) Accordingly the method *callAsyncSubmitRequest()* of your Specific SP connector will be called for Asynchronous Submit Request.

```
package com.cdac.sp;

import in.cdacmumbai.eservices.client.Database;
import in.cdacmumbai.eservices.registrationForm.FormDocument;
import in.cdacmumbai.eservices.registrationForm.FormDocument.Form;
import in.cdacmumbai.registrationResponseSchema.RegistrationResponseDocument;
import
in.cdacmumbai.registrationResponseSchema.RegistrationResponseDocument.RegistrationRes
ponse;
import in.gov.mit.eGov.indiaOne.schema.errorresponse.ErrorResponseDocument;
```

```
import
in.gov.mit.eGov.indiaOne.schema.errorresponse.ErrorResponseDocument.ErrorResponse;
import
in.gov.mit.eGov.indiaOne.schema.errorresponse.ErrorResponseDocument.ErrorResponse.Err
or;
import javax.ejb.CreateException;
import javax.ejb.EJBException;
import javax.ejb.SessionBean;
import javax.ejb.SessionContext;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.xml.crypto.dsig.XMLObject;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

import org.apache.xmlbeans.XmlException;
import org.apache.xmlbeans.XmlObject;
import org.w3c.dom.DOMImplementation;
import org.w3c.dom.Document;
import org.w3c.dom.DocumentType;
import org.w3c.dom.Node;
import org.w3c.dom.Text;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;

import sun.misc.BASE64Decoder;

import com.sun.org.apache.bcel.internal.generic.GETSTATIC;

//import com.cdac.ReportTest;

import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
```

```
import java.io.StringReader;
```

```
import java.math.BigInteger;
import java.rmi.RemoteException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.*;
import connector.sp.util.ResponseType;
import connector.sp.util.SPResponse;
public class SPConnectorClientBean implements SessionBean {
    private static final long serialVersionUID = 1L;

    private SessionContext ctx = null;

    //data base conn

    public Object callSyncSubmitRequest(Object body, String transactionid,
        String classid, String correlationID) throws XmlException, RemoteException,
        IOException
    {
        try{
            System.out.println(" i am in callSyncSubmitRequest ");
            XmlObject respObj;
            XmlObject xmlobj = XmlObject.Factory.parse( body.toString());

            if(classid.equals("http://192.168.0.140/150"))
            {

                //If Registration Service.

                FormDocument formDoc=FormDocument.Factory.parse(xmlobj.toString());
                Form rForm=formDoc.getForm();

                String personname="" + rForm.getContent().getName();
                String shortname = "" + rForm.getContent().getShortName();
                String guardiantype = "" + rForm.getContent().getGuardianType();
                String guardiannname="" + rForm.getContent().getGuardianName() ;
                String dateofbirth = "" + rForm.getContent().getDateOfBirth();
                String gender = "" + rForm.getContent().getGender();
                String nationality ="" + rForm.getContent().getNationality();
```

```

String permanentstate="" + rForm.getContent().getPermanentState();
String permanentdistrict="" + rForm.getContent().getPermanentDistrict();
String permanentsubdistrict="" + rForm.getContent().getPermanentSubDistrict();
String permanentcity="" + rForm.getContent().getPermanentCity();
String permanentPinCode="" + rForm.getContent().getPermanentPinCode();

SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
Date dateOfBirth=sdf.parse(dateOfBirth);
String query = "INSERT INTO commonform(applicantid, \"name\", \" +
    \"shortname, gurdiantype, gudianname,\" +
    \" dateOfBirth,gender,\" +
    \" nationality, \" +
    \"permanentstate, permanentdistrict, permanentsubdistrict, permanentcity, \" +
    \" permanentpincode)\" +
    \" VALUES (nextval('uid_seq'),\"+ personname+ \"\", \" + shortname+ \"\", \" +
guardiantype + \"\", \" + gudianname + \"\", \" +
    +dateOfBirth+ \"\", \" + gender+ \"\", \" + nationality+ \"\", \" + permanentstate+ \"\", \" +
permanentdistrict+ \"\", \" + permanentsubdistrict+ \"\", \" +
    + permanentcity+ \"\", \" + permanentPinCode+ \"\")";

Connection con=Database.getConnection();
PreparedStatement pstmt = con.prepareStatement(query);
Statement stmt=con.createStatement();
int i=stmt.executeUpdate(query);
stmt.close();
pstmt.close();
con.close();
ResultSet rs= null;
long id=0;
if (i==1)
{
    //Success Condition.
    //Extract Registration Id.
    String queryId="select currval('uid_seq')";
    rs=stmt.executeQuery(queryId);
    if(rs.next())
    {
        id=rs.getLong(1);
    }
    System.out.println("Registration ID"+id);
}

//Prepare Response XML.

```

```

        RegistrationResponseDocument
rd=RegistrationResponseDocument.Factory.newInstance();
        RegistrationResponse resp= rd.addNewRegistrationResponse();
        resp.setApplicationId(""+id);
        resp.setApplicantName(""+ personname);
        resp.setResponseStatus("SUCCESS");
        resp.setResponseDetails("SUCCESS");

        respObj=XmlObject.Factory.parse(rd.toString());
    }
    else
        //if(classid.equals("http://192.168.0.140/151"))
    {
        // Insert into Income cetificate table.
        in.cdacmumbai.eservices.incomeCertificate.FormDocument
form=in.cdacmumbai.eservices.incomeCertificate.FormDocument.Factory.parse(xmlobj.toString());
        String registrationId=form.getForm().getContent().getRegistrationID();
        String
noOfFamilyMembers=form.getForm().getContent().getNumberOfFamilyMembers();
        String mIncome=form.getForm().getContent().getMonthlyIncome();
        String sourceOfIncome=form.getForm().getContent().getSourceOfIncome();
        String purpose=form.getForm().getContent().getPurposeForApplying();

        String query="Insert into incomecert
values(""+registrationId+"",""+noOfFamilyMembers+"",""+mIncome+"",""+sourceOfIncome+"','
"+purpose+"");

        Connection con=Database.getConnection();
        PreparedStatement pstmt = con.prepareStatement(query);
        Statement stmt=con.createStatement();
        int i=stmt.executeUpdate(query);
        stmt.close();
        pstmt.close();
        con.close();
        ResultSet rs= null;
        long id=0;
        if (i==1)
        {
            //Success Condition.
            //Extract Application Id.
            String queryId="select curval('uid_seq)";

            rs=stmt.executeQuery(queryId);

```

```
        if(rs.next())
        {
            id=rs.getLong(1);
        }
        System.out.println("Application ID"+id);
    }

    //Create Response.
    //
    RegistrationResponseDocument
rd=RegistrationResponseDocument.Factory.newInstance();
    RegistrationResponse resp= rd.addNewRegistrationResponse();
    resp.setApplicationId(""+id);
resp.setResponseStatus("SUCCESS");
    resp.setResponseDetails("SUCCESS");

    respObj=XmlObject.Factory.parse(rd.toString());
}

// Generate Response Document.

    SPResponse response=new SPResponse();
    //If no error set Response Type as RESPONSE.
    response.setResponseType(ResponseType.RESPONSE);
    response.setResponse(respObj);

}catch (Exception E) {
    E.printStackTrace();
//Else if error Prepare Error object.
    response.setResponseType(ResponseType.ERROR);
    response.setResponse(obj);
}

return response;

}

public Object callAsyncSubmitRequest(Object body, String transactionid,
    String classid, String correlationID) throws XmlException, RemoteException,
IOException
```

```

{
    System.out.println(" i am in callAsyncSubmitRequest ");
    XmlObject xmlObj=XmlObject.Factory.parse(body.toString());
    try{
        //Retrive Xml Data.
        in.cdacmumbai.eservices.asynSubmitReq.FormDocument
formDoc=in.cdacmumbai.eservices.asynSubmitReq.FormDocument.Factory.parse(xmlObj.to
String());
        String fileNo=formDoc.getForm().getContent().getServiceID();
        String query="update incomecert set
status='received',correlation_id='"+correlationID+"', class_id='"+classid+"'"+ " where
fileno='"+fileNo+"'";

        Connection con=Database.getConnection();
        PreparedStatement pstmt = con.prepareStatement(query);
        Statement stmt=con.createStatement();
        int i=stmt.executeUpdate(query);

        stmt.close();
        pstmt.close();
        con.close();
        SPResponse response=new SPResponse();
// If NO error set Response Type as ACKNOWLEDGEMENT.
        response.setResponseType(ResponseType.ACKNOWLEDGEMENT);
//

    }catch (Exception E) {
        Else if error Prepare an Error object.
        XmlObject error = XmlObject.Factory.parse(body.toString());
        response.setResponseType(ResponseType.ERROR);
        response.setResponse(error);
    }
    return response;
}

public Object getErrorObject()
{
    ErrorResponseDocument errordoc = null;
    errordoc = ErrorResponseDocument.Factory
        .newInstance();
    ErrorResponse newerrorresponse = errordoc
        .addNewErrorResponse();
}

```

```

Error error = newerrorresponse.addNewError();
error.setRaisedBy("SP Application");
error.setNumber(new BigInteger("7000"));
error.setType("Application Error");
String[] str = new String[1];
str[0] = "There was problem processing your request, Please retry after some time.";
error.setTextArray(str);
error.setLocation("SP");
XmlObject returnerror = error.doc;
Object obj=returnerror;
System.out.println("in error obj"+ returnerror.xmlText());
return obj;
}

public void setSessionContext(SessionContext ctx) throws EJBException,
    RemoteException {
    System.out.println("setSessionContext() method called");
    this.ctx = ctx;
}

public void ejbCreate() throws RemoteException,CreateException {
    System.out.println("ejbCreate()");
}

public void ejbRemove() throws EJBException, RemoteException {
    System.out.println("ejbRemove()");
}

public void ejbActivate() throws EJBException, RemoteException {

```

```

System.out.println("ejbActivate()");

}
public void ejbPassivate() throws EJBException, RemoteException {
    System.out.println("ejbPassivate()");
}

```

```
}

```

Snippet 16 : Code for SPConnector Client.java

Discussion:

- i) SP Generic connector calls *callAsyncSubmitRequest ()* method.
- ii) Code in red in *Snippet 16* retrieves the data from Xml format using class `in.cdacmumbai.eservices.asynSubmitReq.FormDocument` which is generated using schema given in *Snippet 5*.
- iii) Insert correlation Id into database against file number as it will be required for sending asynchronous response sometime later.
- iv) Code in blue sets the response parameter to 'Acknowledgement' in `setResponseType` return an object of type `SPResponse` and sends it back to SP Generic Connector.

6.10 Submitting the Response from SP Application to SSDG

In asynchronous request scenario, the SP application after processing the data at some later point from the time of submission submits result to SSDG. SSDG stores the result and forwards it to SAP when SAP polls SSDG for result.

Recipe 7 Submit Response from SP Application to SSDG for Asynchronous Request

As the application sent by SAP is in asynchronous mode, SP after processing the request sends the result to SSDG. SAP need to Poll SSDG for receiving the result submitted by SP Application.

Solution:

SP submits the response in the Body field of Response as an `XmlObject ()` to SSDG along with the values received in the request from SSDG.

```
package in.cdacmumbai.eservices.client;
import in.cdacmumbai.certificateStates.CertificateDocument;
```

```
import in.cdacmumbai.certificateStates.CertificateDocument.Certificate;
import in.cdacmumbai.certificateStates.CertificateResponseDocument.CertificateResponse;
import in.cdacmumbai.certificateStates.StatusResponseDocument.StatusResponse;

import java.io.IOException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.xmlbeans.XmlObject;

public class UpdateStatus extends HttpServlet {

    @Override
    public void init(ServletConfig config) throws ServletException {
        super.init(config);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
        ServletException, IOException {

        //Get Data From HttpServletRequest.
        String fileNo=request.getParameter("FileNO");
        String status=request.getParameter("status");
        String rejectedReason=request.getParameter("reason");

        String corrId="";
        String classId="";
        try
        {
            Connection con=Database.getConnection();
            String query="SELECT correlation_id FROM incomecert where
file_no='"+fileNo.trim()+"'";
            Statement
            stmt=con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCU
R_UPDATABLE);
            ResultSet rs=stmt.executeQuery(query);
```

```

rs.beforeFirst();
if(rs!=null)
{
    while(rs.next())
    {
        corrId=rs.getString("correlation_id");
        classId=rs.getString("class_id");
    } }
System.out.println("correlation id"+corrId);

CertificateDocument certDoc=CertificateDocument.Factory.newInstance();

Certificate cert=certDoc.addNewCertificate();
CertificateResponse certResp=cert.addNewCertificateResponse();
certResp.setServiceID(classId);
certResp.setStatus("SUCCESS");
StatusResponse
statusResp=certResp.addNewResponsedetail().addNewStatusResponse();

if(!rejectedReason.equals(""))
{
    statusResp.setStatusData(status+" Reason : "+rejectedReason);
}
else
{
    statusResp.setStatusData(status);
}

XmlObject resxml=(XmlObject)certDoc;
// Call EJB.
SendCertificateAsResponse resp=new SendCertificateAsResponse();
int st=resp.commitResponse(corrId,"",classId,resxml.toString());
if(st==1)
{
    //Update Status.
    query="UPDATE eforms " + "SET status='"+status+"' " +
        " WHERE eformid = '"+fileNo+"'";

    System.out.println("UPDATED QUERY : "+query);
    boolean b=stmt.execute(query);
    System.out.println("UPDATED STATUS : "+b);
}

```

```
    }  
    catch(Exception e)  
    {  
    }  
}  
  
public void destroy() {  
}  
}
```

Snippet 17 : Sending Response from SP Side for Asynchronous Submit Request

Discussion:

- i) *Snippet 17* shows the response submission to SSDG from SP Application.
- ii) Here class calls ***commitResponse()*** method of *SendResponse* class given in *Snippet 18*
- iii) This class receives the data to be sent to the SSDG from *updatestatus.html* form given in code in pink.
- iv) It then retrieves the correlation Id & class Id of Asynchronous Request received against the File Number from database as shown in code in red.
- v) Code in blue creates *XmlObject ()* using class *CertificateDocument* which is generated using *AsyncPollResponseSchema* in *Snippet 17*.
- vi) It calls ***commitResponse ()*** method of *SendResponse* class and passes appropriate parameters required for sending response to SSDG.

```
package in.cdacmumbai.eservices.client;  
  
import in.cdacmumbai.certificateStates.CertificateDocument;  
import in.cdacmumbai.certificateStates.CertificateDataDocument.CertificateData;  
import in.cdacmumbai.certificateStates.CertificateDocument.Certificate;  
import in.cdacmumbai.certificateStates.CertificateResponseDocument.CertificateResponse;  
import in.cdacmumbai.certificateStates.StatusResponseDocument.StatusResponse;  
  
import java.rmi.RemoteException;  
import java.sql.Connection;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.sql.Statement;
```

```
import java.util.Enumeration;
import java.util.Properties;
import java.util.ResourceBundle;
import javax.ejb.CreateException;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.rmi.PortableRemoteObject;

import org.apache.xmlbeans.XmlObject;

import com.cdac.AsyncSPResponse.SPResponseEJBHome;
import com.cdac.AsyncSPResponse.SPResponseEJBRemote;
import com.cdac.messages.async.ReturnResponse;
import com.cdac.messages.async.CommitResponse;
import com.cdac.messages.async.parameters.RequestStatus;
import com.cdac.messages.async.parameters.ResponseType;
public class SendCertificateAsResponse {

public int commitResponse(String correlationId,String transactionId, String classId,Object
returnResponseBody)
{
    int status=0;
    try
    {
        String value=null;
        String key1=null;
        ResourceBundle labels = ResourceBundle.getBundle("gateway");
        Enumeration bundleKeys = labels.getKeys();
        Properties gateway=new Properties();
        while (bundleKeys.hasMoreElements())
        {
            key1 = (String)bundleKeys.nextElement();
            value = labels.getString(key1);
            gateway.setProperty(key1,value) ;
        }
        String targetEndPoint=(String)gateway.get("targetEndPoint");
        //Call EJB.
        Context ctx = null;
        ctx = new InitialContext();
        SPResponseEJBHome home =
        (SPResponseEJBHome)ctx.lookup("SPConnectorResponseComponent");
        System.out.println("BusinessClass::sendResponse Starts...");
        System.out.println("Got Home object");
```

```

SPResponseEJBRemote remote = home.create();
//process
String ss="<?xml
version='1.0?'><FormDetails><PersonFirstName>FVSIVVJKO</PersonFirstName><Person
MiddleName>OHV
KIVJKCVNA</PersonMiddleName><PersonLastName>CKBVJKVNCVNKN</PersonLast
Name></FormDetails>";
Object resxml=null;
resxml=XmlObject.Factory.parse(ss.toString());
CommitResponse commitResponse=new CommitResponse();
commitResponse.setTargetEndPointURL("http://192.168.0.37:8080/gateway/services/NSDG
Service");
commitResponse.setTransactionId("123456789123456789123456789123");
commitResponse.setClassId("http://202.141.151.140/48");
commitResponse.setResponseType(ResponseType.RESPONSE);
commitResponse.setMessageBody(resxml);
System.out.println("calling EJB");
ReturnResponse returnResponse=remote.makeSubmitResponse(commitResponse);
if (returnResponse.getStatus().equals(RequestStatus.RESUBMIT))
{
    System.out.println("status: "+returnResponse.getStatus());
    System.out.println("Reason:"+returnResponse.getReason());
}
else if (returnResponse.getStatus().equals(RequestStatus.FAILED))
{
    System.out.println("i am end of exec"+ returnResponse.getStatus());
    String[] errorResponse= returnResponse.getErrors();

    for(int i=0;i<errorResponse.length;i++)
    {
        System.out.println("printing error :"+errorResponse[i] );
    }
}
else if(returnResponse.getStatus().equals(RequestStatus.SUCCESS))
{
    System.out.println("status :"+ returnResponse.getStatus());
    System.out.println("correlationId :"+ returnResponse.getCorrelationId());
    System.out.println("TransactionID :"+ returnResponse.getTransactionId());
    System.out.println("AuditID :"+ returnResponse.getAuditId());
}
status=1;
}
}
catch(Exception e)

```

```
{
    e.printStackTrace();
    status=0;
}
return status;
}
```

Snippet 18 : Code for submitting Response from SP to SSDG

Discussion:

- i) Code in blue creates Object of ReturnResponse sets the parameters for Sending Submit Response.
- ii) The subsequent line calls method *makeSubmitResponse ()* of SP Response Component.
- iii) makeSubmitResponse returns the response, after receiving response check the status of response.

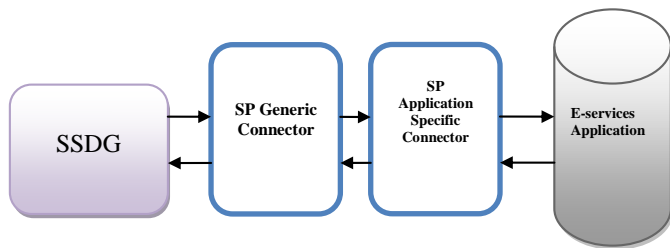


Figure 19 : E-services application submitting response to SSDG for Asynchronous Request

6.11 Polling the SSDG to retrieve the Result

Recipe 8 Poll the SSDG to retrieve the result submitted by SP Application

SAP will retrieve the values received from e-services SP application as per e-services response schema mentioned in *Snippet 4*

Solution:

- i) SAP submits the request through asynchronous mode and gets the submit acknowledgement from SSDG. SAP will use the details given in submit acknowledgement to Poll SSDG to receive the response submitted by SP application.
- ii) SAP then makes a submit poll call to retrieve the response.

Flow of Submit Poll from SAP to SSDG:

PollInfo.html → CheckStatus.java → BusinessClass.java → SAP Generic Connector → SSDG.

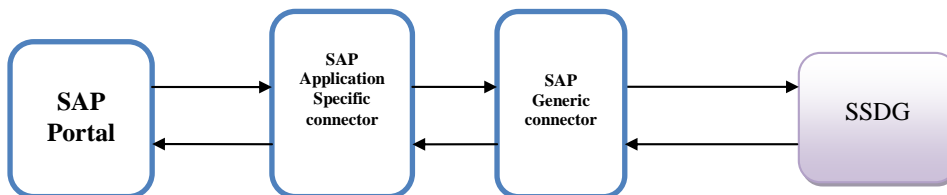


Figure 20 : SAP Polling SSDG for response

```
<html
<head>
<title>Application Status Form</title>
</head>
<body><form name="frmMinPadivam" method="POST" action="PollAction">
</form>
</body>
</html>
```

Snippet 19 : PollInfo.html

```
package in.cdacmumbai.eservices.client;

import in.cdacmumbai.certificateStates.CertificateDocument;
import in.cdacmumbai.certificateStates.CertificateDataDocument.CertificateData;
import in.cdacmumbai.certificateStates.CertificateDocument.Certificate;
import in.cdacmumbai.certificateStates.CertificateResponseDocument.CertificateResponse;
import
in.cdacmumbai.certificateStates.CertificateResponseDocument.CertificateResponse.Response
detail;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import connector.messages.*;
import connector.messages.parameters.*;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```
import org.apache.xmlbeans.XmlException;
import org.apache.xmlbeans.XmlObject;

public class CheckStatus extends HttpServlet {

    @Override
    public void init(ServletConfig config) throws ServletException {
        super.init(config);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {

        //Get Data From HttpServletRequest.

        String fileNo=request.getParameter("fileno");
        boolean flag=true;
        SubmitPollResponse pollResponse=new SubmitPollResponse();
        String corrId="";
        try
        {
            //Extract the Correlation Id against given File Number.
            Connection con=Database.getConnection();
            String query="SELECT correlation_id FROM sappoll_map where
file_no="+fileNo.trim()+"";
            Statement
stmt=con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_
UPDATABLE);
            ResultSet rs=stmt.executeQuery(query);
            rs.beforeFirst();
            if(rs!=null)
            {
                while(rs.next())
                {
                    corrId=rs.getString("correlation_id");
                }
            }
            System.out.println("correlation id"+corrId);
            rs.close();
            stmt.close();
            con.close();
        }
    }
}
```

```

//Call SAP Connector
    BusinessClass bb= new BusinessClass();
//Call Synchronous Service of Connector.
    pollResponse=bb.makeSubmitPollCall(corrId);
    flag=false;
}
catch(Exception e)
{
    flag=true; //raise error condition.
}

if(!flag) //if NO error
{
    if(corrId!=null)
    {

        if(pollResponse.getStatus().equals(RequestStatus.SUCCESS))
        {

            if(pollResponse.getResponseDocumentType().equals(ResponseDocumentType.ACKNOWLEDGEMENT))
            {

                System.out.println("Within acknowledgement");
                System.out.println("Correlation ID : "+pollResponse.getCorrelationId());
                System.out.println("Response End PointURL :
"+pollResponse.getResponseEndPointURL());
                System.out.println("Poll Interval : "+pollResponse.getPollInterval());

                RequestDispatcher rd=request.getRequestDispatcher("/ack.jsp");
                request.setAttribute("File No", fileNo);
                request.setAttribute("result","ACK");
                rd.forward(request,response);

            }
        }
    }
else
if(pollResponse.getResponseDocumentType().equals(ResponseDocumentType.RESPONSE))
{

    System.out.println("Within Response");
    XmlObject obj=(XmlObject) pollResponse.getResponse();
    RequestDispatcher rd=request.getRequestDispatcher("/ack.jsp");

}
}
try

```

```
{
    CertificateDocument
certDoc=CertificateDocument.Factory.parse(obj.toString());

    Certificate cert=certDoc.getCertificate();
    CertificateResponse certResp=cert.getCertificateResponse();
    Responsetdetail respDetail=certResp.getResponsedetail();
    request.setAttribute("File No",certResp.getServiceID());
    request.setAttribute("status",certResp.getStatus());

    if(respDetail.isSetCertificateData())
    {
        System.out.println("IN isSetCertificateData");
        request.setAttribute("result","Certificate");
        CertificateData certData=respDetail.getCertificateData();
        String certName="/Cert/"+certData.getFileName();
        request.setAttribute("File No", fileNo);
        request.setAttribute("filename",certName );
        File file=new
File(request.getSession().getServletContext().getRealPath("/") +certName);
        FileOutputStream fout=new FileOutputStream(file);
        fout.write(certData.getData());
        fout.flush();
        fout.close();
    }
    else if(respDetail.isSetStatusResponse())
    {
        System.out.println("IN isSetStatusResponse");
        request.setAttribute("File No", fileNo);
        request.setAttribute("result","Status");
        String data=respDetail.getStatusResponse().getStatusData();
```

```
request.setAttribute("data",data);
    }
    else
    if(respDetail.isSetBusinessError())
    {
        System.out.println("IN isSetBusinessError");
        request.setAttribute("result","Error");
    }
```

```

        rd.forward(request,response);
    }
    catch(XmlException e)
    {

    }
}
else if(pollResponse.getStatus().equals(RequestStatus.FAILED))
{
    System.out.println("Request FAILED.");
    String [] errors=pollResponse.getErrors();
    String error="";
    for(int i=0;i<errors.length;i++)
    {
        System.out.println(errors[i]);
        error=error+"\n"+errors[i];
    }

    rd.forward(request,response);
}

}
}
else
{
    RequestDispatcher rd=request.getRequestDispatcher("/error.jsp");
    request.setAttribute("error","There is some problem in processing your request. <br/>
Please Try again later.");
    rd.forward(request,response);
}}public void destroy() { }}

```

Snippet 20 : Code for CheckStatus.java

Discussion:

- i) Code in blue in *Snippet 20* retrieves the citizen file number.
- ii) The Correlation Id of the async request for this file number is then retrived from database.
- iii) Code in pink calls the *makeSubmitPollCall ()* method.
- iv) makeSubmitPollCall returns response and check its status.

```
package in.cdacmumbai.eservices.client;

import org.apache.xmlbeans.XmlException;
import org.apache.xmlbeans.XmlObject;

import connector.sap.SAPConnector;
import connector.sap.SAPConnectorImpl;

import java.util.*;

public class BusinessClass {
    public SubmitPollResponse makeSubmitPollCall(String correlationId){

        String classId="http://192.168.0.140/130";

        String transactionId="";
```

```
// String correlationId="5D30E369B9AD4670B761C17BBA14E672";
String
targetEndPoint="http://nsdgstaging.cdacmumbai.in/gateway/services/NSDGService";
SubmitPoll pollRequest=new SubmitPoll();
SubmitPollResponse pollResponse=new SubmitPollResponse();
pollRequest.setTargetEndPointURL(targetEndPoint);
    pollRequest.setClassId(classId);
    pollRequest.setTransactionId(transactionId);
    pollRequest.setCorrelationId(correlationId);
    pollRequest.setNsdgTest(NSDGTTest.NO );

    SAPConnector connector = new SAPConnectorImpl();

    pollResponse = connector.makeSubmitPoll(pollRequest);

    return pollResponse;
}
```

*Snippet 21 : Submit Poll Operation***Discussion:**

- i) Parameters required for Submit Poll are called using setter methods of SubmitPoll as seen in highlighting done in blue.
- ii) Code highlighted in pink calls the `makeSubmitPoll()` of SAP Generic Connector.

6.12 Retrieve the result of Submit Poll method

After the Submit Poll Request, Gateway returns an acknowledgement in case SP has not submitted response or returns Response submitted by SP Application. SAP will then retrieve values from the response.

Recipe 9 Retrieve the result of Submit Poll method**Solution**

- i) If SAP receives a response on polling SSDG, SAP reads the response in `CertificateDocument` class fields which is generated from AsyncPollResponse schema in *Snippet 22*
- ii) For retrieving all fields, SAP first needs to deserialize the XML using XmlObject () sent by SP in the Body field of SAP generic connector.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:be="http://www.mit.gov.in/eGov/schema/NSDG/errorresponse"
  targetNamespace="http://www.cdacmumbai.in/AsyncPollResponse"
  xmlns="http://www.cdacmumbai.in/AsyncPollResponse"
  elementFormDefault="qualified" version="1.0"
  attributeFormDefault="unqualified" id="AsyncPollResponse">

  <xsd:import
    namespace="http://www.mit.gov.in/eGov/schema/NSDG/errorresponse"
    schemaLocation="Bussinesserror.xsd" />

  <xsd:element name="Certificate">
```

```

        <xsd:complexType>
            <xsd:choice>

                <xsd:element ref="CertificateResponse" />
                <xsd:element ref="PersonelInfoRequest" />
                <xsd:element ref="PersonelInfoResponse" />
            </xsd:choice>
        </xsd:complexType>
    </xsd:element>
    <!-- In this schema if certificate data is not avaiilable for any bussiness reason you
    can send bussiness Error -->
    <xsd:element name="CertificateResponse">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="ServiceID" type="xsd:string" />
                <xsd:element name="status" type="xsd:string" />
                <xsd:element name="responsetdetail">
                    <xsd:complexType>
                        <xsd:choice>
                            <xsd:element
ref="certificateData" />
                            <xsd:element
name="BusinessError">
                                <xsd:complexType>
                                    <xsd:sequence>
                                        <xsd:element
ref="be:ErrorResponse" />
                                    </xsd:sequence>
                                </xsd:complexType>
                            </xsd:element>
                        </xsd:choice>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    </xsd:element>
    </xsd:choice>
</xsd:complexType>
</xsd:element>
</xsd:element>
ref="StatusResponse" />
</xsd:choice>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>

```

```

    <xsd:element name="StatusResponse">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="StatusData" type="xsd:string" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>

    <xsd:element name="certificateData">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="FileName" type="xsd:string" />
          <xsd:element name="FileType" type="xsd:string" />
          <xsd:element name="data" type="xsd:base64Binary" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>

    <xsd:element name="PersonelInfoRequest">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="RegistrationID" type="xsd:string"
/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="PersonelInfoResponse">
      <xsd:complexType>
        <xsd:sequence>

<xsd:element name="Name">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:pattern value="([a-
zA-Z.])*"/>
          <xsd:minLength
value="1"/>
          <xsd:maxLength
value="99"/>

```

```

        </xsd:restriction>
        </xsd:simpleType>
    </xsd:element>

    <xsd:element name="ShortName">
        <xsd:simpleType>
            <xsd:restriction base="xsd:string">
                <xsd:pattern value="[a-
zA-Z .]*/>
                <xsd:minLength
value="1"/>
                <xsd:maxLength
value="99"/>
            </xsd:restriction>
        </xsd:simpleType>
    </xsd:element>

    <xsd:element name="GuardianType">
        <xsd:simpleType>
            <xsd:restriction base="xsd:string">
                <xsd:pattern
value="Father|Husband|Guardian"/>
            </xsd:restriction>
        </xsd:simpleType>
    </xsd:element>

    <xsd:element name="GuardianName">
        <xsd:simpleType>
            <xsd:restriction base="xsd:string">
                <xsd:pattern value="[a-
zA-Z .]*/>
                <xsd:minLength
value="1"/>
                <xsd:maxLength
value="99"/>
            </xsd:restriction>
        </xsd:simpleType>
    </xsd:element>

    <xsd:element name="MotherName">
        <xsd:simpleType>
            <xsd:restriction base="xsd:string">
                <xsd:pattern value="[a-
zA-Z .]*/>

```

```

value="1"/>
value="99"/>
value="8"/>
value="10"/>
value="M|F|T"/>
value="01|02|03|04|05|06|07|08|09|10"/>

```

```

<xsd:minLength
<xsd:maxLength
</xsd:restriction>
</xsd:simpleType>
</xsd:element>

<xsd:element name="DateOfBirth">
  <xsd:simpleType>
  <xsd:restriction base="xsd:string">
    <xsd:minLength
      <xsd:maxLength
    </xsd:restriction>
    </xsd:simpleType>
  </xsd:element>

  <xsd:element name="Gender">
    <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern
    </xsd:restriction>
    </xsd:simpleType>
  </xsd:element>

  <xsd:element name="Caste">
    <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

  <xsd:element name="Religion">
    <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="0[1-7]"/>

```

```

        </xsd:restriction>
        </xsd:simpleType>
    </xsd:element>

    <xsd:element name = "MaritalStatus" >
        <xsd:simpleType>
            <xsd:restriction base="xsd:string">
                <xsd:pattern value="[1-
4]"/>

                </xsd:restriction>
            </xsd:simpleType>
        </xsd:element>

        <xsd:element name="IdentificationMark">
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:minLength
value="1"/>
                    <xsd:maxLength
value="50"/>

                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>

            <xsd:element name="EducationAttained">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:pattern value="(0)[1-
9][1-2][0-9][30|31|99]"/>

                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>

                <xsd:element name="Occupation" >
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:pattern value="(0)[1-
9][1-2][0-9][30]"/>

                            </xsd:restriction>
                        </xsd:simpleType>
                    </xsd:element>

```

```

<xsd:element name="Nationality">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="([a-
zA-Z ])*"/>
      <xsd:minLength
value="1"/>
      <xsd:maxLength
value="50"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

<xsd:element name="PresentState">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:minLength
value="1"/>
      <xsd:maxLength
value="50"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

<xsd:element name="PresentDistrict">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:minLength
value="1"/>
      <xsd:maxLength
value="50"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

<xsd:element name="PresentSubDistrict">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:minLength
value="1"/>
      <xsd:maxLength
value="60"/>
    </xsd:restriction>
  </xsd:simpleType>

```

```
</xsd:element>

<xsd:element name="PresentCity">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:minLength
value="1"/>
      <xsd:maxLength
value="50"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

<xsd:element name="PresentPostOffice">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:minLength
value="1"/>
      <xsd:maxLength
value="60"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

<xsd:element name="PresentBlockNo">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:minLength
value="1"/>
      <xsd:maxLength
value="60"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

<xsd:element name="PresentHouseNo">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:minLength
value="1"/>
      <xsd:maxLength
value="60"/>
    </xsd:restriction>
  </xsd:simpleType>
```

```

</xsd:element>

<xsd:element name="PresentPinCode">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[0-
9]{6}"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

<xsd:element name="PermanentState">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:minLength
value="1"/>
      <xsd:maxLength
value="50"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

<xsd:element name="PermanentDistrict">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:minLength
value="1"/>
      <xsd:maxLength
value="50"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

<xsd:element name="PermanentSubDistrict">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:minLength
value="1"/>
      <xsd:maxLength
value="60"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

```

```
<xsd:element name="PermanentCity">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:minLength
value="1"/>
      <xsd:maxLength
value="50"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

<xsd:element name="PermanentPostOffice">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:minLength
value="1"/>
      <xsd:maxLength
value="60"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

<xsd:element name="PermanentBlockNo">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:minLength
value="1"/>
      <xsd:maxLength
value="60"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

<xsd:element name="PermanentHouseNo">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:minLength
value="1"/>
      <xsd:maxLength
value="60"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

```
<xsd:element name="PermanentPinCode">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[0-9]{6}" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

<xsd:element name="IPAddress">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\.[0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\.[0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\.[0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

Snippet 22 : Asynchronous Poll Response Schema

```
package in.cdacmumbai.eservices.client;

import in.cdacmumbai.certificateStates.CertificateDocument;
import in.cdacmumbai.certificateStates.CertificateDataDocument.CertificateData;
import in.cdacmumbai.certificateStates.CertificateDocument.Certificate;
import in.cdacmumbai.certificateStates.CertificateResponseDocument.CertificateResponse;
import
in.cdacmumbai.certificateStates.CertificateResponseDocument.CertificateResponse.Response
detail;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.xmlbeans.XmlException;
import org.apache.xmlbeans.XmlObject;
import connector.messages.*;
import connector.messages.parameters.*;
public class CheckStatus extends HttpServlet {

    @Override
    public void init(ServletConfig config) throws ServletException {
        super.init(config);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {

        //Get Data From HttpServletRequest.
        SubmitPollResponse pollResponse=new SubmitPollResponse();
```

```
String fileNo=request.getParameter("fileno");
boolean flag=true;
String corrId="";
try
{
//Extract the Correlation Id against given File Number.
    Connection con=Database.getConnection();
    String query="SELECT correlation_id FROM sappoll_map where
file_no="+fileNo.trim()+"";
    Statement
stmt=con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_
UPDATABLE);
    ResultSet rs=stmt.executeQuery(query);
    rs.beforeFirst();
    if(rs!=null)
    {
        while(rs.next())
        {
            corrId=rs.getString("correlation_id");
        }
    }
    System.out.println("correlation id"+corrId);
    rs.close();
    stmt.close();
    con.close();

//Call SAP Connector
    BusinessClass bb= new BusinessClass();
//Call Synchronous Service of Connector.
    pollResponse =bb.makeSubmitPollCall(corrId);
    flag=false;
}
catch(Exception e)
{
    flag=true; //raise error condition.
}

if(!flag) //if NO error
{
    if(corrId!=null)
    {
if(pollResponse.getStatus().equals(RequestStatus.SUCCESS))
```

```
{
if(pollResponse.getResponseDocumentType().equals(ResponseDocumentType.ACKNOWLEDGEMENT))
{
System.out.println("Correlation ID : "+pollResponse.getCorrelationId());
System.out.println("Response End PointURL : "+pollResponse.getResponseEndPointURL());
System.out.println("Poll Interval : "+pollResponse.getPollInterval());

RequestDispatcher rd=request.getRequestDispatcher("/ack.jsp");
request.setAttribute("File No", fileNo);
request.setAttribute("result","ACK");
rd.forward(request,response);

}

if(pollResponse.getResponseDocumentType().equals(ResponseDocumentType.RESPONSE))
{
System.out.println("Within Response");
XmlObject obj=(XmlObject) pollResponse.getResponse() ;
RequestDispatcher rd=request.getRequestDispatcher("/ack.jsp");

try
{
CertificateDocument
certDoc=CertificateDocument.Factory.parse(obj.toString());

Certificate cert=certDoc.getCertificate();
CertificateResponse certResp=cert.getCertificateResponse();
Responsedetail respDetail=certResp.getResponsedetail();
request.setAttribute("File No",certResp.getServiceID());
request.setAttribute("status",certResp.getStatus());

if(respDetail.isSetCertificateData())
{
System.out.println("IN isSetCertificateData");
request.setAttribute("result","Certificate");
CertificateData certData=respDetail.getCertificateData();
```

```
        String certName="/Cert/"+certData.getFileName();
        request.setAttribute("File No", fileNo);
        request.setAttribute("filename",certName );
        File file=new
File(request.getSession().getServletContext().getRealPath("/") +certName);
        FileOutputStream fout=new FileOutputStream(file);
        fout.write(certData.getData());
        fout.flush();
        fout.close();
    }
    else if(respDetail.isSetStatusResponse())
    {
        System.out.println("IN isSetStatusResponse");
        request.setAttribute("File No", fileNo);
        request.setAttribute("result","Status");
        String data=respDetail.getStatusResponse().getStatusData();

        request.setAttribute("data",data);
    }
    else
    if(respDetail.isSetBusinessError())
    {
        System.out.println("IN isSetBusinessError");
        request.setAttribute("result","Error");
    }

    rd.forward(request,response);
}
catch(XmlException e)
{
}
}
}
else
{
    if(pollResponse.getStatus().equals(RequestStatus.FAILED))
    {
System.out.println("Request FAILED.");
        String [] errors=pollResponse.getErrors();
        String error="";
        for(int i=0;i<errors.length;i++)
        {
            System.out.println(errors[i]);
        }
    }
}
```

```
        error=error+"\n"+errors[i];
    }
    RequestDispatcher rd=request.getRequestDispatcher("error.jsp");
    request.setAttribute("message"," There is Some Problem.Please,Try After
SomeTime.");

    rd.forward(request,response);
}

}
}
else
{
    RequestDispatcher rd=request.getRequestDispatcher("/error.jsp");
    request.setAttribute("error","There is some problem in processing yuor request. <br/>
Please Try again later.");
    rd.forward(request,response);
}
}

public void destroy() {
}
}
```

Snippet 23 : Code for CheckStatus.java

Discussion:

- i) Submit method call given in code in pink returns a SubmitResponse which contains details of response from SSDG.
- ii) IF condition given in code in blue checks if Acknowledgement is received.
- iii) IF condition given in code in pink checks if Response is received from SP. If yes, then values of response are retrieved using CertificateDocument class which is generated using code given in *Snippet 23*.
- iv) Code in green retrieves the data sent by SP in response.

6.13 Deleting Request submitted to SSDG

SAP can request the deletion of requests for which SAP has already received the response through document poll.

Recipe 10 Executing Delete Request

Requests for which response has been received may be deleted using the *makeDeleteRequest()* operation.

Solution:

Call method *makeDeleteRequest ()* defined in SAP generic connector.If there is response the record is successfully deleted from the SSDG.If there is error, read error details and correct the document for resending document for deletion request. *Snippet 24* shows the calling of *makeDeleteRequest ()* method on SAP generic connector.

```
package in.cdacmumbai.eservices.client;

import org.apache.xmlbeans.XmlException;
import org.apache.xmlbeans.XmlObject;

import connector.sap.SAPConnector;
import connector.sap.SAPConnectorImpl;

import java.util.*;

import connector.messages.*;
import connector.messages.parameters.*;

public class BusinessClass {

public DeleteResponse makeDeleteRequest(DeleteRequest deleteRequest){
DeleteResponse deleteResponse=new DeleteResponse();

String classId="http://192.168.0.140/130";
String transactionId="";
String correlationId="IDD174252C6648A38D6E6D410B5F43";
```

```

String targetEndPoint="http://192.168.0.37:8080/gateway/services/NSDGService";
deleteRequest.setTargetEndPointURL(targetEndPoint);
deleteRequest.setClassId(classId);
deleteRequest.setTransactionId(transactionId);
deleteRequest.setCorrelationId(correlationId);
deleteRequest.setNsdgTest(NSDGTTest.NO );
SAPConnector connector = new SAPConnectorImpl();
deleteResponse = connector.makeDeleteRequest(deleteRequest );
if(deleteResponse.getStatus().equals(RequestStatus.RESUBMIT))
{
    System.out.println("Request failed. Please RESUBMIT.");
    System.out.println(deleteResponse.getReason());

}
else if(deleteResponse.getStatus().equals(RequestStatus.FAILED))
{
    System.out.println("Request FAILED.");
    String [] errors=deleteResponse.getErrors();
    String error="";
    for(int i=0;i<errors.length;i++)
    {
        System.out.println(errors[i]);
        error=error+"\n"+errors[i];
    }
}
else if(deleteResponse.getStatus().equals(RequestStatus.SUCCESS))
{
    //Check what is received - Response or ACKNOWLEDGEMENT.
if(deleteResponse.getResponseDocumentType().equals(ResponseDocumentType.RESPONSE
))
    {
        System.out.println("Response is received.");
    }
elseif(deleteResponse.getResponseDocumentType().equals(ResponseDocumentType.ACKN
OWLEDGEMENT))
    {
        System.out.println("Correlation ID : "+deleteResponse.getCorrelationId());
        System.out.println("Response End PointURL :
"+deleteResponse.getResponseEndPointURL());
        System.out.println("Poll Interval : "+deleteResponse.getPollInterval());
    }
}
return deleteResponse;

```

```
}  
}
```

Snippet 24 : Code for Delete Request

Discussion:

- i) Code in pink shows calling setter methods of DeleteRequest class.
- ii) Code in blue calls the *makeDeleteRequest()* method of SAP generic Connector.
- iii) The DeleteResponse contains the values returned by SSDG. It contains the status of the makeDeleteRequest i.e. either SUCCESS (meaning Record has been deleted) or FAILED (if any error occurs e.g. Record doesn't exist).

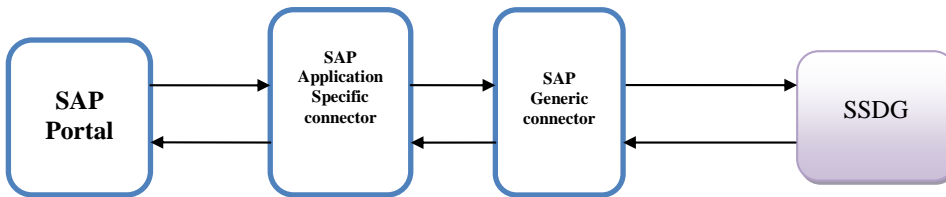


Figure 21 : SAP calling delete request for previously submitted requests.

6.14 Listing Requests submitted to SSDG

There are two types of List Request methods

- List Request for Submit Response from SP
- List Request for Submit Requests from SAP to SP

Recipe 11 Executing List Request for Submit Response

For enquiring about the previous requests which were sent to SP during a specific time interval, SAP uses List Request operation by specifying the schema as specified in Java Connector Manual which will produce a list of requests that were sent to SP and also the status of all such requests

Solution:

Call method *makeListRequestForSubmitResponse ()* defined in SAP generic connector and pass the necessary parameters listed in section as *List Request Input Parameters* in the JAVA Connector Development Manual

Below is the code snippet which shows how the application specific connector can call the *makeListRequestForSubmitResponse* method on generic connector for Submit Request.

```
package in.cdacmumbai.eservices.client;

import connector.messages.*;
import connector.messages.parameters.*;
import org.apache.xmlbeans.XmlException;
import org.apache.xmlbeans.XmlObject;

import connector.sap.SAPConnector;
import connector.sap.SAPConnectorImpl;
import java.util.*;

public class BusinessClass {
public ListResponse makeListRequestForSubmitResponse(ListRequest listRequest){
```

```
String classId="http://192.168.0.140/130";
String transactionId="";
String targetEndPoint="http://192.168.0.37:8080/gateway/services/NSDGService";
String stTime="02/11/2010";
String enTime="05/11/2010";
    DateFormat formatter ;
    Date date ;
    formatter = new SimpleDateFormat("dd/MM/yyyy");
    date = (Date)formatter.parse(stTime);
    System.out.println("Start Time : "+date);
    Calendar startTime=Calendar.getInstance();
    startTime.setTime(date);

    date = (Date)formatter.parse(enTime);
    Calendar endTime=Calendar.getInstance();
    endTime.setTime(date);
    System.out.println("End Time : "+date);

    listRequest.setTargetEndPointURL(targetEndPoint);
    listRequest.setClassId(classId);
    listRequest.setTransactionId(transactionId);
    listRequest.setNsdgTest(NSDGTTest.NO);
    listRequest.setAuthenticationType(AuthenticationType.CLEAR);
    listRequest.setStartTime(startTime);
    listRequest.setEndTime(endTime);

    System.out.println("List Request Object Prepared.");

    SAPConnector connector = new SAPConnectorImpl();
    ListResponse listResponse=new ListResponse();
    listResponse = connector.makeListRequestForSubmitResponse(listRequest);

    if(listResponse.getStatus().equals(RequestStatus.RESUBMIT))
    {
        System.out.println("Request failed. Please RESUBMIT.");
        System.out.println(listResponse.getReason());
    }
    if(listResponse.getStatus().equals(RequestStatus.FAILED))
    {
        System.out.println("Request FAILED.");
        String [] errors=listResponse.getErrors();
        String error="";
        for(int i=0;i<errors.length;i++)
        {
```

```

        System.out.println(errors[i]);
        error=error+"\n"+errors[i];
    }
}
if(listResponse.getStatus().equals(RequestStatus.SUCCESS))
{
    System.out.println("LIST RESP DOC : "+listResponse.getResponse().toString());
}
return listResponse;
}
}

```

Snippet 25 : List Request for Submit Response

Discussion:

- i) Code in blue shows calling setter methods of ListRequest for making List Request call.
- ii) Code in pink calls *makeListRequestForSubmitResponset ()* of SAP Generic Connector.
- iii) ListResponse contains the returned list of submit resposnes for given period.

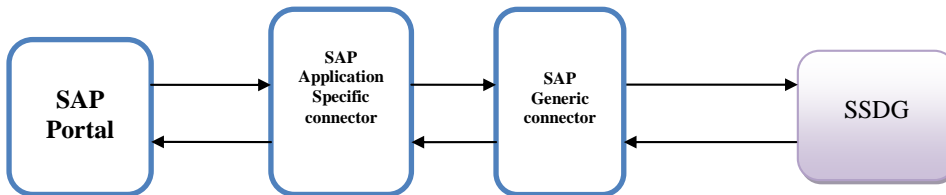


Figure 22 : List Request to track the status of all requests and response from SSDG

Recipe 12 Executing List Response for Submit Request

For enquiring about the responses sent back by the SP against the requests made by SAP during a specific time interval, SAP uses list request operation by specifying the start time and end time which will produce a list of responses that were sent by SP

Solution:

Below is the code snippet which shows how the application specific connector can call the `makeListRequestForSubmitRequest` method on generic connector for Submit Request.

```
package in.cdacmumbai.eservices.client;

import connector.messages.*;
import connector.messages.parameters.*;
import org.apache.xmlbeans.XmlException;
import org.apache.xmlbeans.XmlObject;

import connector.sap.SAPConnector;
import connector.sap.SAPConnectorImpl;
import java.util.*;

public class BusinessClass {
public ListResponse makeListRequestForSubmitResponse(ListRequest listRequest){
    String classId="http://192.168.0.140/130";
    String transactionId="";
    String targetEndPoint="http://192.168.0.37:8080/gateway/services/NSDGService";
String stTime="02/11/2010";
String enTime="05/11/2010";
    DateFormat formatter ;
    Date date ;
    formatter = new SimpleDateFormat("dd/MM/yyyy");
    date = (Date)formatter.parse(stTime);
    System.out.println("Start Time : "+date);
    Calendar startTime=Calendar.getInstance();
    startTime.setTime(date);

    date = (Date)formatter.parse(enTime);
    Calendar endTime=Calendar.getInstance();
    endTime.setTime(date);
    System.out.println("End Time : "+date);

    listRequest.setTargetEndPointURL(targetEndPoint);
    listRequest.setClassId(classId);
    listRequest.setTransactionId(transactionId);
    listRequest.setNsdgTest(NSDGTTest.NO);
}
```

```
listRequest.setAuthenticationType(AuthenticationType.CLEAR);
listRequest.setStartTime(startTime);
listRequest.setEndTime(endTime);

System.out.println("List Request Object Prepared.");

SAPConnector connector = new SAPConnectorImpl();
ListResponse listResponse=new ListResponse();
listResponse = connector. makeListRequestForSubmitResponse(listRequest);

if(listResponse.getStatus().equals(RequestStatus.RESUBMIT))
{
    System.out.println("Request failed. Please RESUBMIT.");
    System.out.println(listResponse.getReason());
}
if(listResponse.getStatus().equals(RequestStatus.FAILED))
{
    System.out.println("Request FAILED.");
    String [] errors=listResponse.getErrors();
    String error="";
    for(int i=0;i<errors.length;i++)
    {
        System.out.println(errors[i]);
        error=error+"\n"+errors[i];
    }
}
if(listResponse.getStatus().equals(RequestStatus.SUCCESS))
{
    System.out.println("LIST RESP DOC : "+listResponse.getResponse().toString());
}
return listResponse;
}
}
```

Snippet 26 : List Request for Submit Request

Discussion:

- i) Code in blue in *Snippet 26* calls the setter methods of ListRequest class.
- ii) Code in blue calls `makeListRequestForSubmitRequest ()` of SAP Generic Connector.
- iii) ListResponse returns list of Requests made for specified time period.

Additional Resources

This cookbook has been developed to help the developers in getting their job done. The examples and scenarios used are as self explanatory and lucid as can be reasonable along with a fair portion of description, explanation and screen shots to drive home the point to the developers. A fair number of scenarios have been covered in this cookbook. If in the process of reading the recipes in this cookbook, the user is left with some unanswered questions, he may refer the Java Connector Development Manual to find more information.