

2013

User Manual

AspectBPEL Plug-ins

This document is a user manual that explains how to install the AspectBPEL plug-ins within Eclipse and their main requirements. It provides also a graphical interface to illustrate step by step how to use these plug-ins



Contents

PREFACE	3
Purpose of This Document.....	3
Intended Users.....	3
Requirements.....	3
Introduction to AspectBPEL	4
AspectBPEL Language	4
AspectBPEL Language Construct.....	4
AspectBPEL Plug-ins	6
Installation	6
Getting Started.....	6
1. Create the AspectBPEL aspect model	6
2. Generate/Open AspectBPEL Aspect	8
2.1. Generate AspectBPEL Aspect Code.....	8
2.2. Open AspectBPEL Aspect	8
3. Compile AspectBPEL Aspect Model	9
4. Weave AspectBPEL Aspect Model	10

PREFACE

Purpose of This Document

This user manual aims to familiarize you with the AspectBPEL plug-ins. It describes how to install these plug-ins and provides a user interface to illustrate their main functionalities. The document includes also a description of the AspectBPEL language construct and objective.

Intended Users

This document is intended for the users of Eclipse who are familiar with Web Services and BPEL processes. It will enable you to understand how to create an AspectBPEL aspect model, generate its corresponding AspectBPEL code, compile it and weave it in a BPEL process.

Requirements

Before running the plug-ins, you have to make sure that you have a **new version of Eclipse with BPEL installed**. To install BPEL, you can go to Help → Install New Software then insert the following link: <http://download.eclipse.org/bpel/site/>

On the other hand, **your computer must be connected to the internet**.

Introduction to AspectBPEL

AspectBPEL Language

The AspectBPEL language allows the description and specification of security BPEL aspects. It is a minimalist language built on top of the current AOP (Aspect Oriented Programming) technologies that are based on advice-pointcut model. We developed part of the *AspectBPEL* with notations and expressions close to those of the current AOP languages, but with all the abstraction needed to specify the security BPEL aspects. These notations and expressions are programming language independent and without referring to low-level implementation details. The following are the main features provided by *AspectBPEL*:

- Automatic code manipulation such as code addition, substitution, deletion, etc.
- Specification of particular join points where security code would be injected.
- Description and specification of BPEL aspects.
- Description and specification of reusable BPEL aspects.
- Independency of programming language.
- Intermediary abstractness between English and XPath Language.

AspectBPEL Language Construct

A BPEL aspect starts with the keyword `Aspect`, followed by the aspect name. Next comes the aspect code that starts and ends respectively by the keywords `BeginAspect` and `EndAspect`. The aspect code is based on AOP and consists of one or many `BPEL_Location_Behavior` constructs. Each is composed of a `BPEL_Insertion_Point` and `BPEL_Location_Identifier`, where the behavior code should be injected.

The list of AspectBPEL pointcuts, identified in our language as `BPEL_Location_Identifier`, is grouped into three categories: *Actions*, *Controls* and *Faults*.

These pointcuts allow matching all the activities of a BPEL process, where a BPEL code can be inserted before, after or even replace the matched one.

The name of each activity is enclosed in `< :: >` symbols that follows the pointcut. The activity's name represents the activity's signature.

A detailed explanation of the components of the aspect code will be illustrated in the next page.

<i>BPEL_Aspect</i>	::=	Aspect <i>Aspect_Name</i> <i>BPEL_Aspect_Body</i>
<i>BPEL_Aspect_Body</i>	::=	BeginAspect <i>BPEL_Location_Behavior</i> * EndAspect
<i>BPEL_Location_Behavior</i>	::=	<i>BPEL_Insertion_Point</i> <i>BPEL_Location_Identifier</i> <i>BPEL_Behavior_Code</i>
<i>BPEL_Insertion_Point</i>	::=	Before After Replace
<i>BPEL_Location_Identifier</i>	::=	Assign < <i>Signature</i> > Invoke < <i>Signature</i> > Receive < <i>Signature</i> > Reply < <i>Signature</i> > Empty < <i>Signature</i> > If < <i>Signature</i> > Pick < <i>Signature</i> > While < <i>Signature</i> > Foreach < <i>Signature</i> > RepeatUntil < <i>Signature</i> > Wait < <i>Signature</i> > Sequence < <i>Signature</i> > Scope < <i>Signature</i> > Flow < <i>Signature</i> > Exit < <i>Signature</i> > Throw < <i>Signature</i> > Rethrow < <i>Signature</i> >
<i>BPEL_Behavior_Code</i>	::=	BeginBehavior <i>BPEL_Code_Statements</i> EndBehavior

a) *BPEL Location Behavior*: Is based on the advicepointcut model of AOP. It is the abstract representation of an advice-pointcut combination in an aspect. An aspect may include one or many *BPEL_Location_Behavior*. Each *BPEL_Location_Behavior* is composed of the *BPEL_Insertion_Point*, *BPEL_Location_Identifier* and *BPEL_Behavior_Code*.

b) *BPEL Insertion Point*: Specifies the point of code insertion after identifying the location. The *BPEL_Insertion_Point* can have the following three values: Before, After or Replace. The Replace means remove the code at the identified location and replace it with the new code, while the Before or After means keep the old code at the identified location and insert the new code before or after it respectively.

c) *BPEL Location Identifier*: Identifies the joint point or sets of joint points in the program where the changes specified in the *BPEL_Behavior_Code* should be applied. The list of identifiers used in the *BPEL_Location_Identifier* corresponds to BPEL activities together with their signatures (i.e. name, id and parameters). The activities used for identifying locations are divided into three categories: *Actions*, *Control* and *Fault*.

d) *BPEL Behavior Code*: Contains code written in XPath language, that will be weaved to a BPEL Process. The code will be inserted before/after or replace the location identifier previously stated.

AspectBPEL Plug-ins

Installation

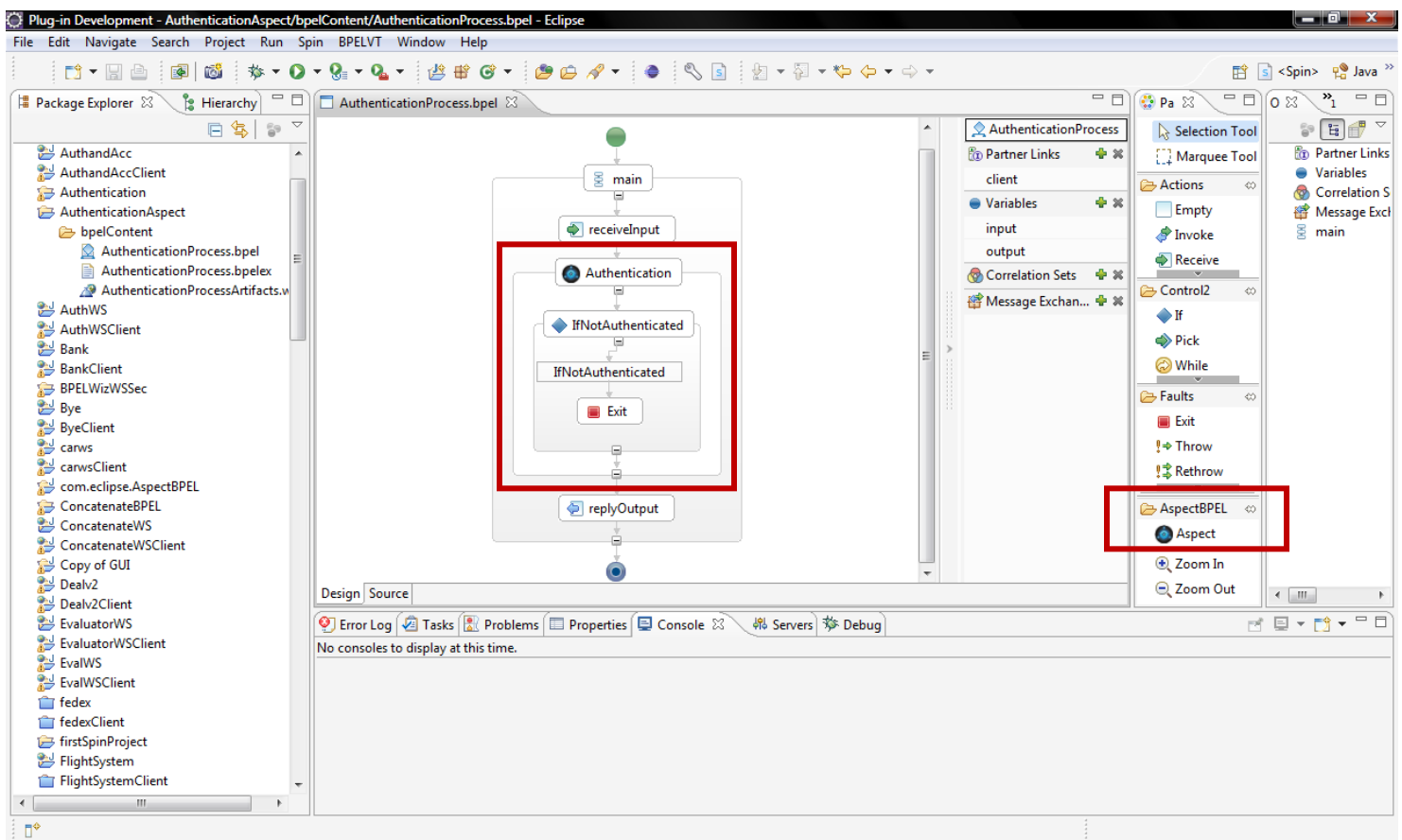
In order to run the AspectBPEL plug-ins, copy the following jar files into the eclipse plugins folder:

- org.eclipse.bpel.aspectbpel.candw_1.0.0.201301081244.jar
- org.eclipse.bpel.aspectbpel.model_1.0.0.jar
- org.eclipse.bpel.aspectbpel.ui_1.0.0.201301081245.jar
- org.antlr.runtime_3.0.0.v200803061811.jar

Getting Started

1. Create the AspectBPEL aspect model

To start creating the aspect model, you have to create first a new BPEL process in your workspace. Then, drag the “Aspect” element from the Palette, drop it in your process and add the BPEL activities in the inserted aspect. These activities represent the code that will be weaved later on in the BPEL process. (It’s the *BPEL_Behavior_Code* element in AspectBPEL) The figure below shows an example of an Authentication aspect.



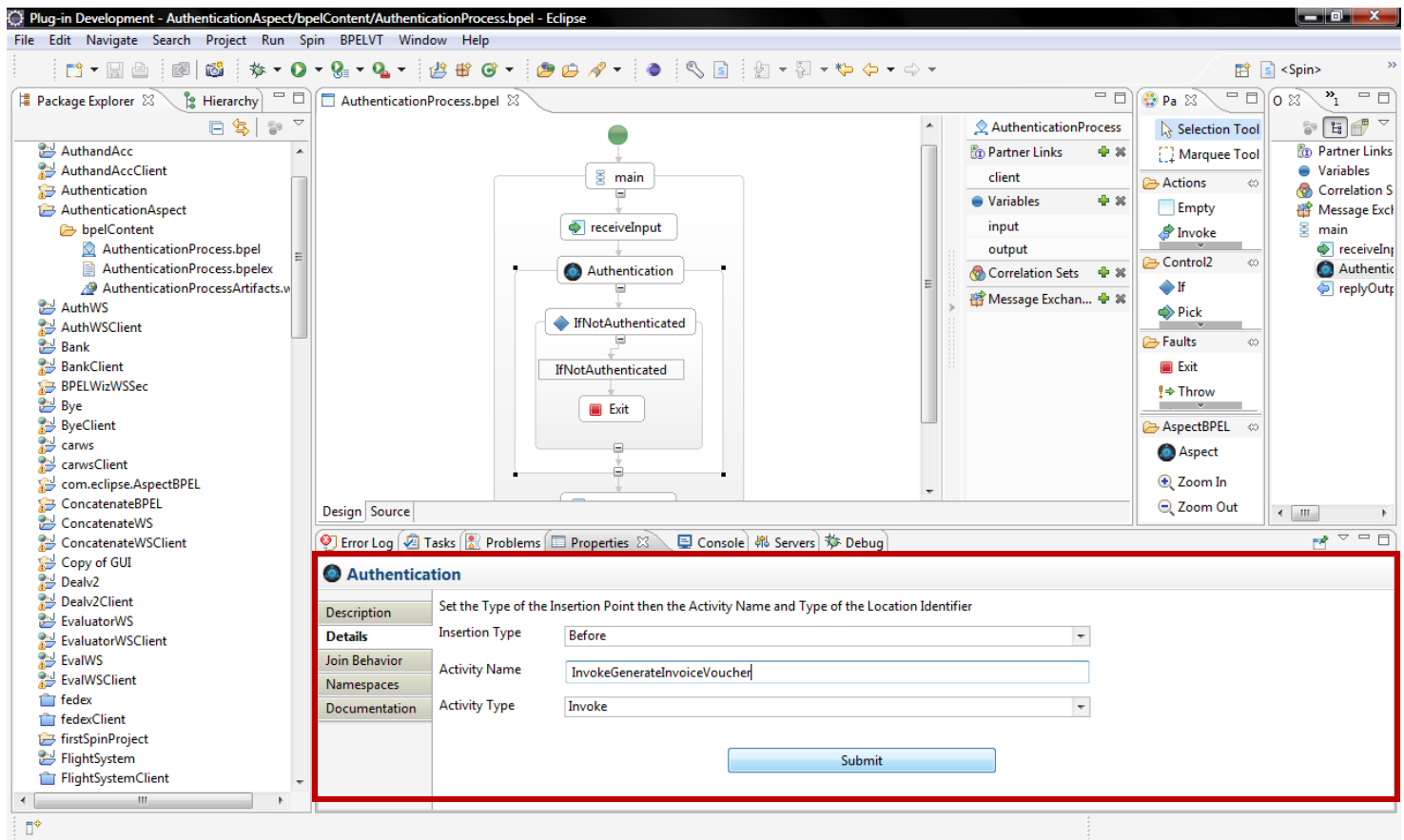
The next step is to specify the properties of the AspectBPEL aspect.

Right click on the aspect and select “show in properties” menu item. In the “Description” tab, you can set the name of your aspect (In this example, it is called Authentication).

In the “Details” tab, you have to specify:

1. The insertion type representing *BPEL_Insertion_Point* in AspectBPEL that could be Before, After or Replace.
2. The activity name and the activity type (any of the BPEL activities) to generate the *BPEL_Location_Identifier*.

An example is shown in the following figure.



Once the Submit button is clicked, the code will be generated (check the source of the process).

```
<bpel:extensionActivity>
  <aspectbpe:aspect name="Authentication" type="Before" activityName="InvokeGenerateInvoiceVoucher" activityType="Invoke">
    <bpel:if name="IfNotAuthenticated">
      <bpel:condition>[CDATA[!($input.payload!="hanine")]]</bpel:condition>
      <bpel:exit name="Exit"></bpel:exit>
    </bpel:if>
  </aspectbpe:aspect>
</bpel:extensionActivity>
```

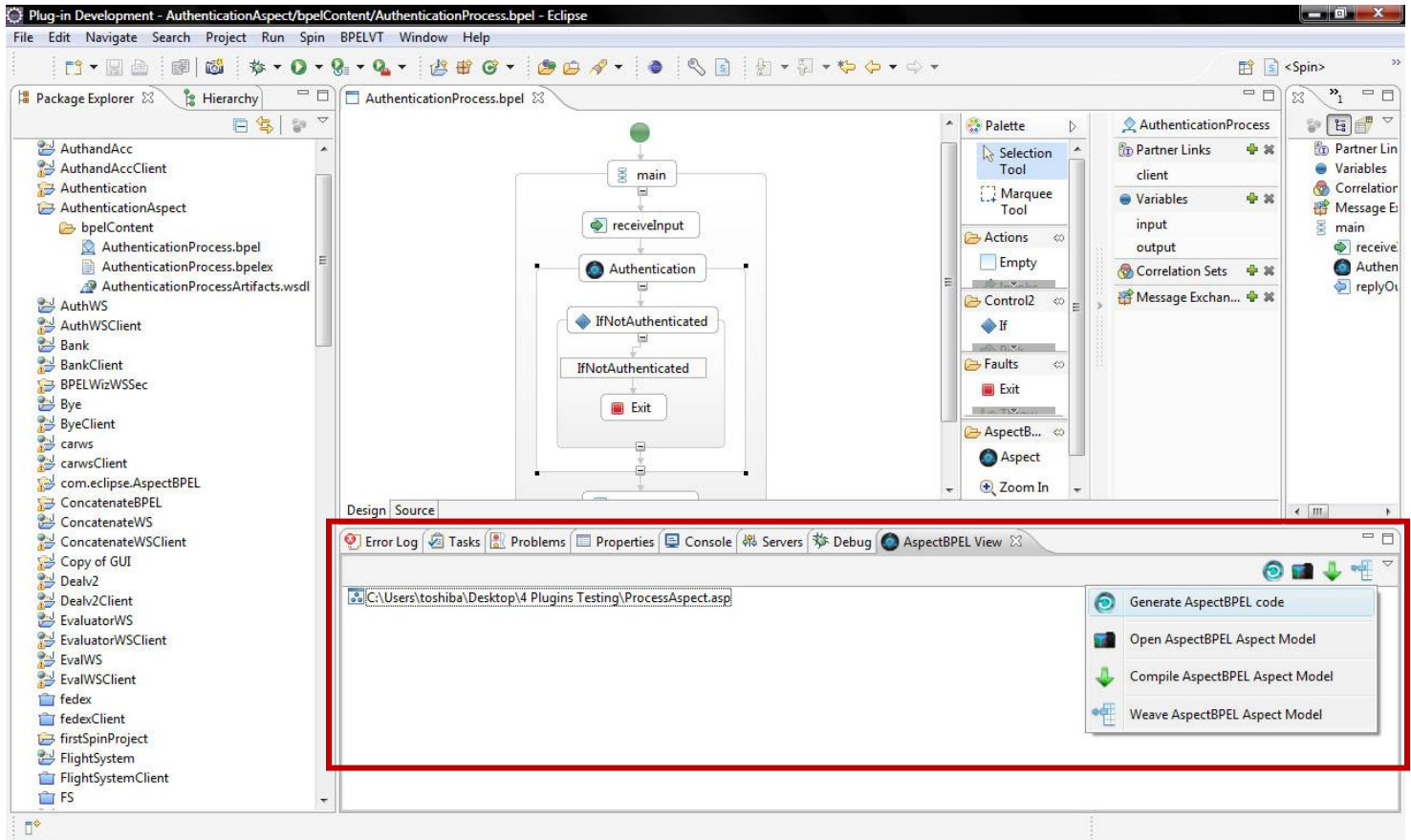
Finally, save your process (.bpel)

To complete the rest of the steps, go to Windows → Show View → Other and select AspectBPEL View under AspectBPEL category in the popup window.

2. Generate/Open AspectBPEL Aspect

2.1. Generate AspectBPEL Aspect Code

Select “Generate AspectBPEL Code” from the menu, choose the process just created and select the path where the generated code will be saved. An .asp file will be generated and will appear in the AspectBPEL view as shown below:



2.2. Open AspectBPEL Aspect

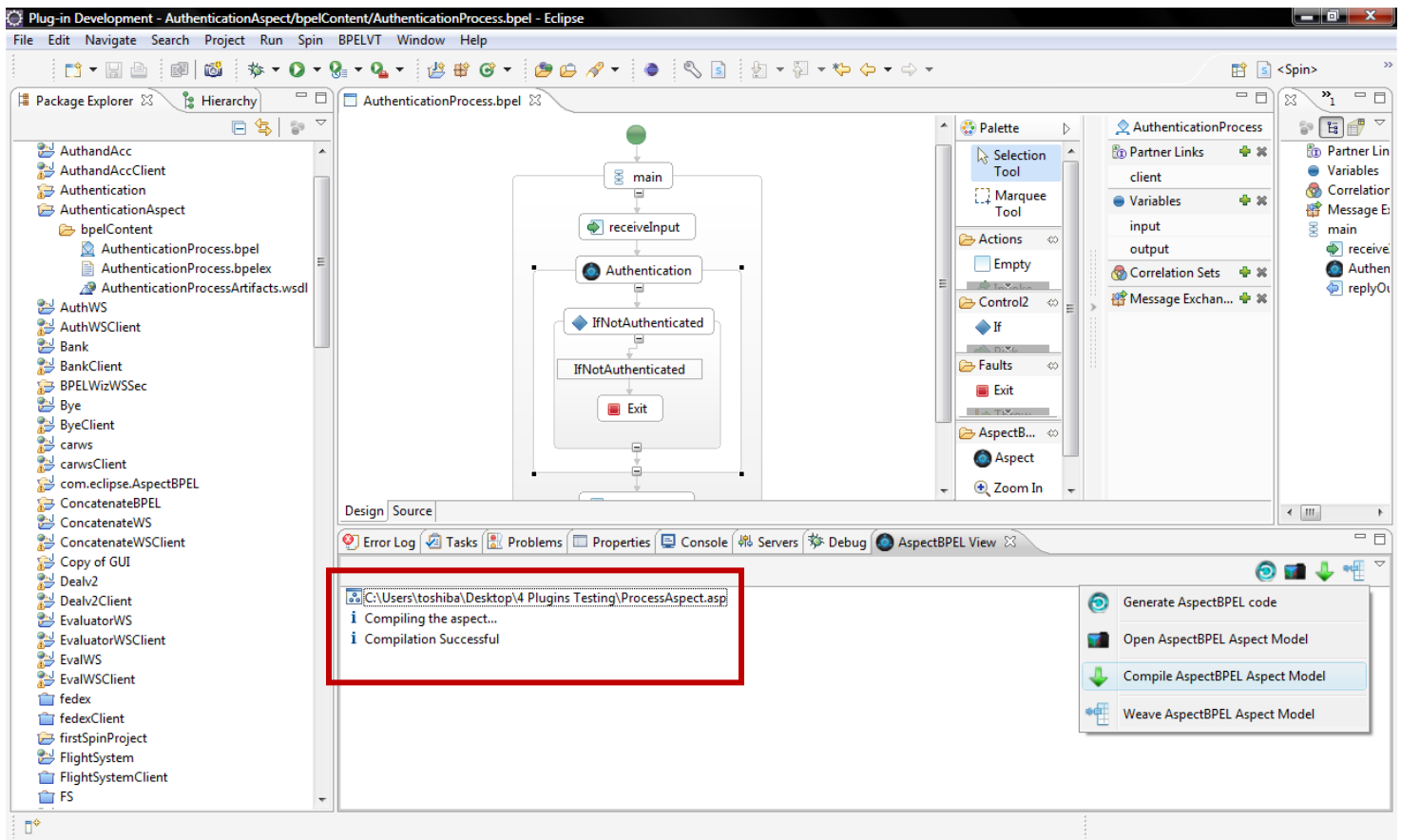
Instead of modeling the Aspect and generating its corresponding code, you may open an existing AspectBPEL file (.asp) by simply selecting the “Open AspectBPEL Aspect Model” from the menu.



The figure above shows an example of an AspectBPEL Aspect.asp file. This aspect is called “Authentication”. If the user is not authenticated, the BPEL process will exit (based on the BPEL_Behavior_Code in the last highlighted part). The behavior code will be weaved “After” the “InvokeGenerateInvoiceVoucher” activity that will be mapped to a BPEL process activity.

3. Compile AspectBPEL Aspect Model

To compile the aspect, select “Compile AspectBPEL Aspect Model” menu item.



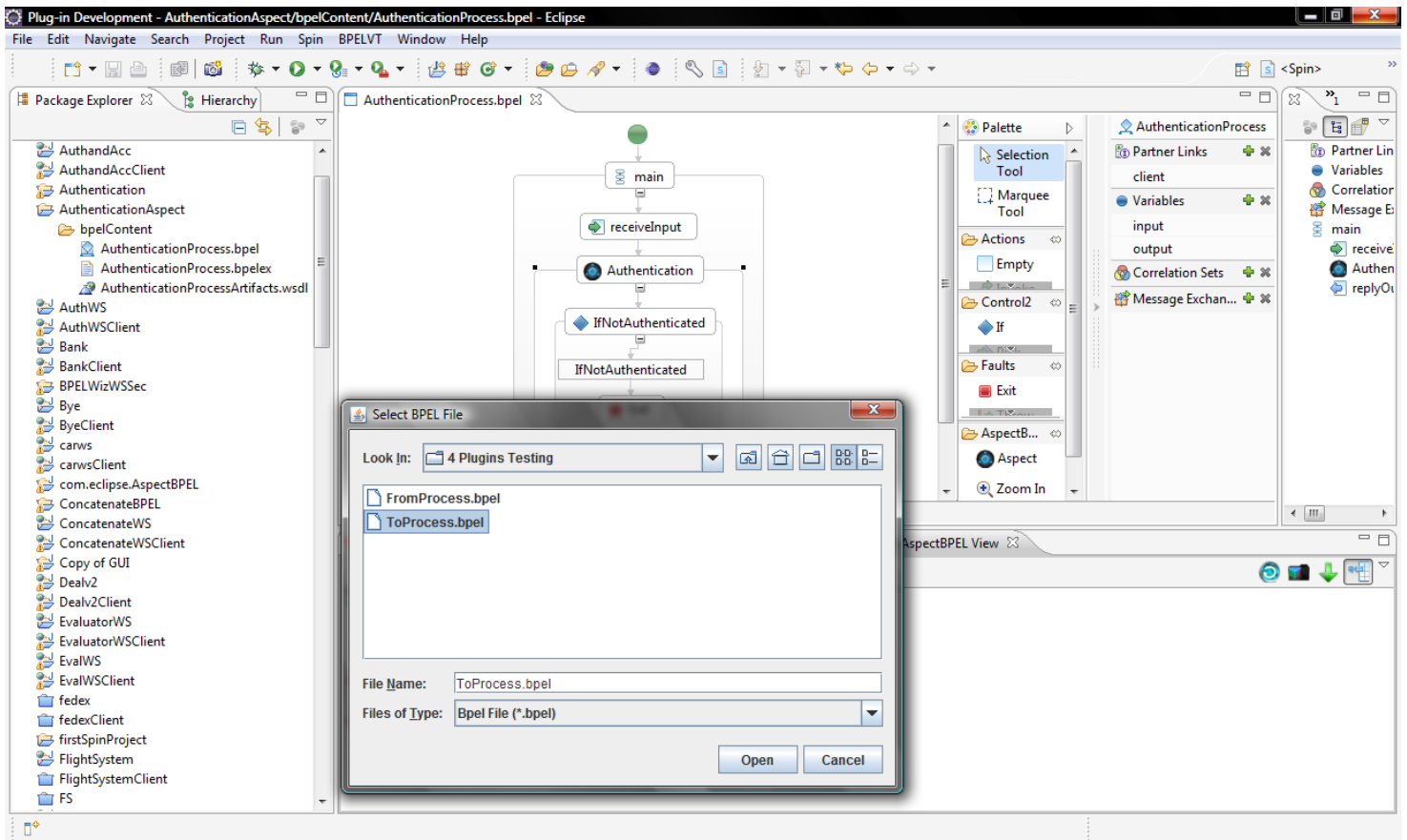
If the compilation was successful, “Compilation Successful” message will appear in the view. And you will be able to move to the last step (weaving).

Otherwise, an error message will show up like below:

i Compiling the aspect...
x Run Aspect Error: Cannot proceed without a functioning aspect



4. Weave AspectBPEL Aspect Model

To weave the aspect, select “Weave AspectBPEL Aspect Model” menu item. A file chooser will appear in order to select the BPEL process in which you want to integrate the aspect you have created.



The weaving may take few seconds or may freeze the eclipse for few seconds because the code will be verified against the BPEL schema definition over the internet to insure its correctness before weaving.

Once the weaving is completed, an appropriate message will appear in the view.

-  Searching for the specified pattern...
-  Weaving Completed

If any error occurs, an error message will show up.

Now the aspect is weaved into the BPEL process. Check your process file code.