Spagic Studio 3
User Guide

Author Andrea Zoppello
Gianfranco Boccalon

Document Goal

In this document we will focus on the feature of Spagic Studio 2 IDE to deliver SOA solutions based on Spagic solution.

Version History

Version/Release n° :	1.0	**Date**	January 28, 2008
Description/Modifications:	First Release (English version)		
Version/Release n° :	2.0	**Date**	May 10, 2010
Description/Modifications:	Updates for Spagic 3		

1 Spagic Studio

Spagic Studio IDE is composed as a set of tools and graphical editors, based on the eclipse platform. In particular within the Spagic studio distribution you'll find:

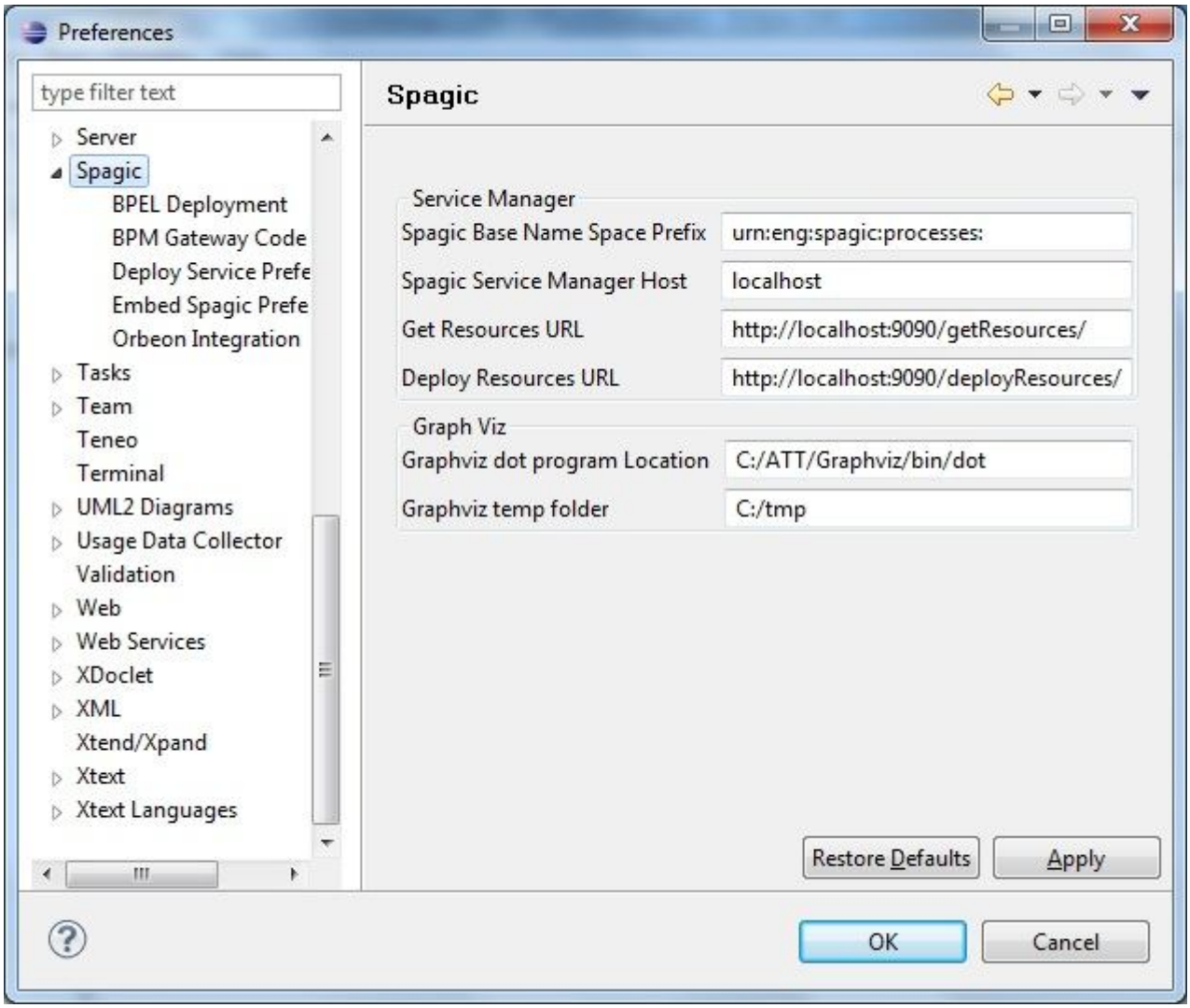
- A Graphical BPMN Editor (based on standard eclipse stp BPMN editor)
- An EMF Tree based editor for Intermediate Model files.
- A set of extensions to add to BPMN files, technology and Spagic detailed information on BPMN files.
  In particular you'll find tools for:
  - Setting preferences using a Spagic Preference Page
  - Creating a Spagic Project
  - Choosing a particular runtime technology for a Pool
  - Drag And Drop technology specific service/service bindings from a runtime view to BPMN Flow elements
  - Fill technologies detailed properties, for each BPMN task after you've annotated this with a particular service, service bindings couple
  - Manage Datasources and deploy it to service manager
  - Generate and Deploy artifacts for a particular runtime, starting from Intermediate Model

In the next sections we're going to explain in detail all these features.

2 Spagic Preference Page

The first time you open Spagic Studio you have to configure preferences page.

The Spagic Preferences Page is integrated in Eclipse preferences dialog (Window\Preferences) as shown in the following image:



- **Service Manager**
  - **Spagic Base Namespace Prefix:** Used to set the base namespace prefix using during code generation
  - **Spagic Service Manager Host:** the host in which runs the service manager
  - **Get Resources URL** : the path of the resources to be picked up.
  - **Deploy Resource URL**: the path in which resources will be deployed

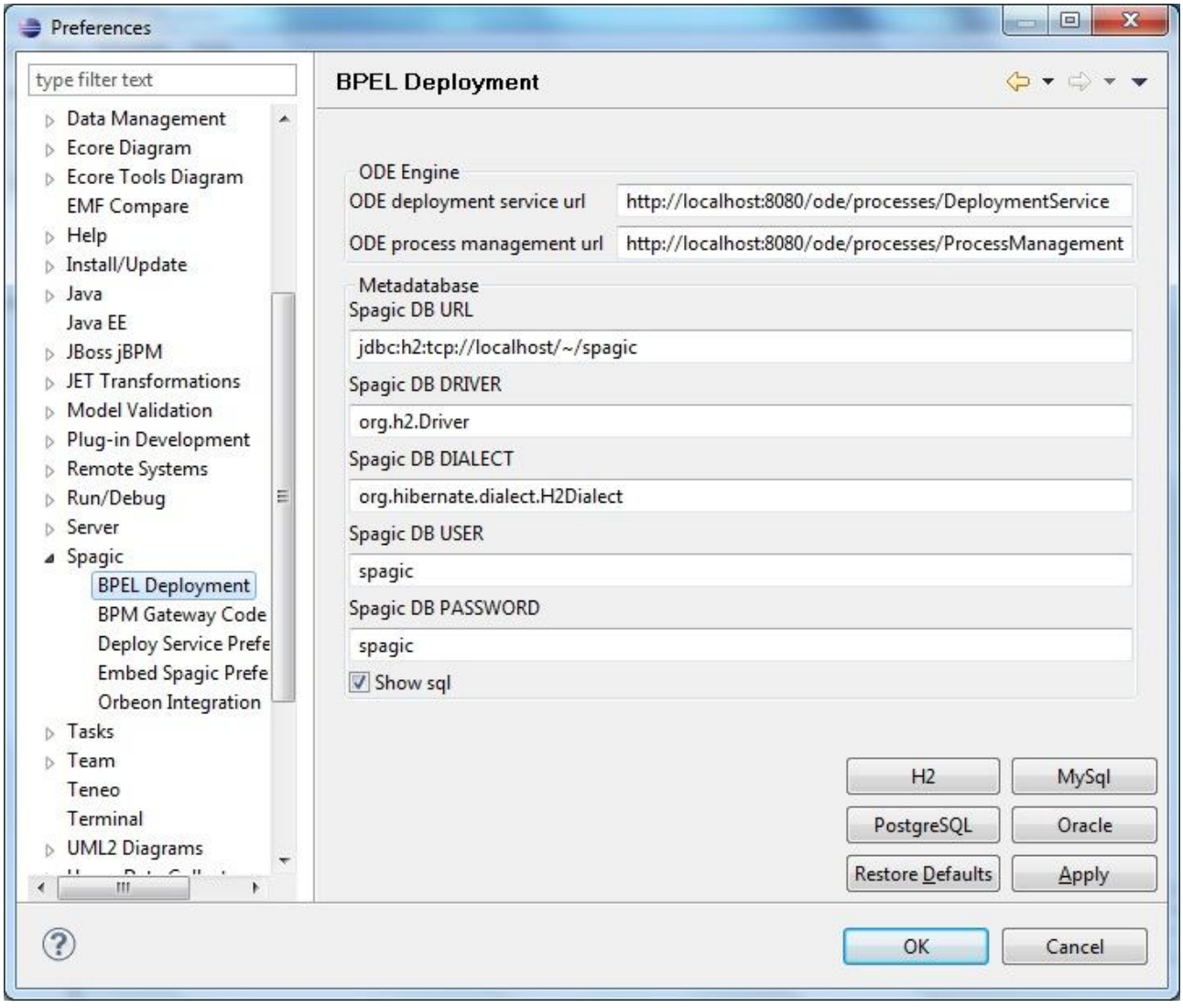
- **GraphViz**
  - **Graphviz Dot Program Location:** The path to graphviz dot program
  - **Graphviz Temp folder:** The path for graphviz temporary folder (Must be created manually)

Moreover, Spagic General Preferences is divided in 5 pages:

- **BPEL Deployment**
- **BPM Gateway Code Generation**
- **Deploy Service Preferences**
- **Embed Spagic Preferences**
- **Orbeon Integration**

2.1 BPEL Deployment

In this page you can set some parameters relate to BPEL deployment:



- **ODE Engine**

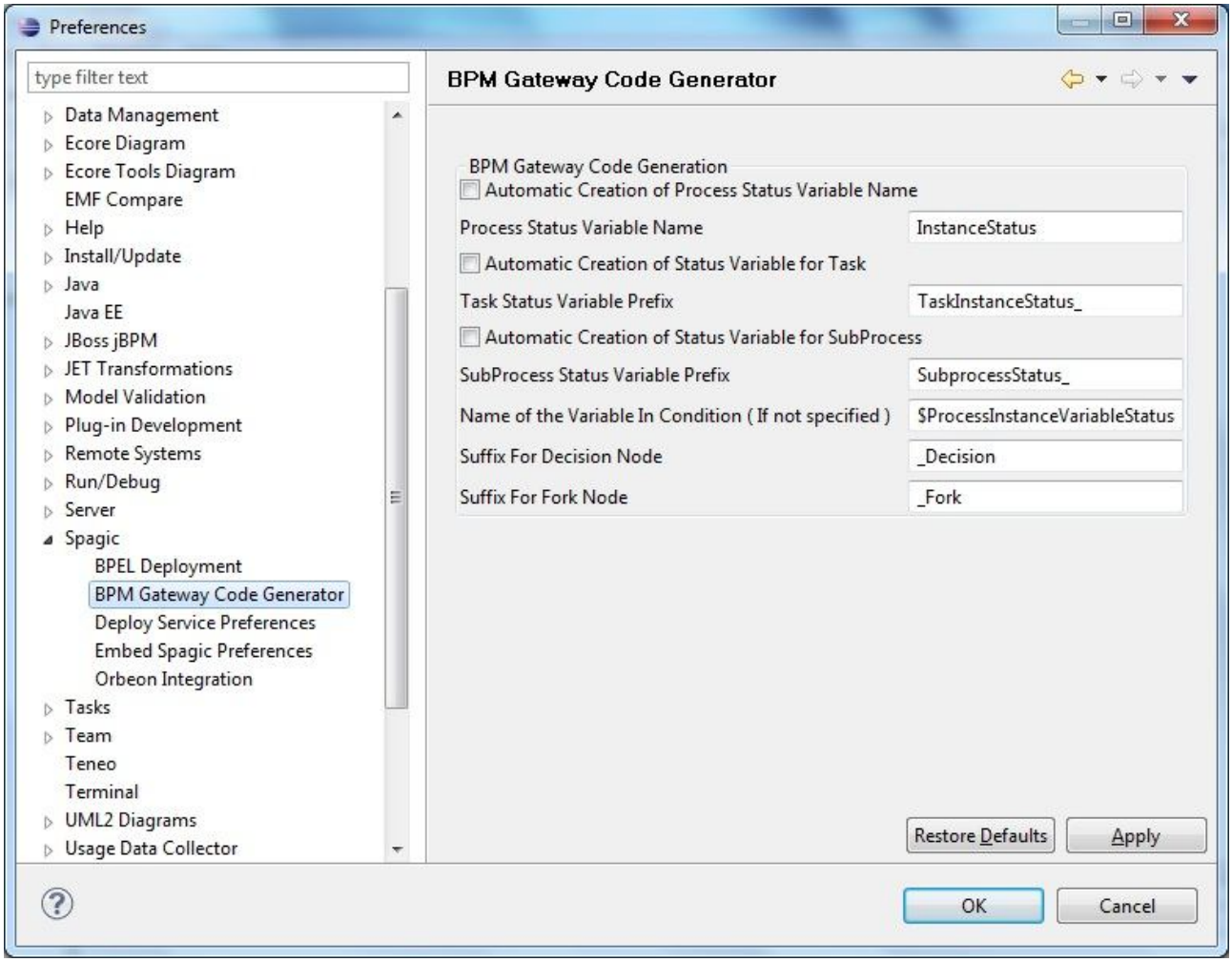
  - **ODE deployment service url:** the url exposed by ode for deployments.
  - **ODE process management url:** the url exposed by ode to get some informations about the engine status

- **Metadatabase (The metadatabase in Spagic 3 is required ONLY FOR BPEL DEPLOYMENT)**

  - **Spagic DB URL**: the jdbc url of the database created in the previous section
  - **Spagic DB Driver**: the full jdbc driver class name
  - **Spagic DB User**: username for accessing the audit DB.
  - **Spagic DB Password**: password for accessing the audit DB.
  - **Spagic DB Dialect**: the name of hibernate class for the database being used (org.hibernate.dialect.MySQLInnoDBDialect for MySQL)
  - **Show SQL**: Check this only for debug purpose

2.2 BPM Gateway CodeGeneration

In this page you can set some parameters to control some features of the BPM Gateway code generator:



- **Automatic Creation of Process Status Variable Name:** Checking this option means you want that for each new process instance you want to create automatically an instance status variable.

- **Process Status Variable Name:** Useful only if you've check "Automatic Creation of Process Status Variable Name", this will let the user to choose the name of the instance status variable**.**

- **Automatic Creation of Status Variable of Task:** Checking this option means you want that for each task you want to create automatically a *task instance status variable*.

- **Task Status Variable Prefix:** Useful only if you've check "Automatic Creation of Status Variable of Task", this will let the user to choose the prefix of the task instance status variable. This is a little different from process instance status variable name, because tasks are multiple so you choose the "prefix" and the task instances variable name will be something like "TaskInstanceStatus_Task1".

- **Automatic Creation of Status Variable for SubProcess:** Checking this option means you want that for each subprocess task you want to create automatically a *subprocess instance status variable*.

- **Subprocess Status Variable Prefix:** Useful only if you've check "Automatic Creation of Status Variable for Subprocess", this will let the user to choose the prefix of the subprocess instance status variable. This is a little different from process instance status variable name, because subprocess task are multiple so you choose the "prefix" and the subprocess instances variable name will be something like "SubprocessStatus_Sub1".

- **Name of the Variable in Condition: THIS PARAMETER IS DEPRECATED.** In in the BPMN Diagram you put a label on the transition outcoming from a decision node (exclusive gateway) and **you've not used an explicit annotation**, the code generator could generate an expression in the form of:

  - **#{<NAME_OF_THE_VARIABLE_IN_CONDITION> = <LABEL_IN_THE_TRANSITION_ON_DIAGRAM>}**

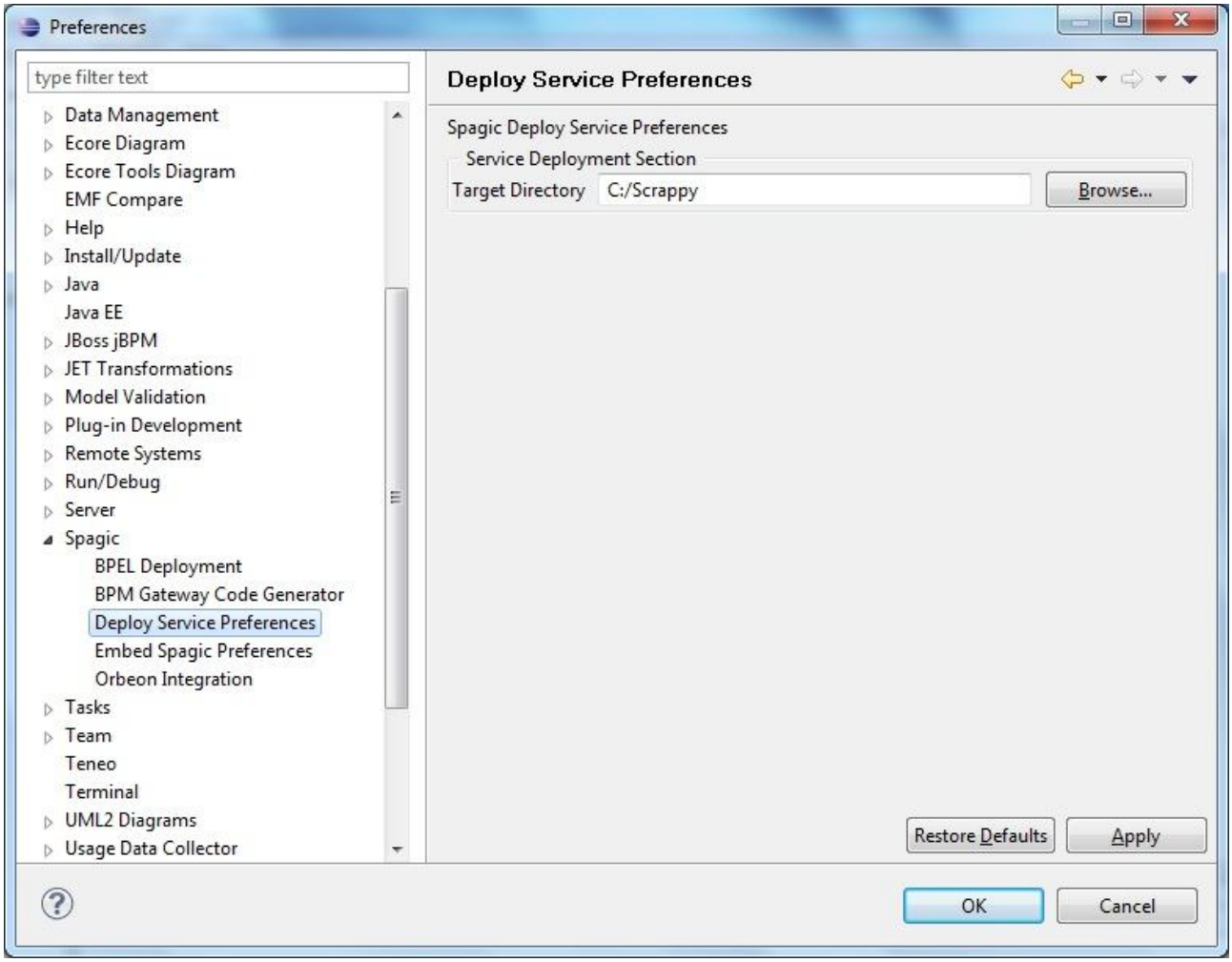
  IMPORTANT: This parameter is provided for back compatibility support with SPAGIC-2 Workflow processes BUT WE DISCOURAGE TO USE THIS.
  THE RIGHT WAY TO MAKE CONDITION ON TRANSITION IS TO USE ANNOTATIONS.

- **Suffix for Decision Node:** During the code Generation a suffix is generated for decision node, with this paramenter you could chose this suffix.

- **Suffix for Fork Node:** During the code Generation a suffix is generated for parallel/fork node, with this paramenter you could chose this suffix.

2.3 Deploy Service Preferences

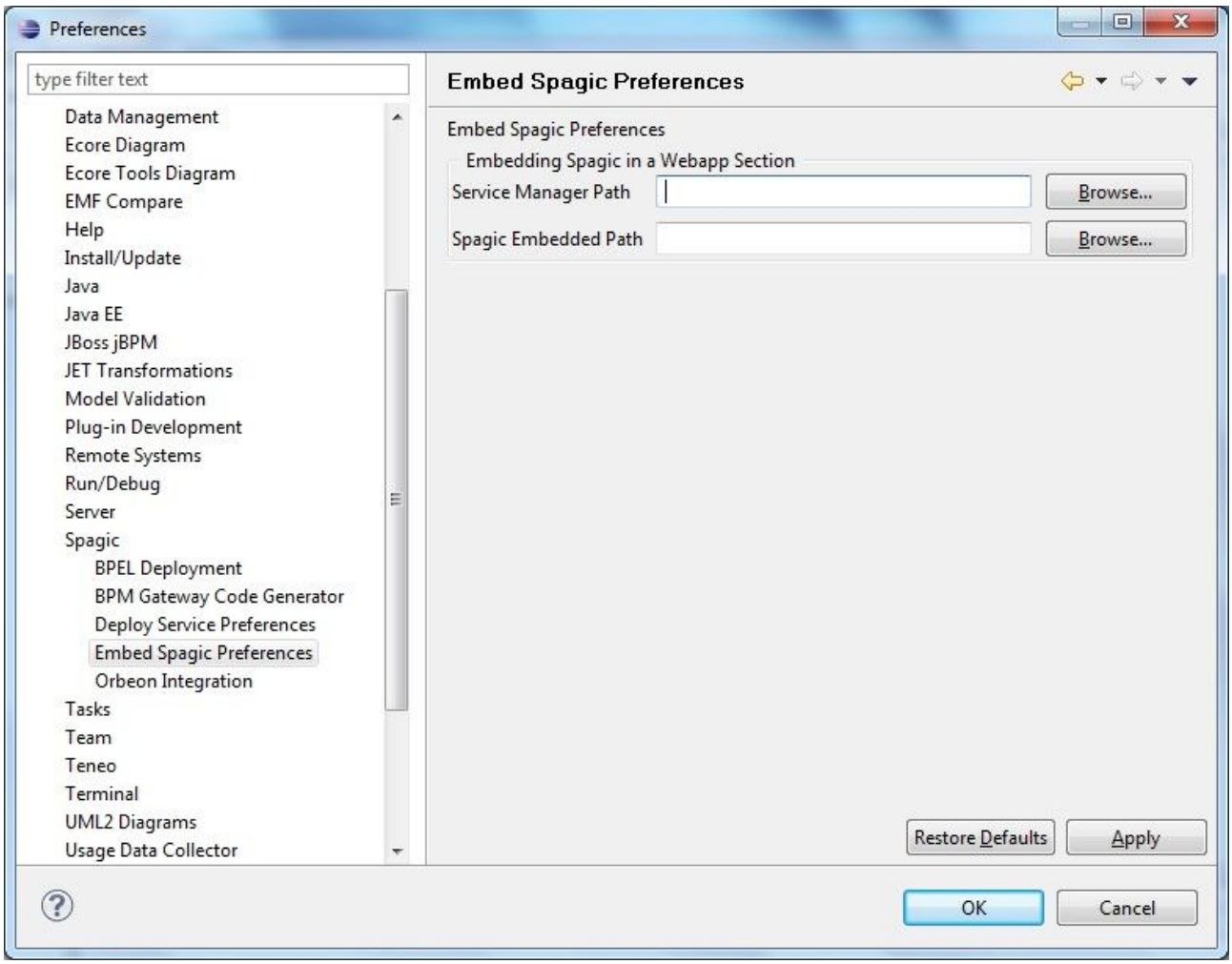
In this page you can set some parameters relate to service deploying :



- **Target Directory: The path to a** SPAGIC INSTANCE HOME (the path of the directory where the services will be placed during the deployment through the deployment serviced). Further clarification on SPAGIC INSTANCE are give in document **"Spagic3 Service Manager Concepts and Guide".**

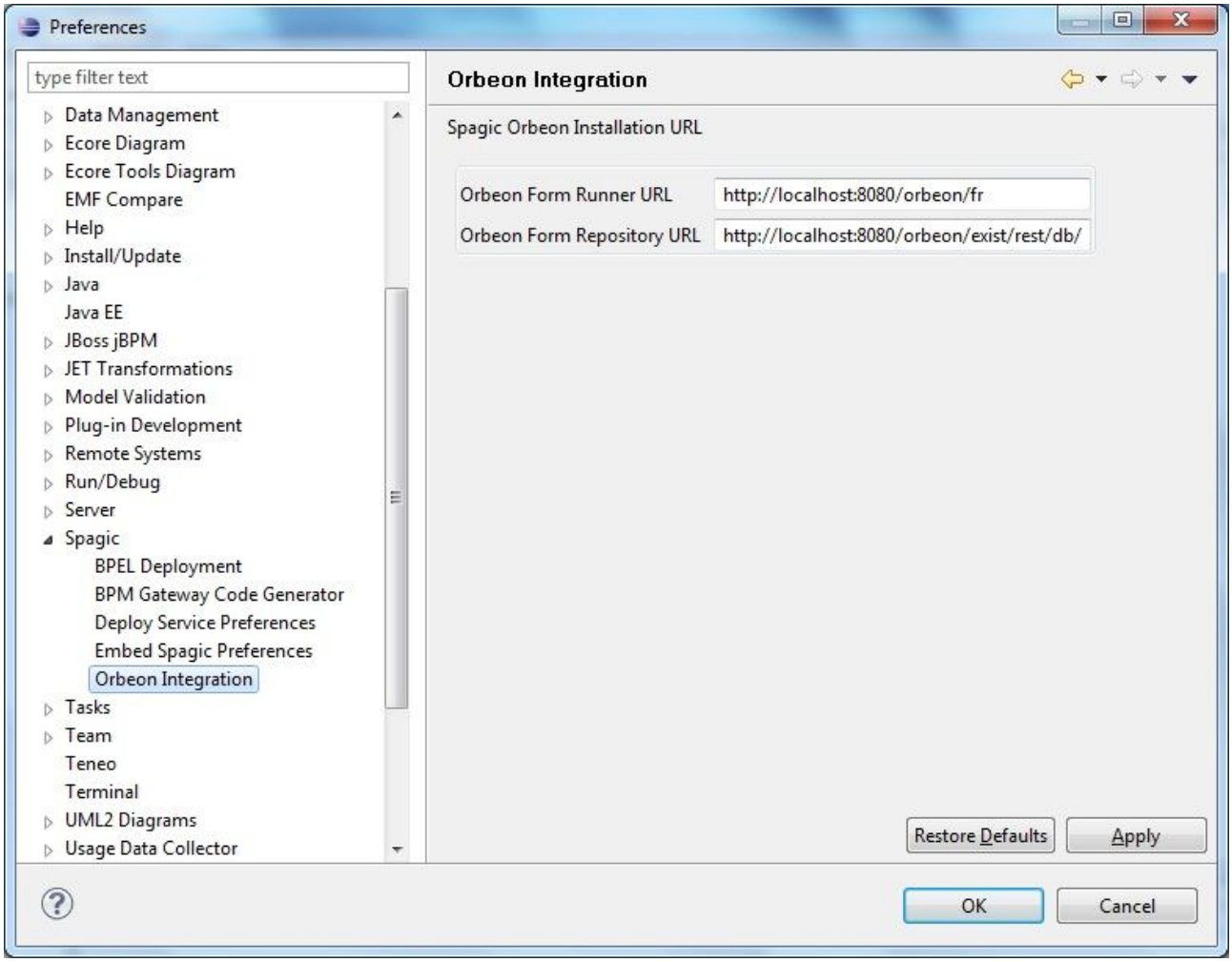
2.4 Embed Spagic Preference

With spagic studio you could easy embed the service manager in your webapp. To do that we need some resources distributed to the spagic distribution.



2.5 Orbeon Integration

In this page you can set some parameters relate to the Orbeon Integration:



- **Orbeon Form Runner URL**: The URL of Orbeon form runner application
- **Orbeon Form Repository URL:** The URL of Orbeon form runner repository

3 Datasources

Some components that can be used during the modeling of an integration process are related to database activity and they need to be configured with datasources to work correctly.

Typically the datasources are defined in the ESB by its own configuration file.

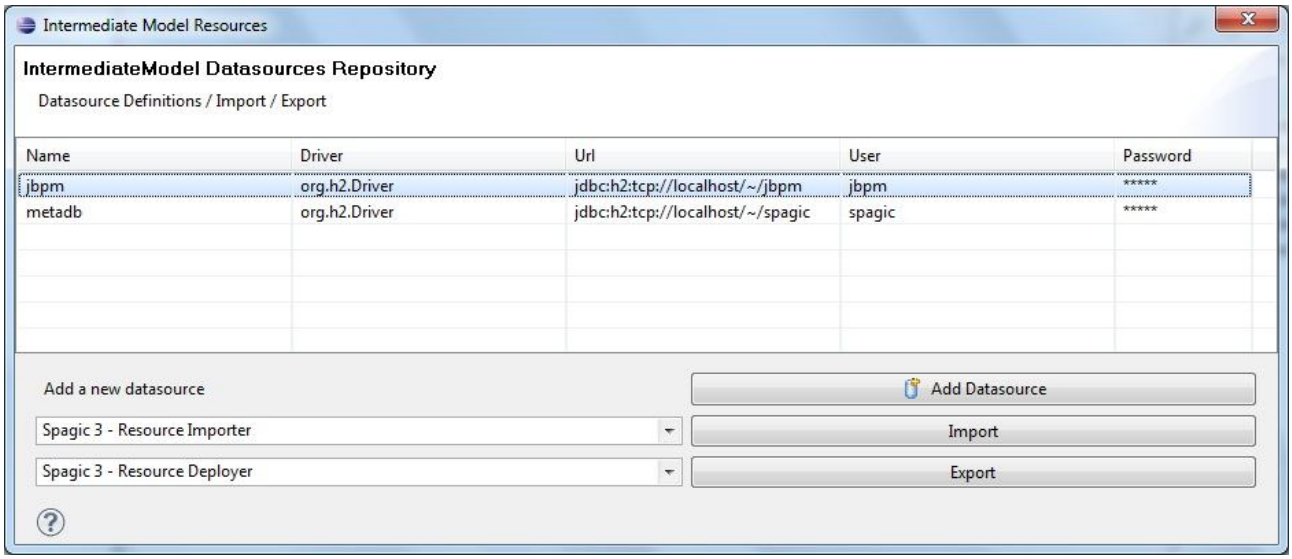
To avoid "missing datasource problems" during the deployment phase we need a way to provide bidirectional synchronization between datasources defined in Spagic Studio and datasources defined in the ESB runtimes.

Spagic Studio provides a local datasource repository where it's possible:

- Import datasources from Spagic Service Manager
- Export datasources to Spagic Service Manager

In Spagic Studio a database configuration utilities is provided in the toolbar by the following button:

If you click on it a dialog like this will open:



Here you can see all the datasources defined in Spagic Studio environment. Here you can:

1. Add a new datasource to Spagic Studio with the "*Add Datasource"* button. Pay attention that this feature will add a datasource to Spagic Studio environment not to Service Manager.

2. Import datasources definition using specific *Importer*; Spagic Studio defines the standard Importer:

   - ***Spagic3 Resource Importer***, try to get datasources list defined in a running Service Manager

3. Export Datasources using a Resource Configurator, Spagic Studio defines a standard Spagic Resource Configurator:

   - ***Spagic Resource Deployer:*** This will generate .ds file and deploy it to a running Service manager.

If during import operations a conflict is detected (for example we can have different configuration in Spagic Studio and ServiceMix for the same datasource), Spagic Studio will ask you to choose one.

4 Working with Spagic Studio(Development and Deployment)

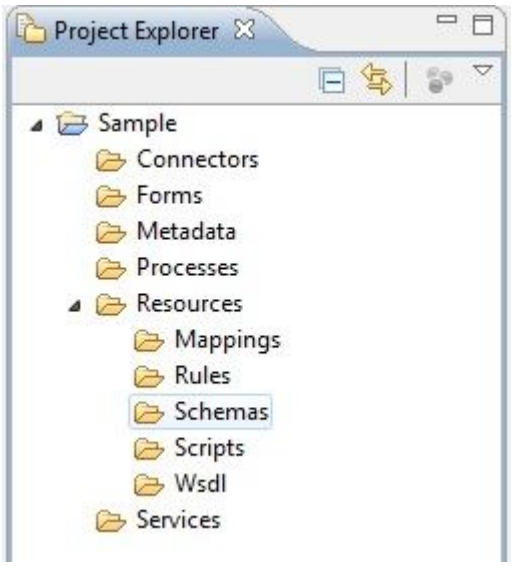
This document cover only the feature of Spagic Studio, not already explained in other documents. To start to work on Spagic, from development toward deployment refer to ***"Spagic3 – Getting Started Document" or to "BPEL Processes with Spagic3" (if you want to work with BPEL).***

5 Spagic Studio Development Features

As we already stated in previous paragraph the best way to work with Spagic is to follow the steps described in Spagic, "Spagic3 – Getting Started" document. On the other side there are some features of spagic that could be useful during deployment phase.

5.1 Understanding Spagic Project Structure.

A first important thing to know is to understand the structure of a Spagic Project in details. When you use the wizard to create a new project this will look like:



- **Connectors :** This folder will contain all the connectors linked to services or processes that we will configure in the project
- **Forms**
- **Metadata**
- **Processes** : This folder i contains all the file that deal with processes. In it there are the BPMN diagrams and the intermediate model of each process
- **Resources:** This folder contains all the resources used by the process and services in the project. The folder is divided in many subfolders, specifying different type of resources:

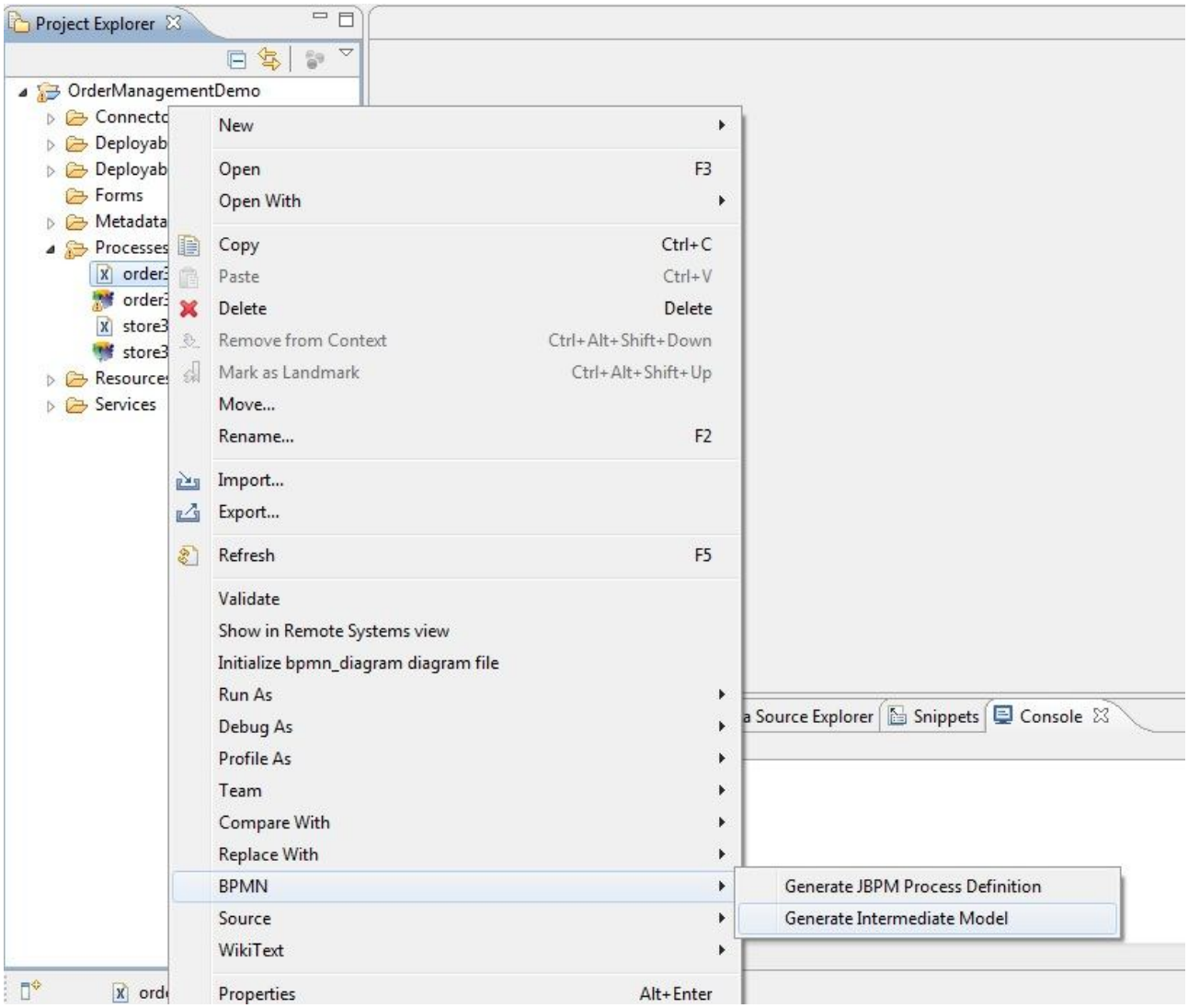
- **Mappings**: This folder contains resources that are used by mapping components. Most of this resource will be XSLT file.
- **Scripts:**This folder contains resources that are used by Scripting Components. Actually groovy is the language for the scripting, so this folder will contain groovy file.
- **SemanticRules**: This folder contains resources that are used by Semantic Validator Component. Rules are expressed in Drools 3.0 syntax.
- **SyntaxRules**: This folder contains resources that are used by the Syntax Validator Component. The validation of normalized messages is performed by xsd files.
- **Wsdl** This folder contains resources that are automatically generated by Spagic Studio if your process contains entry endpoint relative to HTTP Component configured to be a SOAP Provider. Automatic WSDL generation is provided by Spagic Studio for two important reasons:
  - Client applications of your service assembly needs WSDL to automatically generate clients stub (for example with axis)
  - Once you've generated a WSDL for a particular service assembly some type checking and restrictions can be made in the input of the application changing manually the WSDL generated. If you change the WSDL manually the modified version will be deployed in service assembly structure.

- **Services:** in this folder contains all the services of the project.

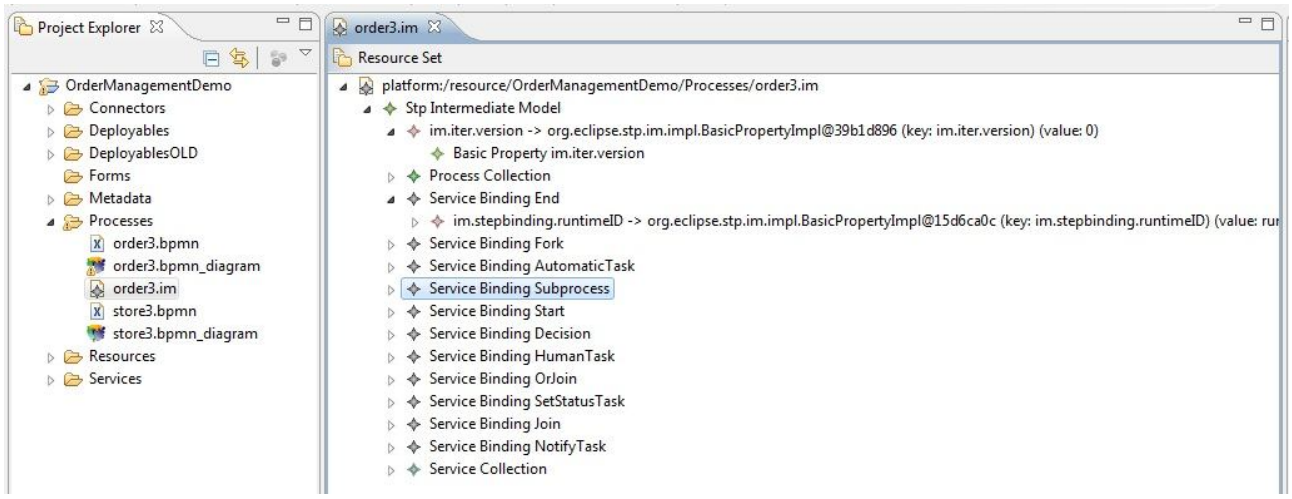
5.2 Generate Intermediate Model from BPMN

To get the Intermediate Model from the BPMN File.
To do that:

- Select a bpmn file
- Open the context menu with a right click and select the "BPMN" submenu
- Select the Generate Intermediate Model



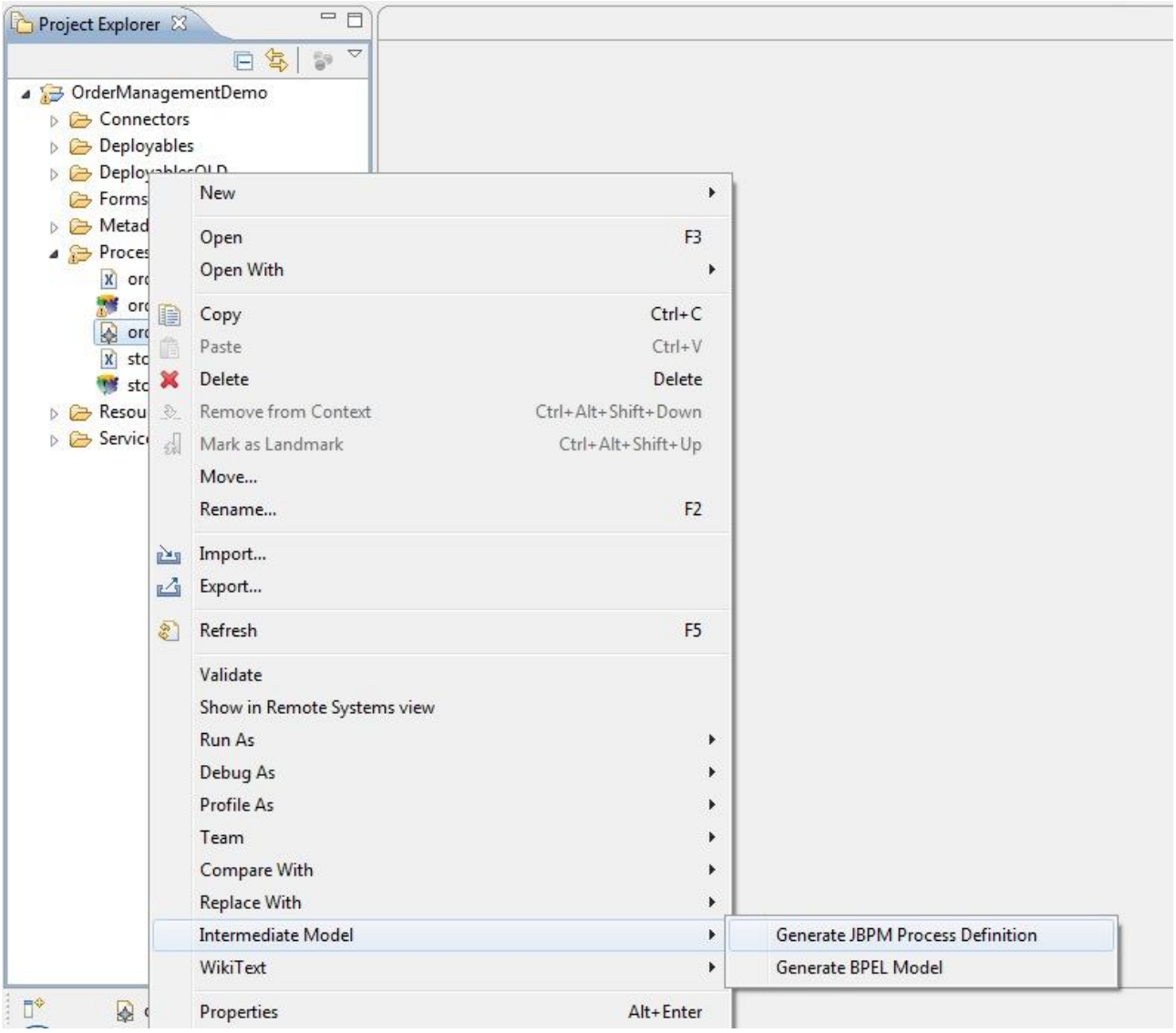
So you could look at the IM:



5.3 Generating JBPM from Intermediate Model

To get the JPDL file from a IM file without deploying it, it's possible to:

- Select the im file
- Open the context menu with a right click and select the "Intermediate Model" submenu
- Select the Generate JBPM Process Definition

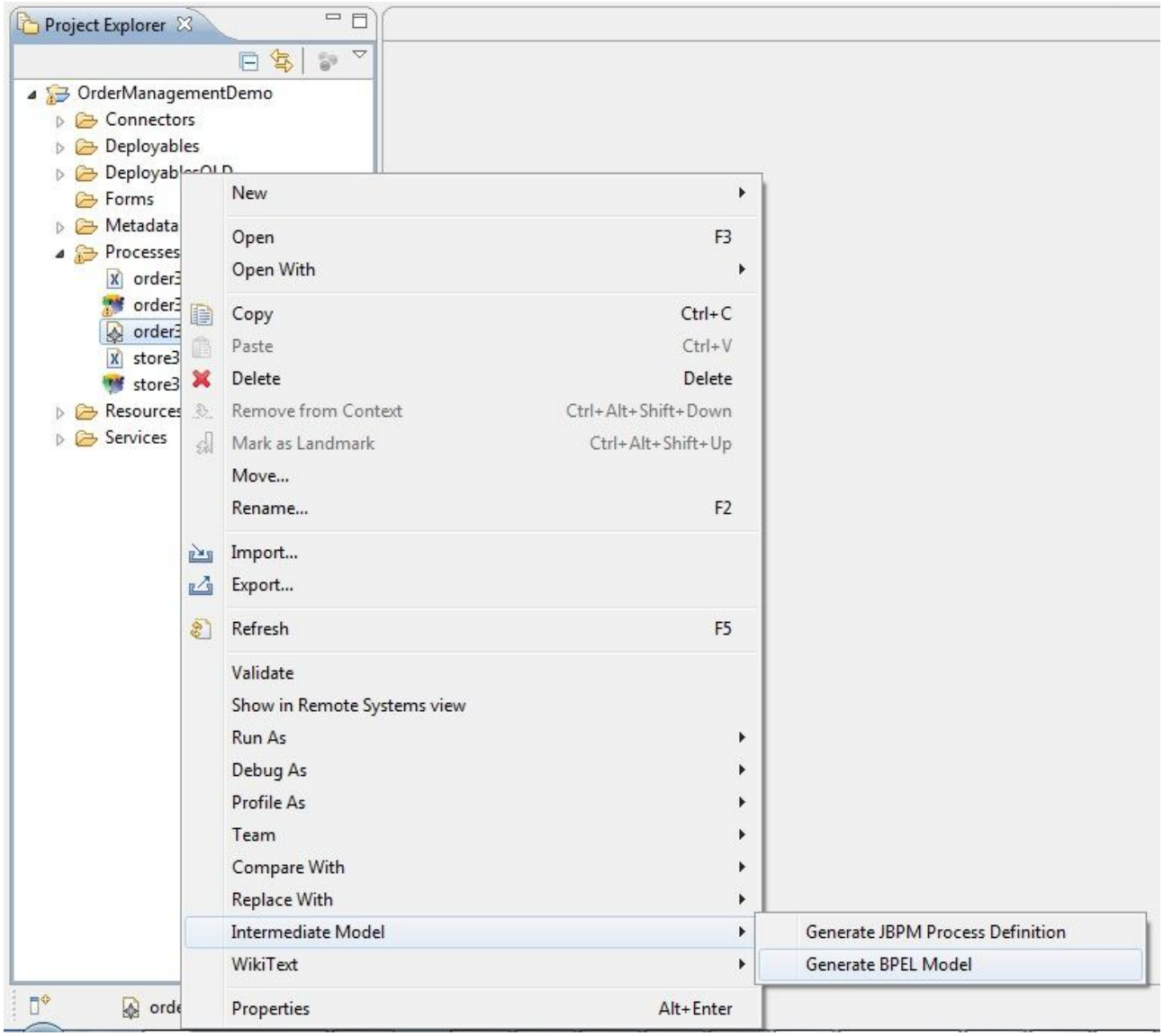


5.4 Generating BPEL From Intermediate Model

To get a BPEL file from a IM file without deploying it, it's possible to:

- Select the im file
- Open the context menu with a right click and select the "Intermediate Model" submenu

- Select the Generate BPEL Model



5.5 Generating JBPM From BPMN

To get the JPDL file from a BPMN file without deploying it, it's possible to:

- Select a bpmn file
- Open the context menu with a right click and select the "BPMN" submenu
- Select the Generate JBPM Process Definition