Spagic Studio 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**	July 29, 2008
Description/Modifications:	Updates for Spagic 2.2.0		

1 Spagic Studio

1.1 Spagic Studio Features

Spagic Studio v2 IDE is a 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
  - Versioning
  - Manage the Observable Attributes
  - Define and Configure Iters
  - Manage Datasources and Namespace
  - Publish the intermediate Model on Database
  - Generate and Deploy processes for a particular runtime, starting from Intermediate Model
  - Managing the processes publication in a UDDI Registries

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

1.2 Observable Attributes – Iter Overview

ESB solutions are very good solutions to define integration scenarios processes between applications and services. So they're perfect if we look at them from a technology point of view. The problem is that in most cases target users of software solutions are interested to business process management scenario.
In other terms they want to have a view in terms of:

1. ***Business Process*** (or logical use case)
   A business process simply describes a scenario as a set of a logical tasks:
   a. Business processes are not necessary related to technology concepts.
   b. Some of these tasks could be described with technologic details, but this is not mandatory.
   c. A particular business process instances is identified by a set of relevant data.

2. ***Integration Processes***
   An integration process describes flows and interaction between technology related services.

3. ***Observable Attributes (Relevant Data)***
   Set of data that will be extracted using rules during process execution.
   Relevant data are referred to both Business and Integration Process.
   Using relevant data we're able to correlate two or more integration process instances to a particular business process instance.

   In most of the cases, for a **particular business domain the defined observable attributes set, will be always the same,** for this reason in the new Spagic Studio observable attributes are defined in a central repository, and then associated to the processes, with simple drag and drop operation.

So there could be business processes without having anything related to technology. Obviously JBI and ESB are related to technology so in Spagic we're interested in Business Processes that has some tasks mapped directly to integration flow.

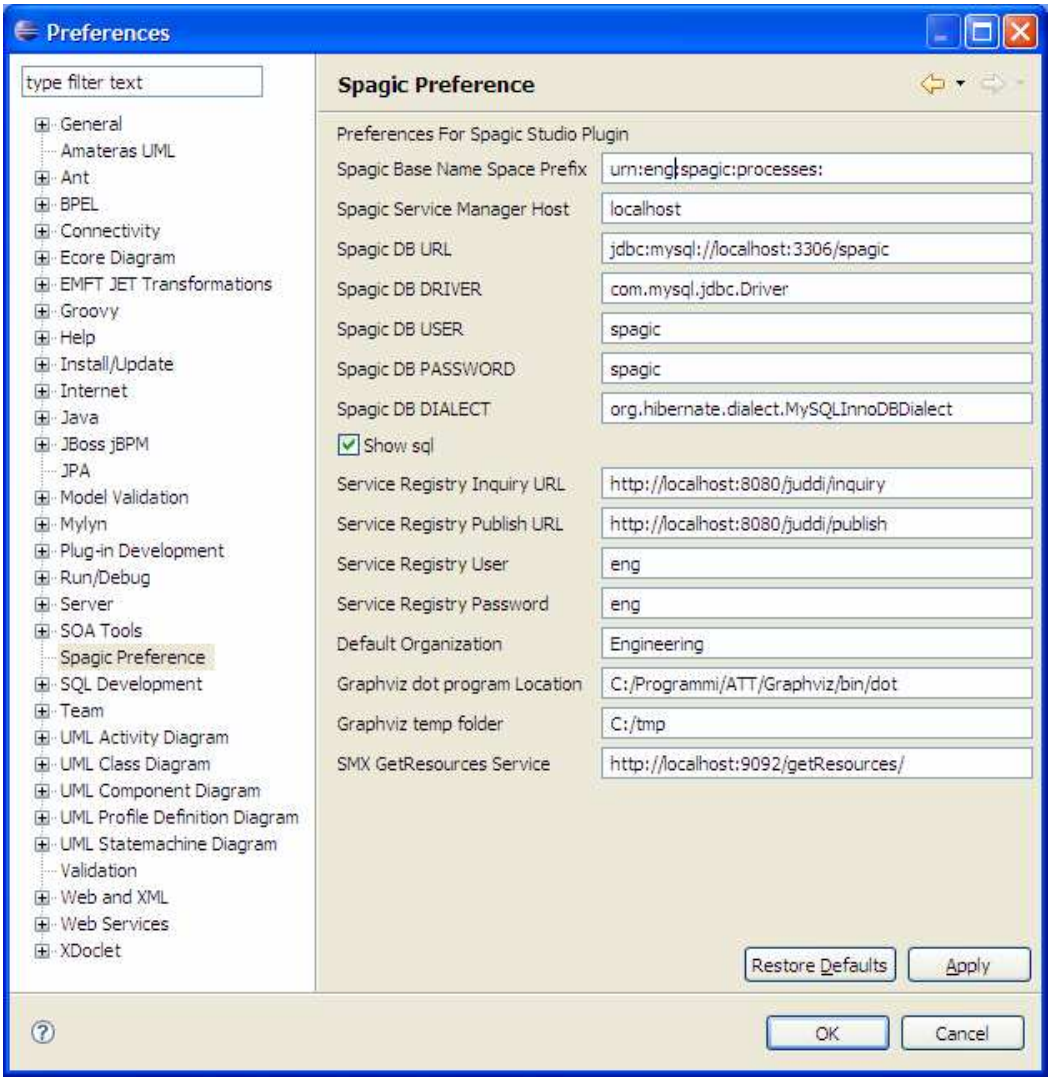
In Spagic we provide:

- A way to describe a business process as a set of integration processes.
  In Spagic a business process is called ***Iter***
- A way to defines ***observable attributes*** and associate this to process elements.
- A way to define the set of observable attributes that univocally identify a business process (iter) instance from the list of these associated to the processes belonging to the iter.

1.3 Spagic Preference Page

The first step to do when you open Spagic Studio is to configure preference page.

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



Some parameters relate to connection with Audit Database and service registry, and others are related to the connection with ServiceMix and graphviz.

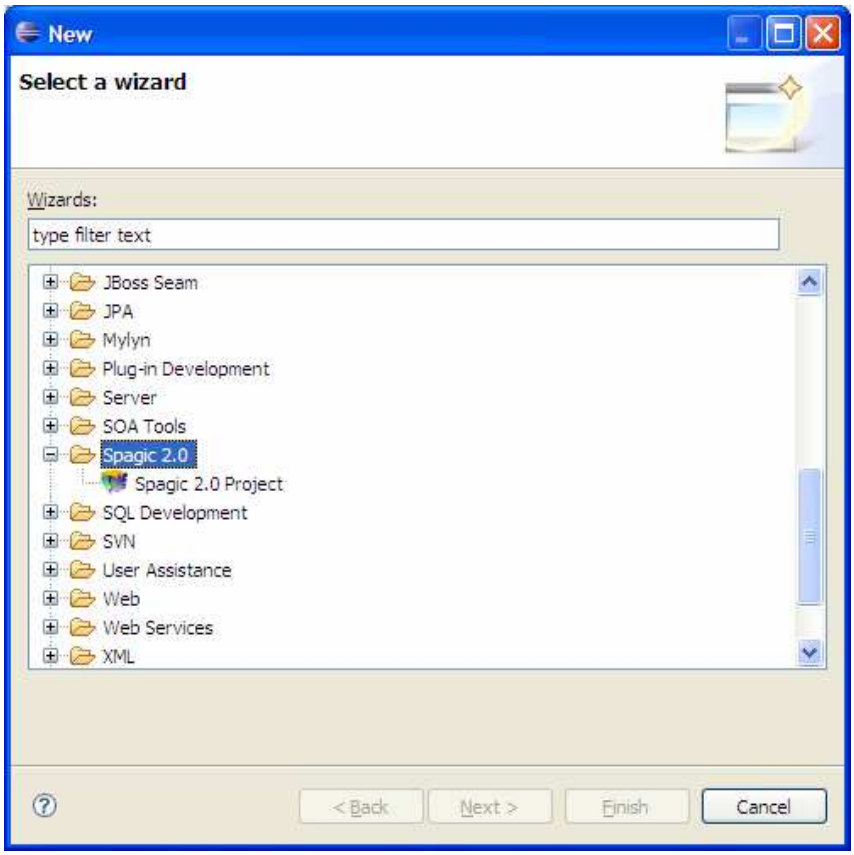
- **Spagic Base Namespace Prefix:** Let's to set the base namespace prefix using during code generation
- **Spagic DB URL**: the jdbc url of the database created in the previous section
- **Spagic DB Driver**: the full jdbc driver class name
- **Spagic DB User**: username for accessing the audit DB.
- **Spagic DB Password**: password for accessing the audit DB.
- **Spagic DB Dialect**: the name of the hibernate class for the database used (org.hibernate.dialect.MySQLInnoDBDialect for MySQL)
- **Show SQL**: Check this only for debug purpose

- **Service Registry Inquiry URL**: the url of inquiry service exposed by jUDDI
- **Service Registry Publish URL**: : the url of publish service exposed by jUDDI
- **Service Registry User**: the user that Spagic Studio use to connect to jUDDI
- **Service Registry Password**: the password that Spagic Studio use to connect to jUDDI
- **Default Organization:** the default organization where to publish the services
- **Graphviz Dot Program Location:** The path to graphviz dot program
- **Graphviz Temp folder:** The path that graphviz will use as temporary folder
- **SMX Get Resources Services:** The url of the services that Spagic Studio uses to get resources information from a running ServiceMix.

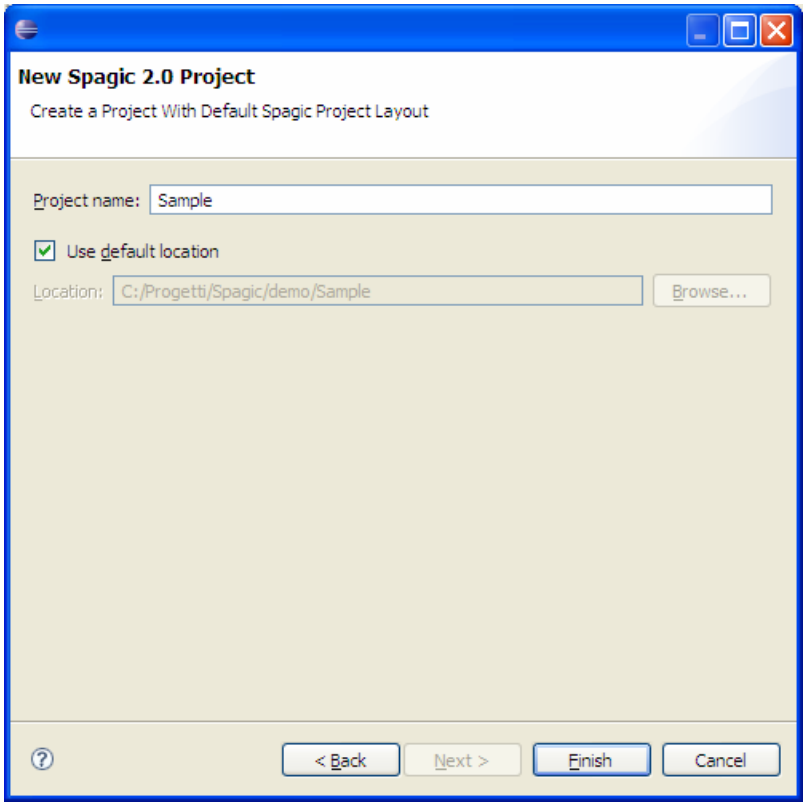
1.4 Creating a Spagic Project

In Spagic Studio the work is organized in Spagic Projects. The better way to understand how a project is organized is to create a new one and to see its structure.

To create a new Spagic Project the Spagic Studio plugin provide a specific wizard in ***File\New\Project*** eclipse menu. So if the tool is installed correctly the new ***Spagic Project Wizard*** should be located in the *Spagic* category:

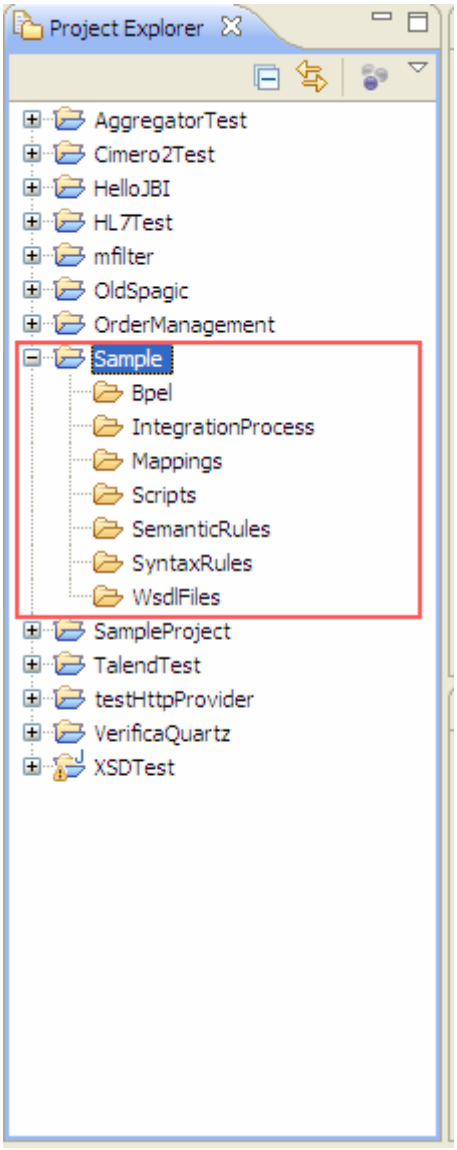


If you click the next button a dialog require to insert the project name and location:



Insert the name (for example “Sample”) of the project and click *Finish*.

In the workspace the Sample Project have just been created with the standard project structure:



As you can see from the image a Spagic Project is composed of the following folders:

- **Bpel:** This folder will contain bpel artifacts if we're going to generate Bpel code, from intermediate model.
- **Deployables**: contains the following files produced by Spagic Studio:
  - **<Process name>_v_<version>-sa.zip**: service assembly to be deployed in ServiceMix.
  - **<Process name>_v_<version>.deploy**: deploy file necessary to publish the process on the Spagic MetaDB, if you don't want to use Spagic Studio but a Spagic command line tool (necessary for example when migrating the processes from test environments to production environments).
  - **<Process name>_v_<version>.properties**: optional file created if within the processes are used some configurable parameters.
- **Integration Process**: This folder is the most important and contains all Spagic file describing the service assembly in terms of endpoints and flow between them.
- **Mappings**: This folder contains resources that are used by the mapping component. Almost of this resource will be XSLT file.

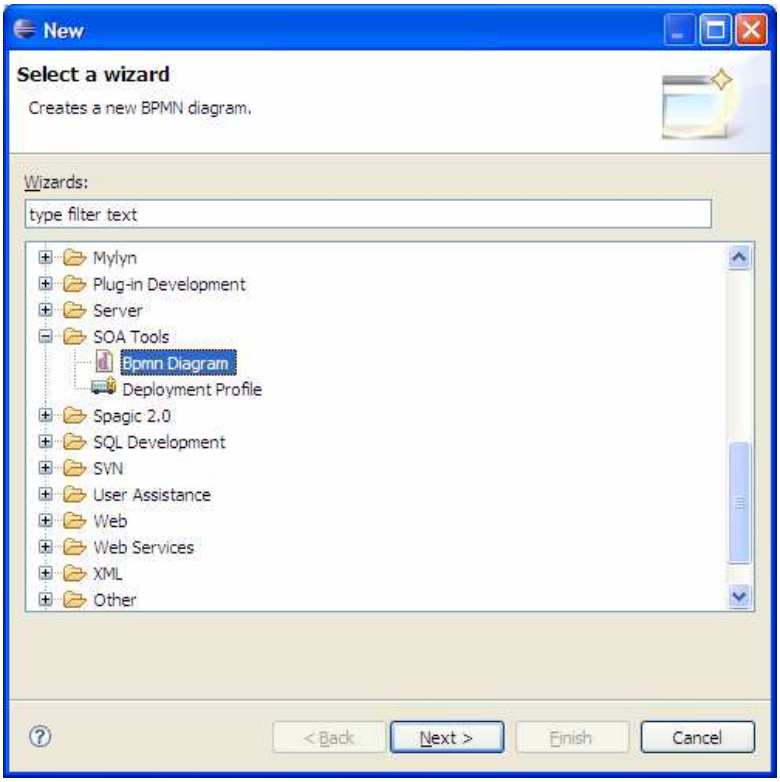
- **Scripts**: This folder contains resources that are used by Scripting Components. Actually groovy is the language for the scripting, so this folder will contain groovy file.
- **SemanticRules**: This folder contains resources that are used by Semantic Validator Component. Rules are expressed in Drools 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.
- **WsdlFiles**: 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 versions manually modified will be deployed in service assembly structure.

The most important folder of this structure is the IntegrationProcess folder because it's the container of the Spagic file that defines Service Assemblies. Other folders are just container of resources organized in standard structure.

1.5 Creating a BPMN Diagram

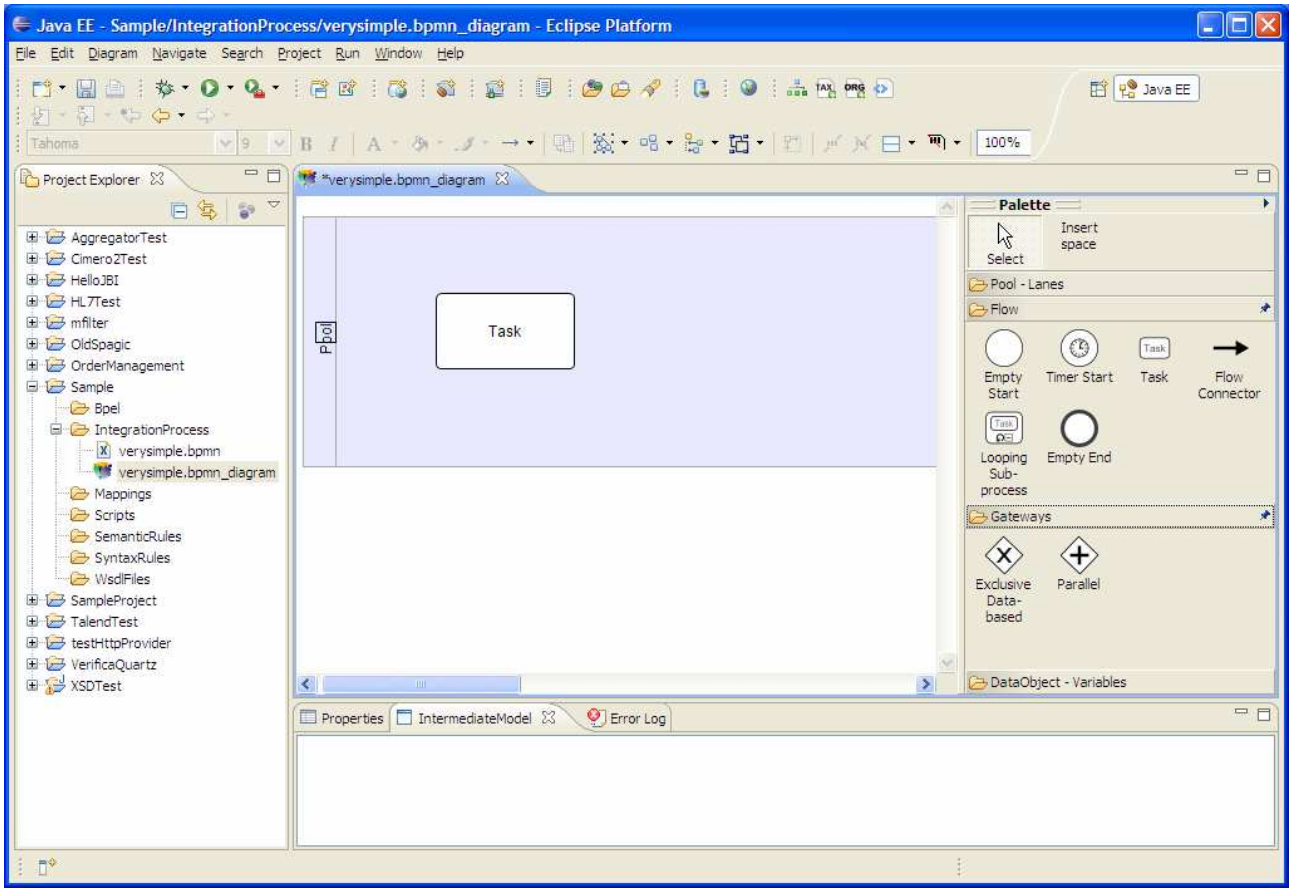
After the creation of the Spagic Project, the next step is to create a new Integration Process.
In this section a very simple example will be created to show the tool usage.

To create the process point the mouse on the IntegrationProcess folder of the "Sample" Spagic project and open the context menu, from here select *File\New\SOA Tools* and choose ***BPMN Diagram***:



Only the container for the processes (preconfigured with the label *IntegrationProcess*) and the file name are required to create the new process.

At the end of the wizard the BPMN file has been created in the project and the Visual BPMN Spagic editor will be opened.

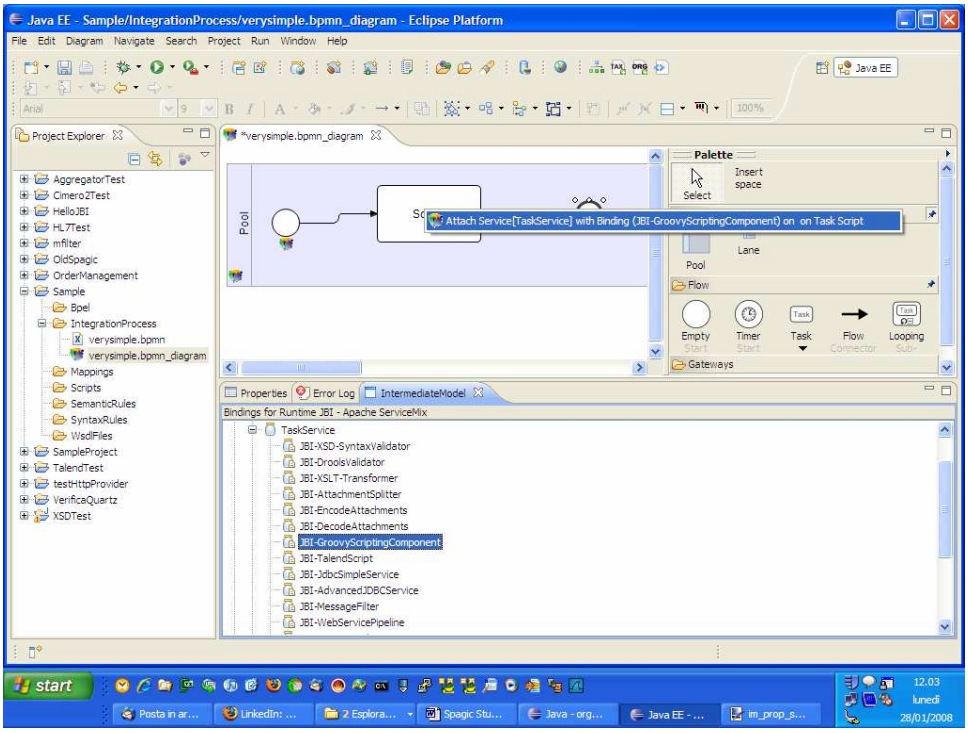


As you can see the Spagic File Editor is a typical “GEF Editor” where you find:

- **The Components Palette**: It’s located at the right of the editor, when you drag a component from the palette to the editing area, a BPMN task is added to the diagram

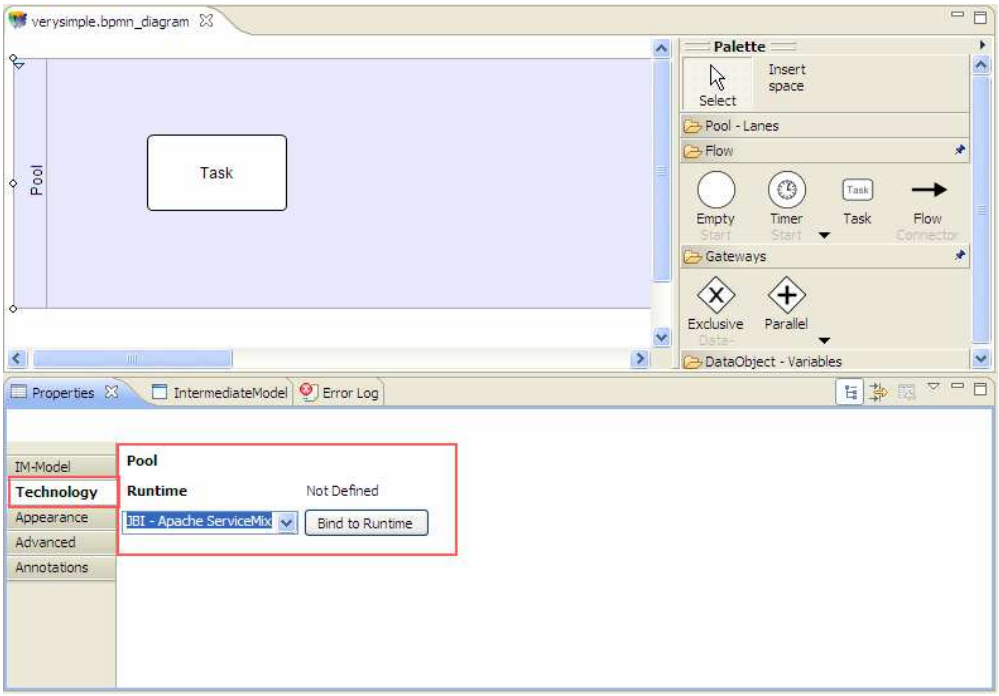
- **The Editing Area**: The area you just use to drag and drop BPMN elements from the palette and to define process flow using sequence or messaging edges.

- **Intermediate Model View**: When you select a particular runtime for a pool, the set of service/service binding available for that runtime will be displayed in this view. From here you could take a particular service binding and drag and drop on BPMN elements as shown:

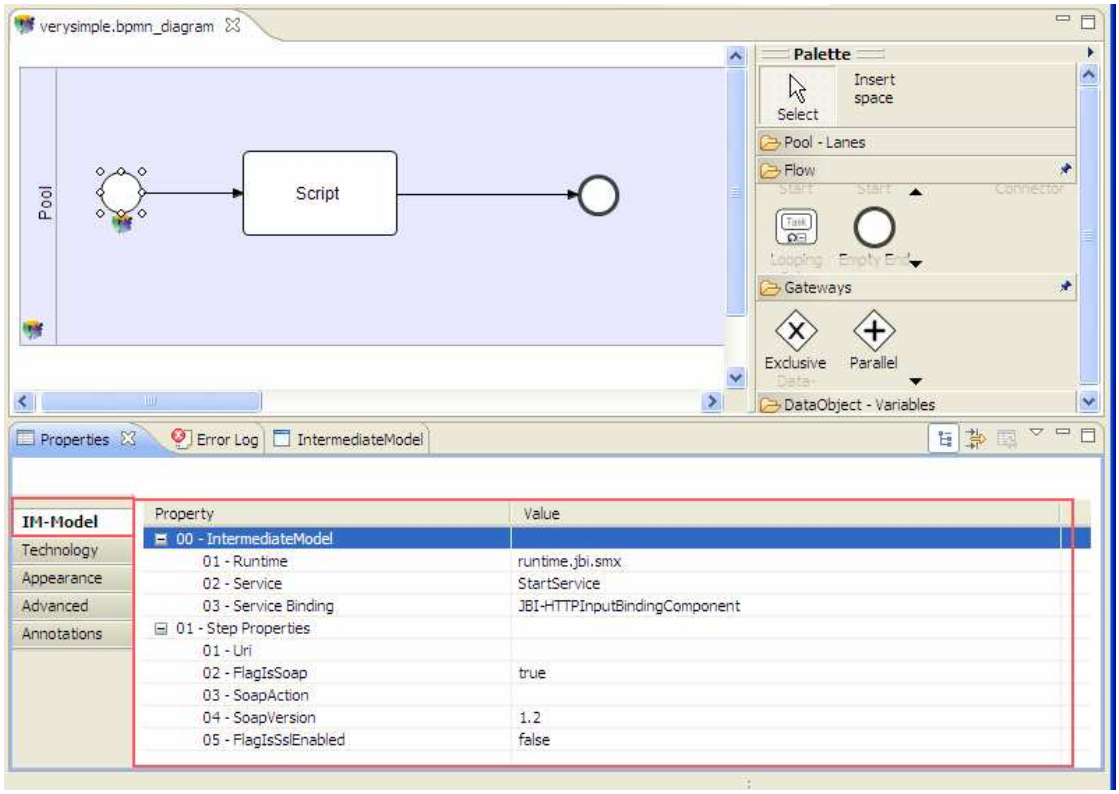


- **Properties View**: When a BPMN element is put in the editing area from the palette, you could view the properties associated to this elements by clicking on this view. This is organized in different properties section, some of this are abstract and related to technologies, some of this are instead very important to define technology related properties, in particular the most important sections for Spagic are:

  - **Technology Section**: It's enabled for BPMN Pools object, and it's the way in which you choose a particular technology for a Pool



- **Intermediate Model Properties Section**: It's enabled for BPMN elements, and let's you to edit all properties defined by service/service binding annotation (for a particular runtime) associated with the BPMN element selected



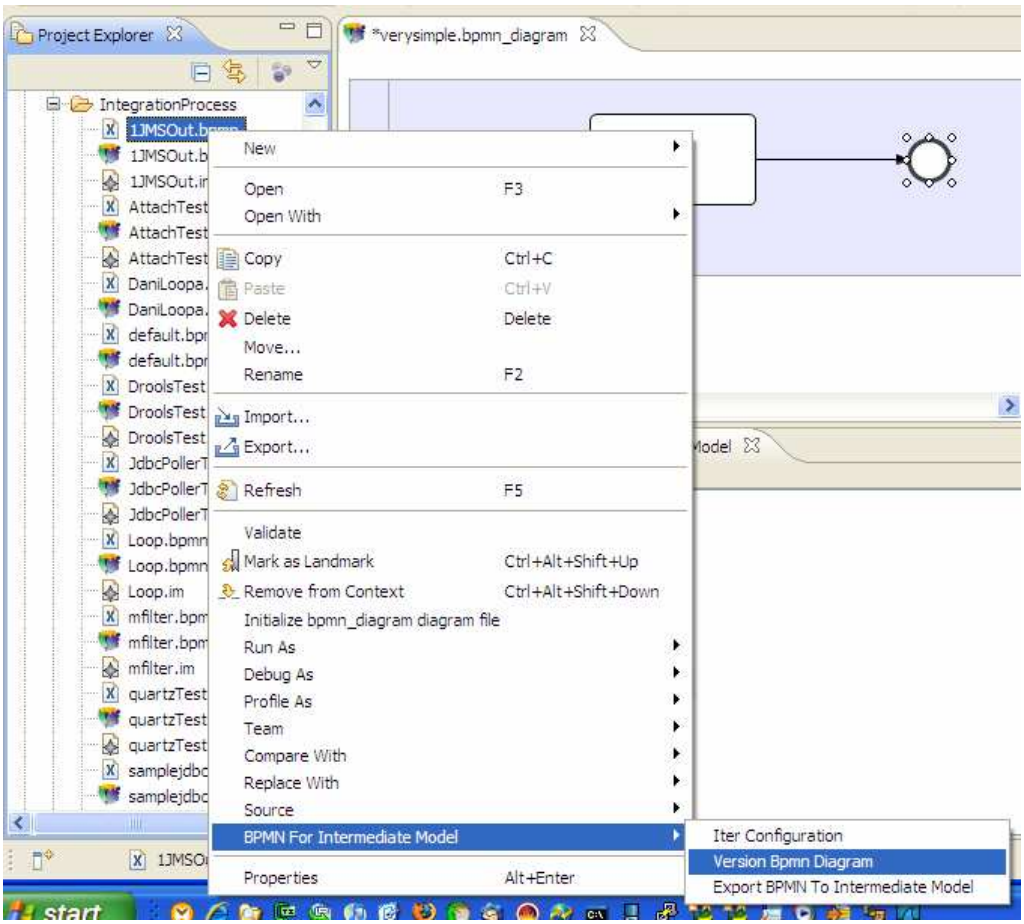
1.6 Versioning of BPMN Diagrams

In this version of Spagic Studio you'll be able to get both in Spagic Service Manager, and in Spagic Metadabase multiple version on the same process.

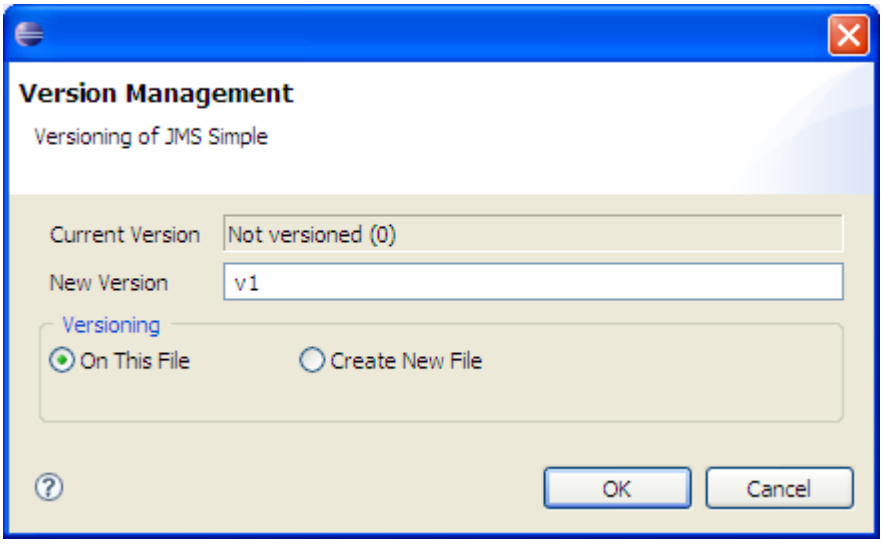
In Spagic Studio versioning is managed at BPMN Diagram Level, this means, that if you're going to change version to a BPMN diagram file, you're going to version all the process (Pool) within a diagram.

Versioning is important, because if for example you want to publish a process that's already in database, you will not able to do without version support.

To manage version on BPMN process, simple open the context menu on "*.bpmn" file and ***select "BPMN For Intermediate Model/Version BPMN Diagram"***



This will open a very simple dialog:



where you could insert the new version number. Here you've two options for Versioning:

- On this file means, the version is updated simply changing the selected BPMN file.
- Create new File (Not implemented yet): if you want to create a new version creating a new file, without change the original one.

1.7 Observable Attributes

As we've seen in section 1.2 Observable Attributes are a very important key concept in Spagic Platform. Basically we want to keep user attention on some data extracted during process execution.

Spagic 2, provide a very simple way to manage observable attributes:

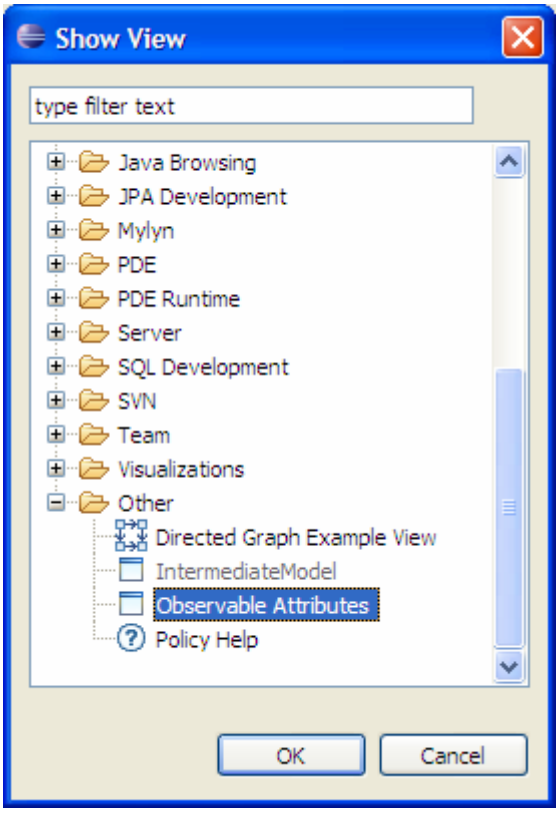
1. Observable attribute definitions and their extraction rules are maintained into a ***centralized repository (Spagic Metadabase)***

2. The Observable attributes ***once defined could be associated with process flow, by a simple drag and drop operation on BPMN DataObject Elements associate with the activity that extract relevant data***.

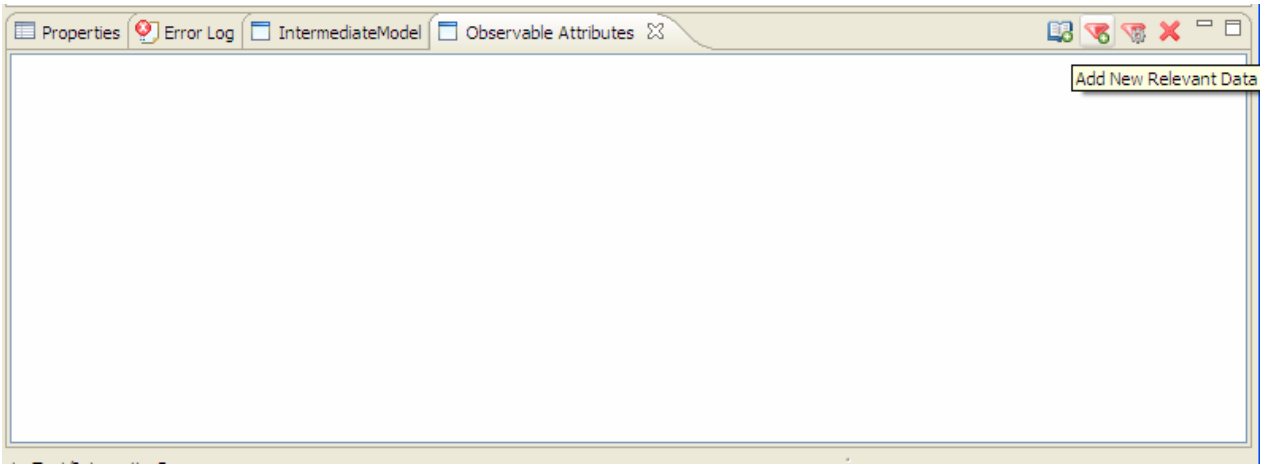
1.7.1 Observable Attribute Repository

To enable the view on Observable Attribute Repository, you need if it's not already there to enable the ***"Observable Attribute"*** View in Spagic Studio. To do this use the ***Window/Show View/Other*** eclipse menu.

Important: Ensure Spagic Database is up and parameters are correct in preference page before open this view.

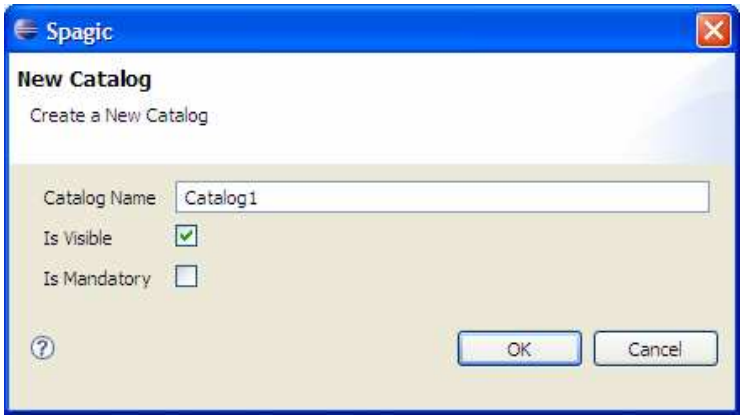


So in the observable attribute you could see all relevant data (ruby icon – stands for important). Please note that the same relevant data in the view could be present also under catalog elements, this do not means the attribute is defined twice but is imply indicate that an attribute could be part of one or more catalog.



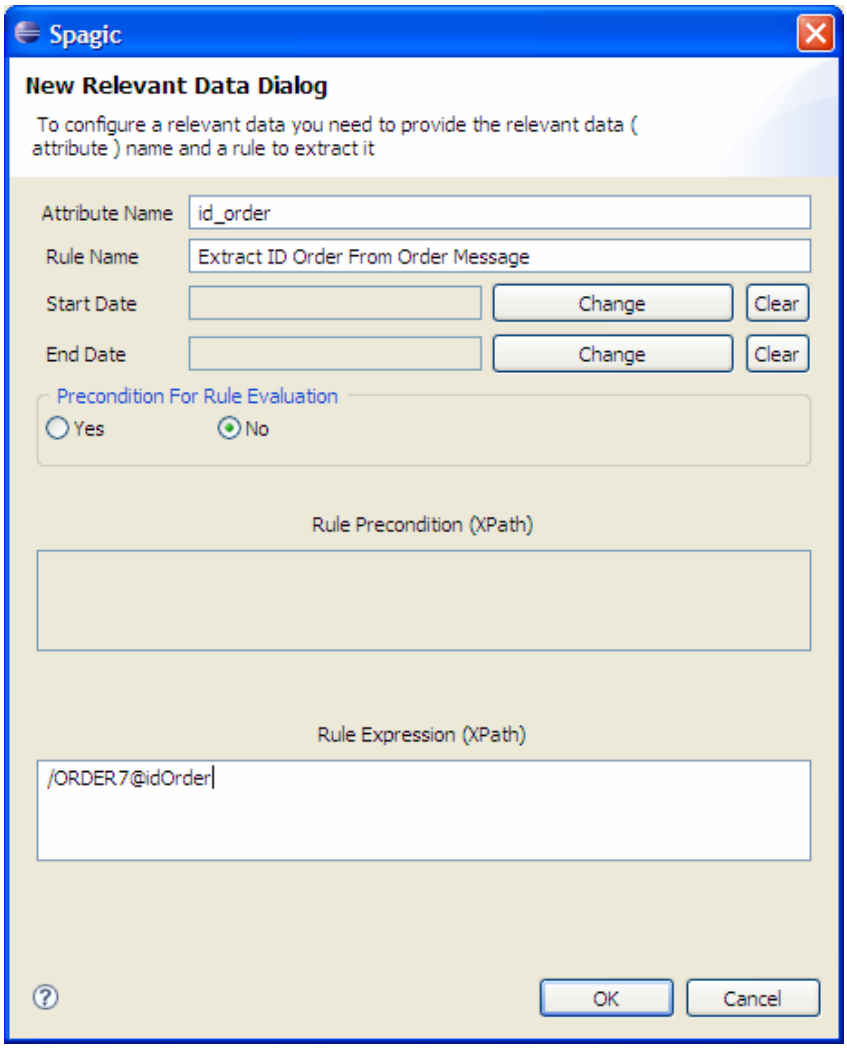
Within the view on the top-right corner, you've four buttons:

1. **Add New Catalog:** Attributes could be organized in catalog, this is only an options to have a better view on Spagic console.

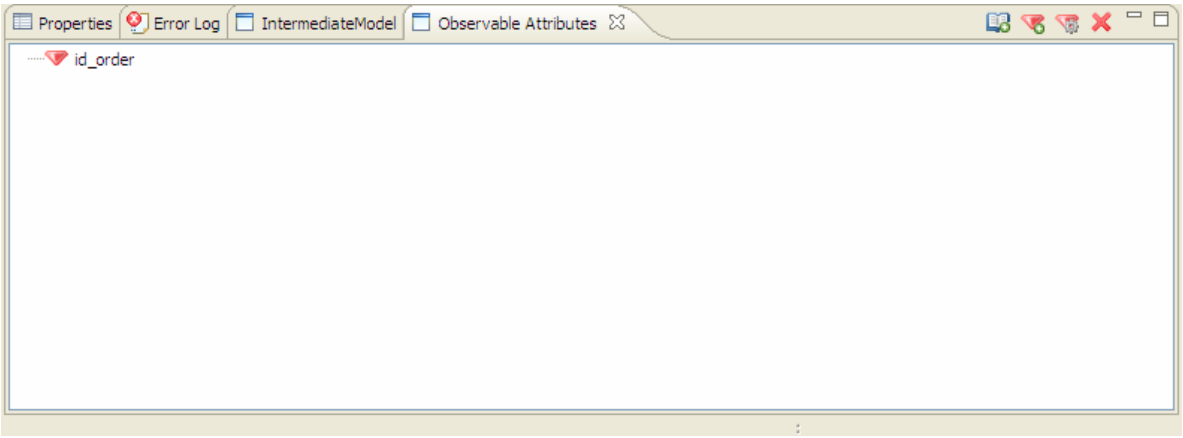


After you finish you'll get the catalog in the repository view:

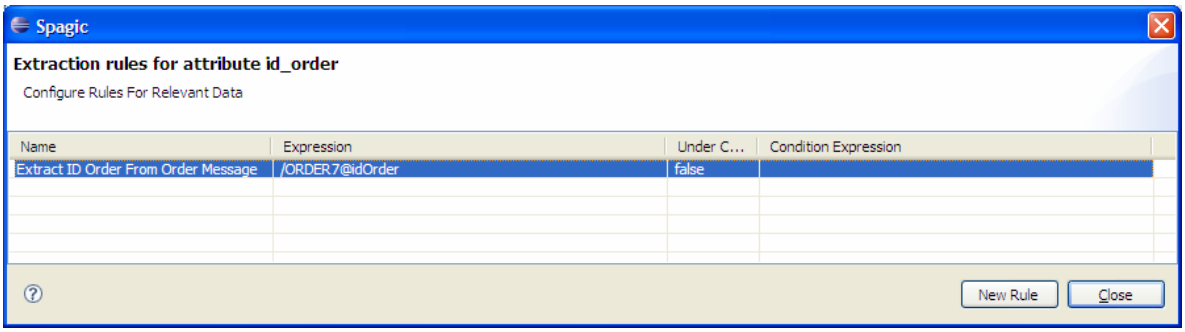
2. **Add New Relevant Data Button:** To add a new relevant data to the repository. To define a relevant data you need at least a name and an extraction rule.



Once you've finish all fields simply press ok, and you'll have the attribute in the repository view:



3. **Configure Relevant Data:** An important thing to keep in mind **is that a relevant data could have more than one extraction rule** (because message types are different, or simply the same relevant data needs different rules for different technologies and another rules within another technology). This button enables to add, remove. Modify extraction rules for the selected relevant data in three.



4. **Remove Button:** To remove relevant data and catalog from repository, please not that the delete of the catalog do not remove the relevant data associated but only the relations within it.

1.7.2 Associating Observable Attributes with Process

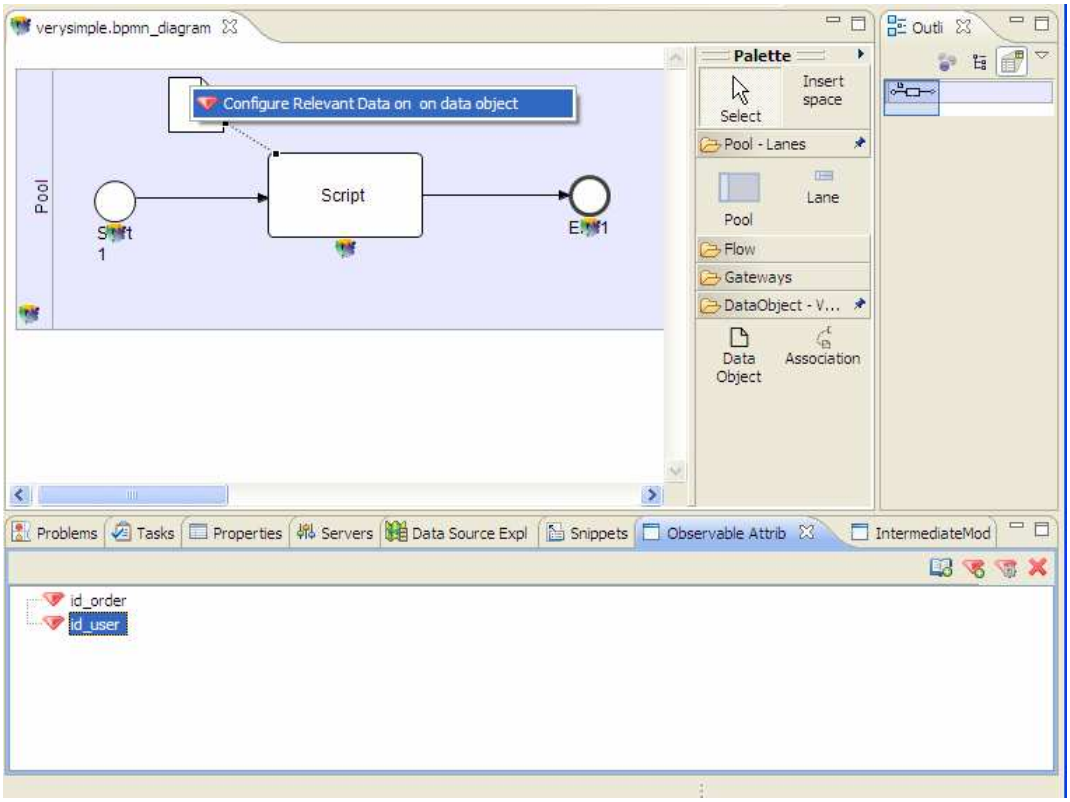
In the previous section we've seen how to manage the observable attributes repository, in this we'll go to see how to associate this relevant data to process steps.

The key concept is that a ***step extract a set of observable attributes*** (one or more) so we need a way to:

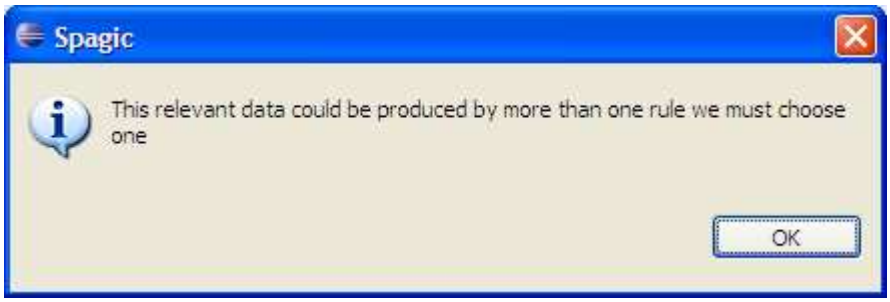
- Define a set of relevant data
- Associate this to a particular process step

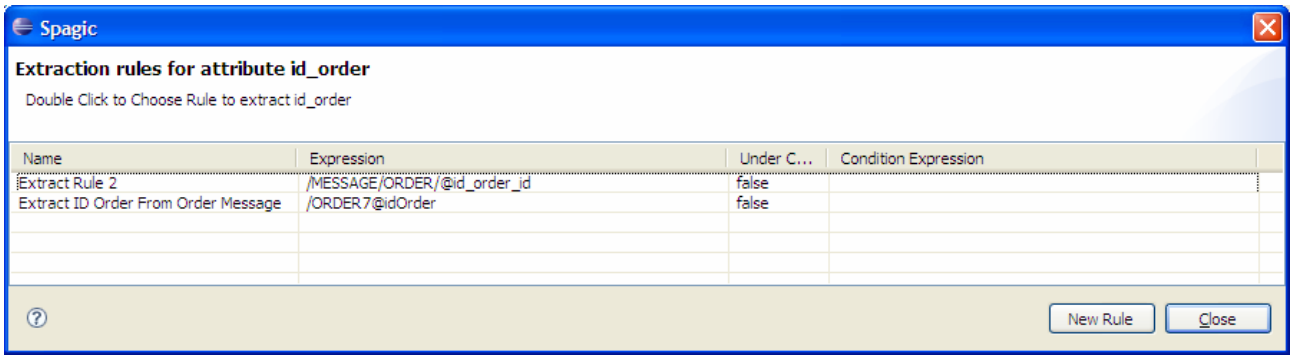
The best way to achieve this is to use the BPMN Data Object element because:

- It could be annotated like other BPMN elements.
  We use a BPMN Data Object to represent a set of Observable Attributes.
  To ***add an Observable attribute to a BPMN DataObject you need only to drag and drop observable attribute from repository view to the BPMN DataObject.***



The only thing you need to pay attention is ***than if an Observable Attribute has more than one extraction rules***, ***when you do the drag and drop on the BPMN DataObject Spagic studio ask you which rule you want to use.***





- It could be easy be linked to BPMN task activities using the “BPMN association” elements, this will definitely solve the problem of the link of observable attributes to process step.

1.8 Define and configure Iters

Spagic supports the concept of ***Iter Types*** and ***Iter***. The link between the concept of ***Iter Type*** and the concept of ***Iter*** is the same that there's between Process and Process Instance.

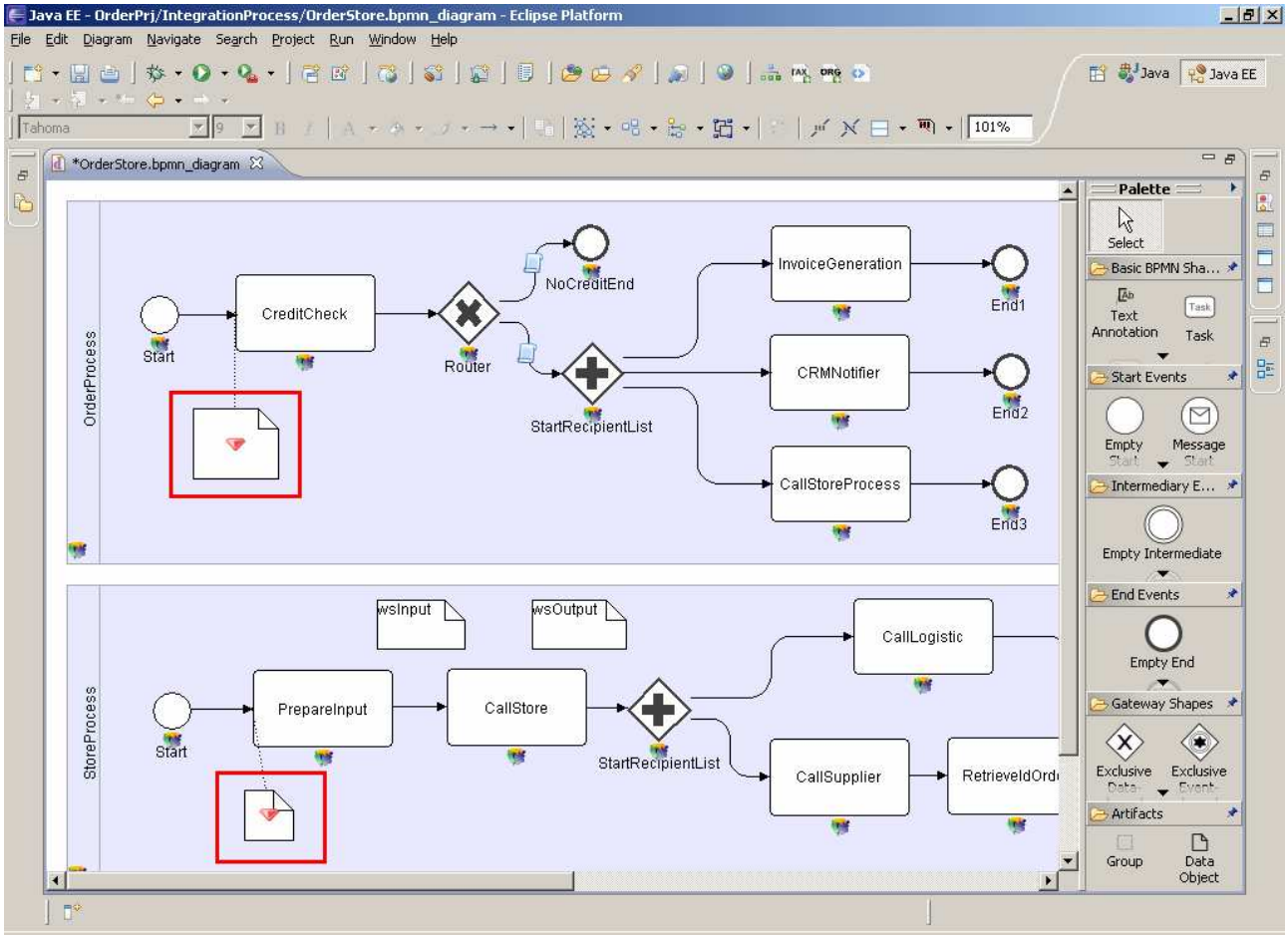
Iter type is a higher level of abstraction of the Process concept. With Iter Type we can **logically correlate two or more Processes that share a set of common attributes**. This is the reason why the same attribute can be produced by different rules. To define an iter type we need to:

- ***Define the Processes that belong to the Iter:*** **All the processes defined within the BPMN pools are considered belonging to the iter, if an iter type for the BPMN diagram is defined.**

- ***Define the attributes set of the Iter*****.** The correlation of process instances to iter instances is done matching the relevant data of process instances***. If two process instances share the same set of relevant data, and their process definitions are in the same Iter Type definition, then they're part of the same Iter Instance.***

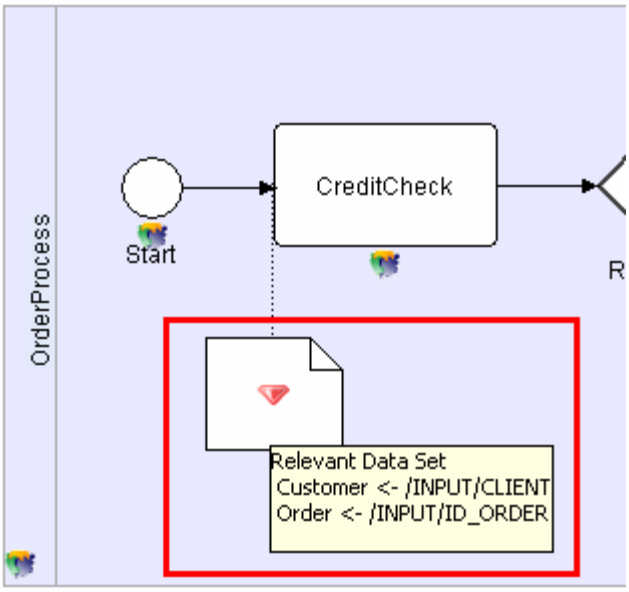
In Spagic Studio the steps to configure an Iter Type are:

1. Define the relevant data to use on each Pool of your BPMN diagram. Please consider the following sample:

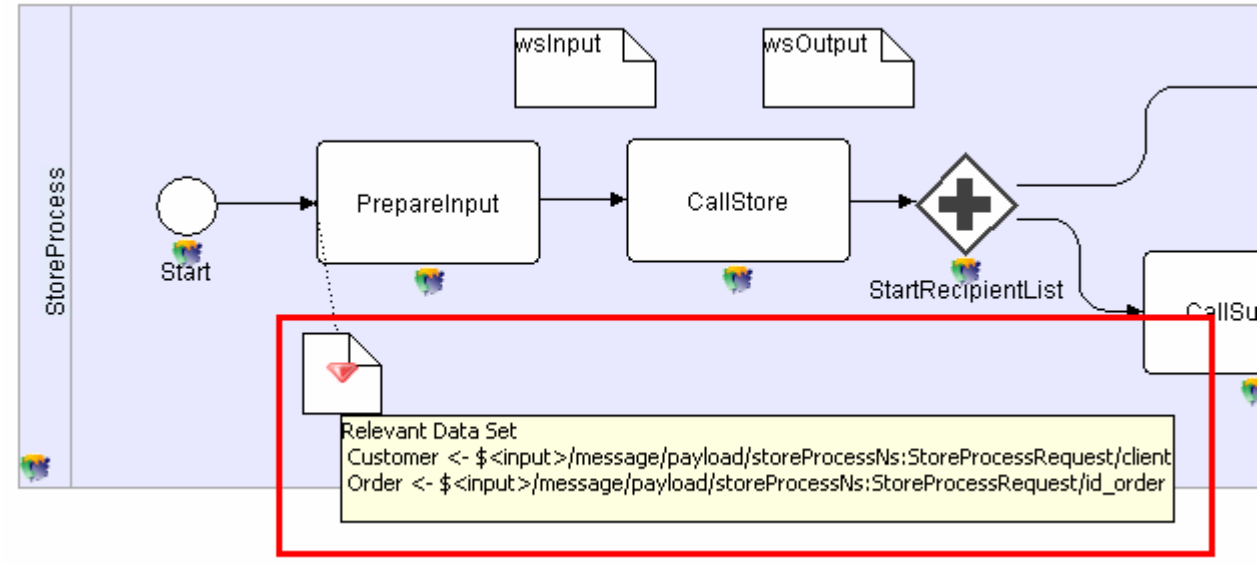


In this BPMN diagram, 2 pools are defined: on both pools some relevant data are configured, and these relevant data will be used to determine that the 2 processes belong to the same iter instance, because the relevant data rules extract the same attributes for both processes.

As you can see in the next screenshots, the attributes that determines the iter membership of the processes are *Customer* and *Order.*

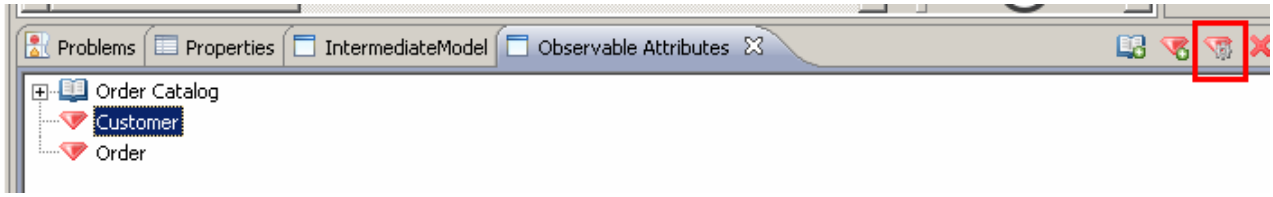


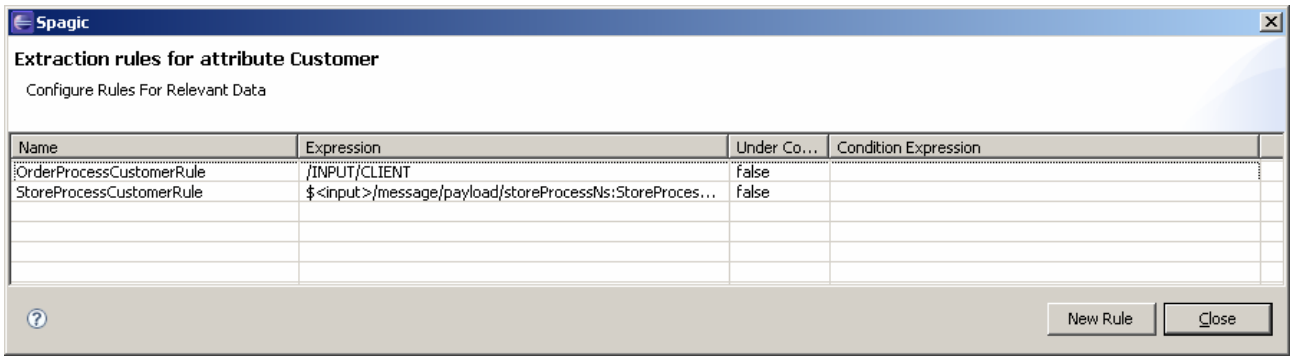
While the iter attributes are the same for both processes, the rules for extracting the 2 attributes are different because the processes work on different data.



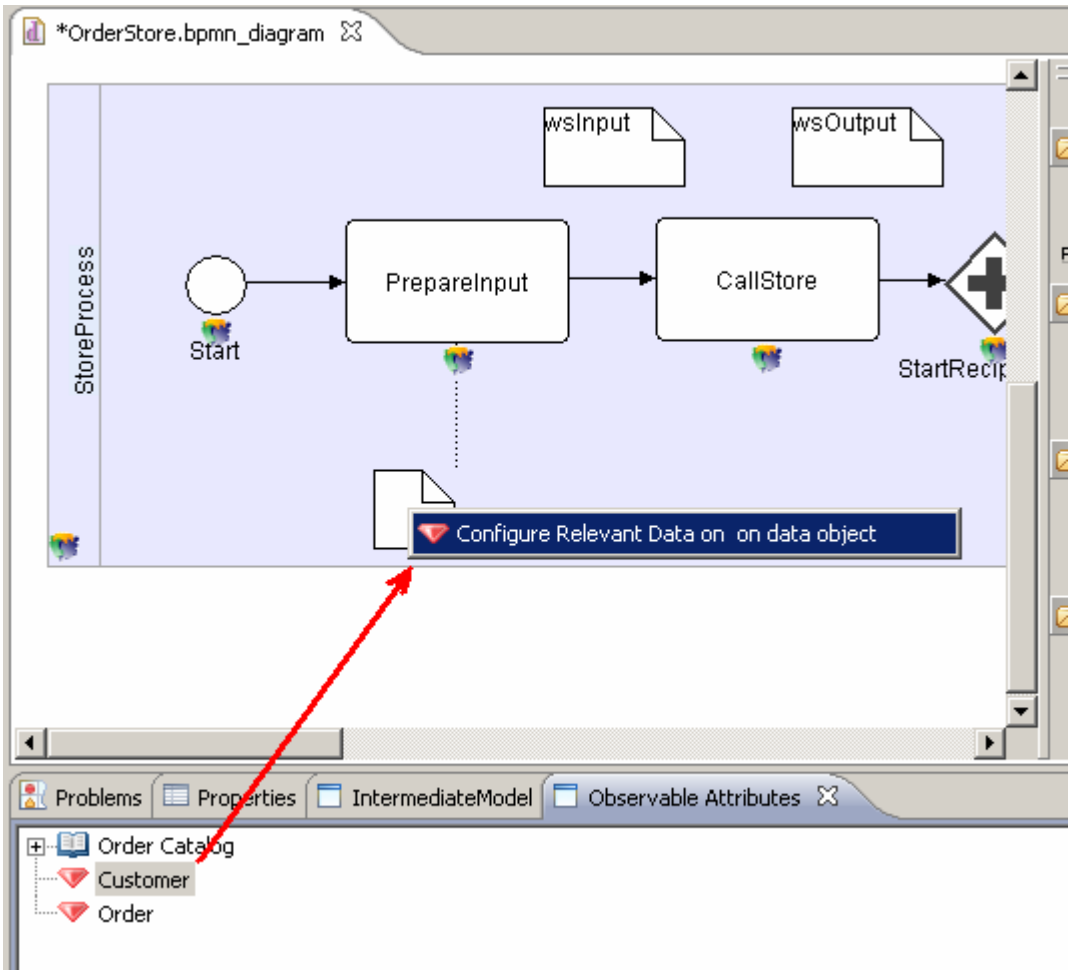
This is possible because when you configure an attribute you have the possibility to configure multiple rules for retrieving this attribute.

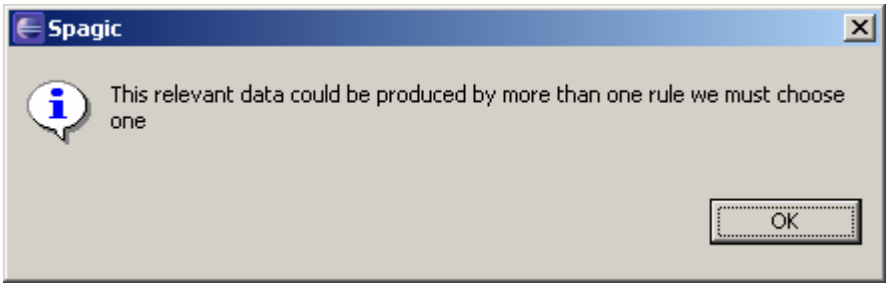
For example on the next picture you can see that we configured the *Customer* attribute with multiple rules.



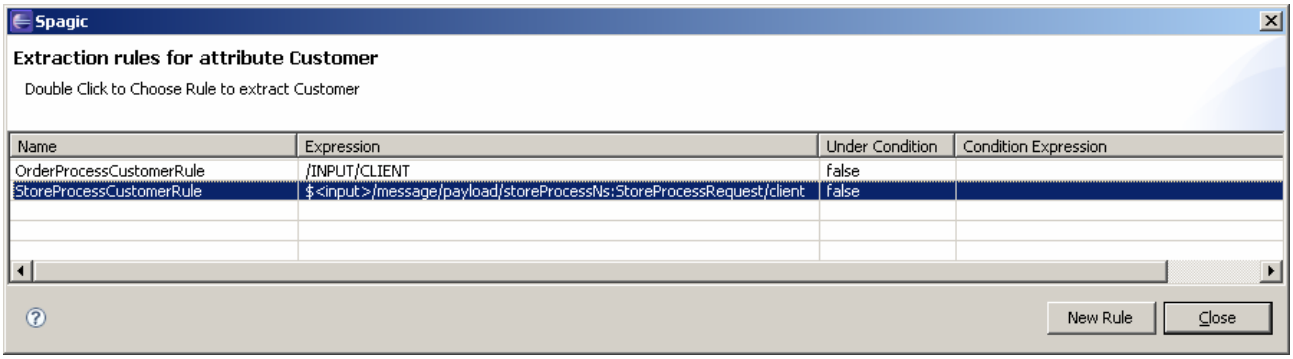


When you drag an Observable attribute on the Data Object, and this attributes has multiple extraction rules configured, Spagic Studio ask you which rule has to be used to extract the attribute.



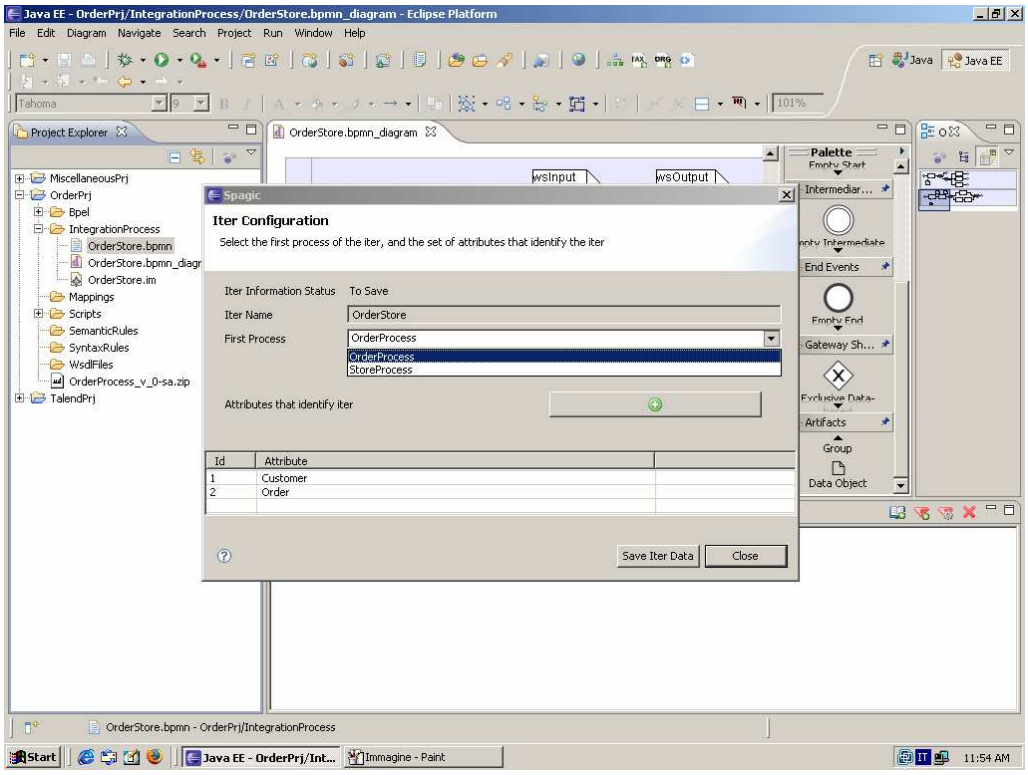


In this case you have to select the rule to apply for the process you are working on.



2. Configure the Iter type: this means specifying the first process of the Iter, and specifying the attributes that define the *Iter Attribute Set*.

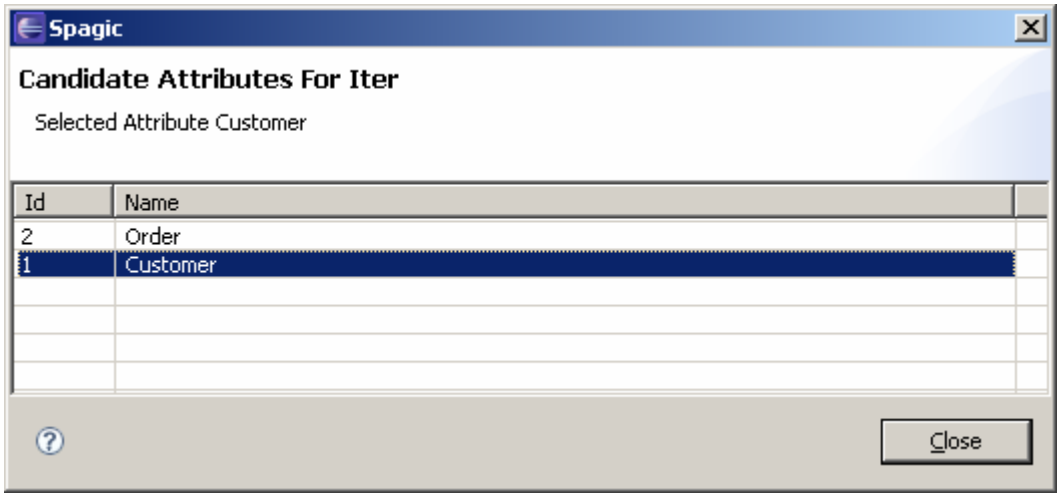
   To do this, select the ".bpmn" file and choose *BPMN For Intermediate Model->Iter Configuration*, the following dialog will appear:



You have to specify the following information:

- The first iter process: Spagic needs to know which is the first Iter process, to be able to determine when to create a new iter instance.

  All other processes in the BPMN diagram will be considered belonging to the iter, no matter of their invocation order.
- The iter attributes: choosing the (+) icon all the common attributes of the pools within the BPMN diagram will be displayed. Select the attributes to compose the iter attributes set.

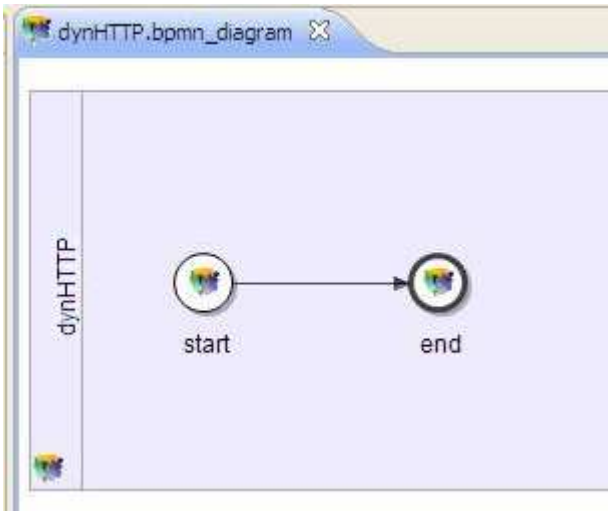


1.9 Using dynamic parameters in the processes

Usually the process configuration is static: this means that once a component in a process is configured, the only way to change it, is using Spagic Studio by changing the BPMN diagram and regenerating all the other files (the *.im* and the *.zip*). Sometimes this approach is uncomfortable, because for example a process has different parameters on a test environment and on a production environment.

To avoid changing the process when moving the process from an environment to another, most components supports the usage of *dynamic parameters*: this means that you can specify the value of a parameter externally to the process (in a properties file), and change this value without changing the process.

Suppose, for example, that you have the following simple process:



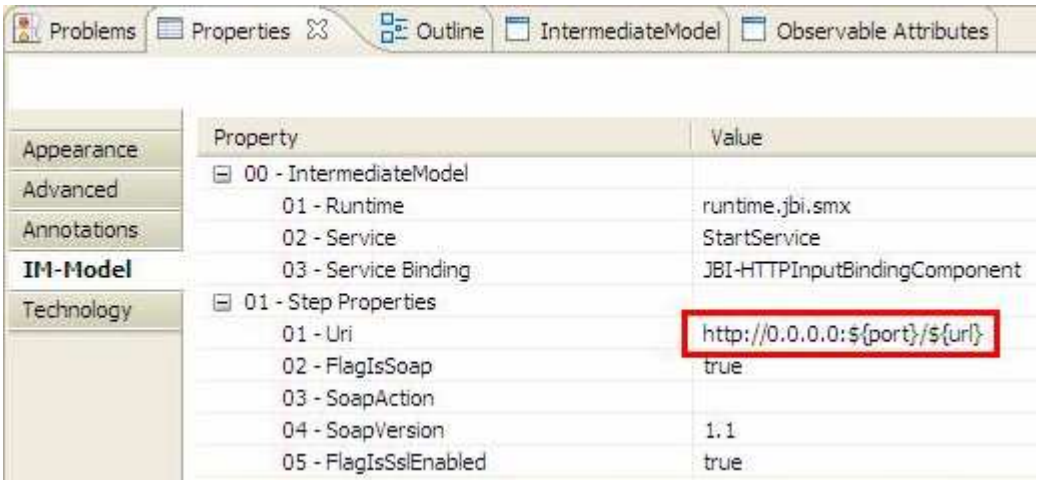
where the first binding component is a HTTP input component that exposes the process through a HTTP URL.

Suppose also that the URL is different between the test environment and the production environment.

In this case you can use dynamic configuration instead of static configuration.

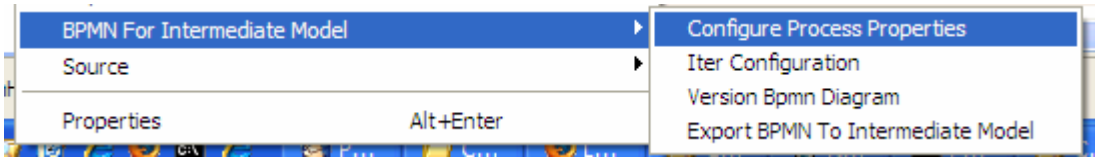
You can specify this kind of parameters using the syntax **${parameter name}** in the component configuration.

For example:

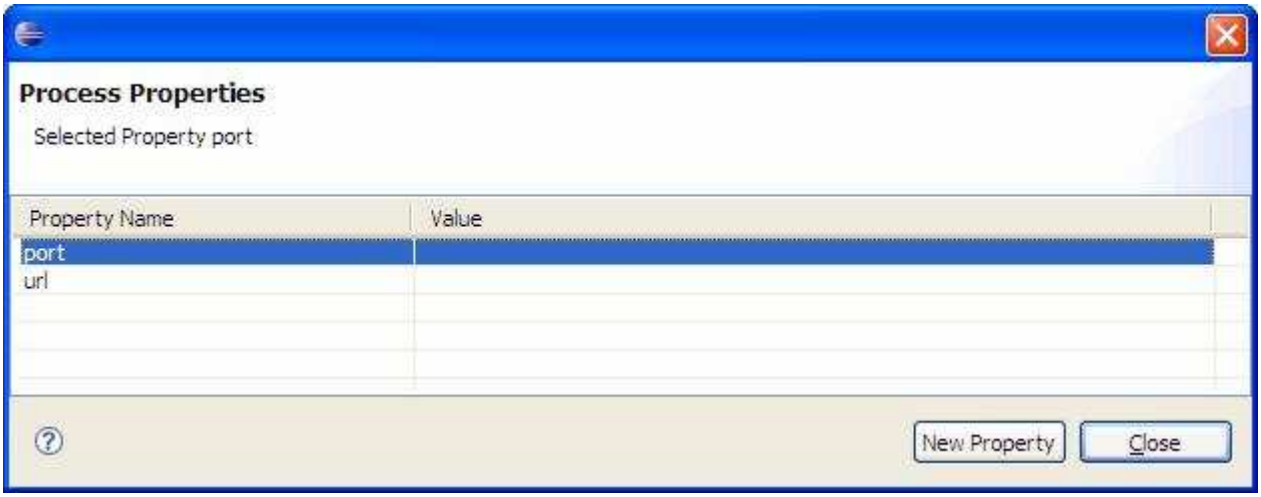


In this sample we used 2 parameters: *port* and *url*.

The initial values of these parameters can be configured selecting BPMN diagram and choosing the option "*Configure Process Properties*".



The following dialog allows defining initial value for these parameters:



After the parameters values are defined, when you create the Service Assembly also a properties file called dynHTTP_v_0.properties will be generated.
You have to put this file in the ServiceMix folder *resources\properties,* **before copying the service assembly to the *hotdeploy* folder**.
Later, when you move this process on another ServiceMix, you can change the properties file for exposing the process on another URL without changing the Service Assembly.

1.10 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.

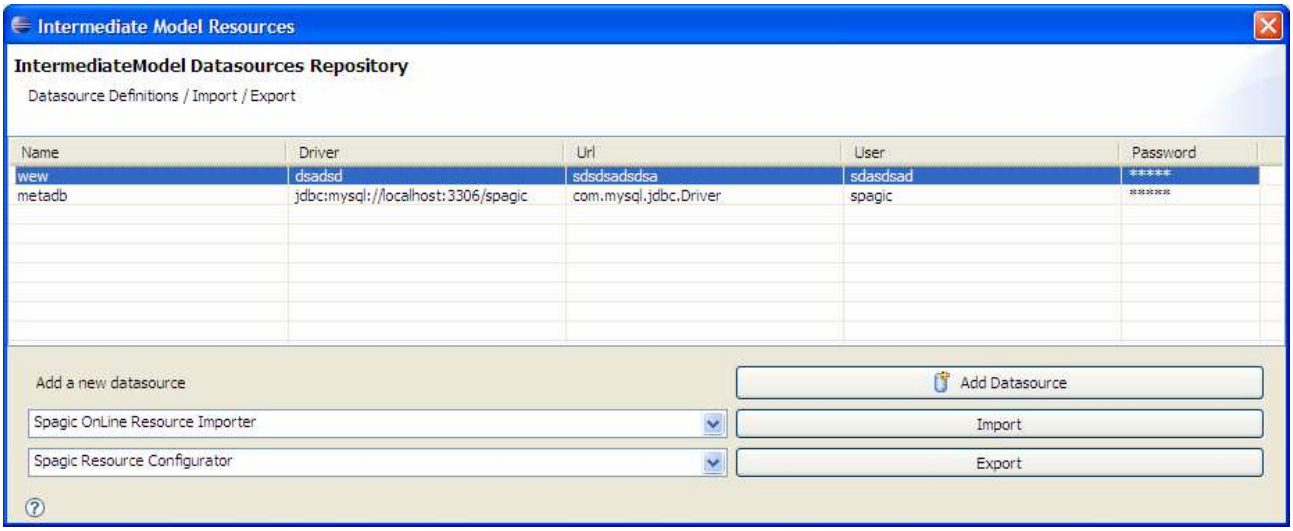
To avoid “missing datasource problems” during the deployment phase we need a way to provide bidirectional synchronization between datasource defined in Spagic Studio and datasources defined in the ESB runtimes.

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

- Import datasources from a particular runtime
- Export datasource configuration for a particular runtime

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 the datasource defined in Spagic Studio environment. From here you can:

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

2. Import datasource definition using specific *Importer*, Spagic Studio defines two standard Importers:

   - ***Spagic Online Resource Importer***, try to get datasources list defined in a running ServiceMix.
   - ***Spagic Configuration File Resource Importer***, ask you to provide the SMX_HOME\conf\jndi.xml and get datasource parsing that file.

3. Export Datasource using a Resource Configurator, Spagic Studio define a standard Spagic Resource Configurator:
   - ***Spagic resource Configurator:*** This will generate the xml configuration part for the selected datasources that you only need to copy to jndi.xml file.

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 decide what you want to do.

1.11 Namespaces

All the messages exchanged in ESB are Normalized Message, where the most important part (the payload) is XML content. In this context XML and related technologies are very important. One of the most important is the XPath technology. A lot of components and concept in Spagic are related to XPath.
To work correctly in Spagic you need an XPath skill. This document doesn't cover XPath.

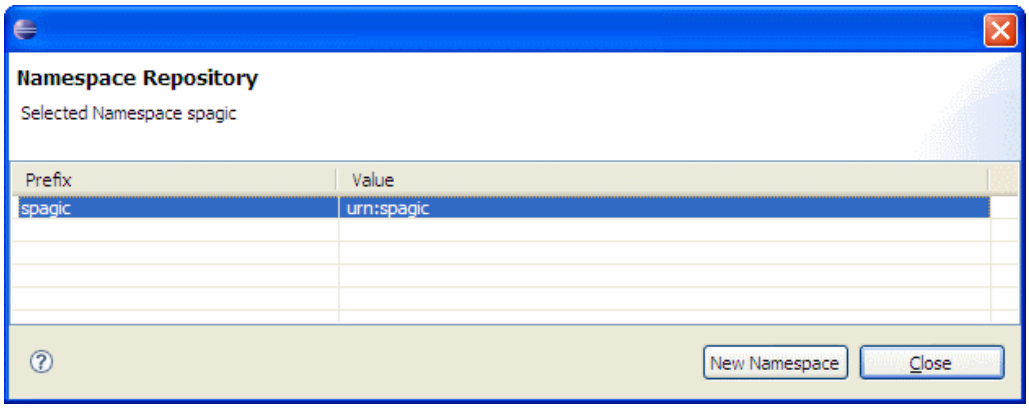
XPath is a quite simple technology to learn but some problems could arise when we're using XPath expression against xml content with namespaces.

It's very important to know that if we want to write an XPath expression using a namespace prefix, the XPath engine must know the namespace value associated to this prefix. Briefly the XPath engine must be aware of all namespace prefix used in XPath expressions. If XPath engine is not configured properly we can have wrong evaluation of expressions.

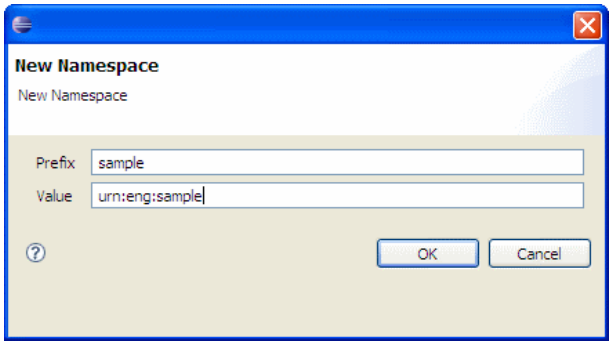
Spagic components that use XPath allow declaring in their configuration all the namespaces used. To avoid specifying this information for each endpoint, in Spagic Studio the following strategies are defined:

- ***All Namespaces are declared globally.*** Using the button showed by the image:

A typical master detail dialog will be opened:



Where it's possible to see, add, and delete namespaces.



The namespace configuration will be automatically generated for each component that requires it, freeing the process developer to do that manually.

1.12 Generate Intermediate Model

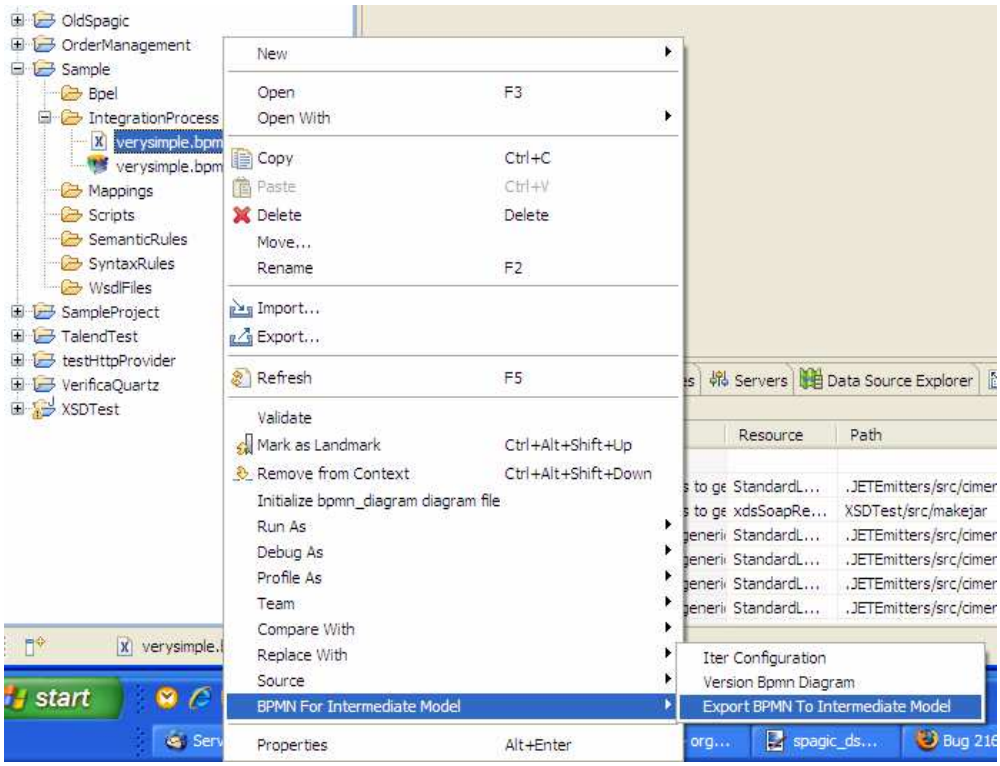
Intermediate model, is a standard EMF Model, that Spagic studio use to decouple “Process Design” and “Process Code Generation/Deployment” phases.

The Intermediate Model is the base model, that Spagic studio use to decouple process editors (BPMN) from code generators (for example Bpel and jbi).

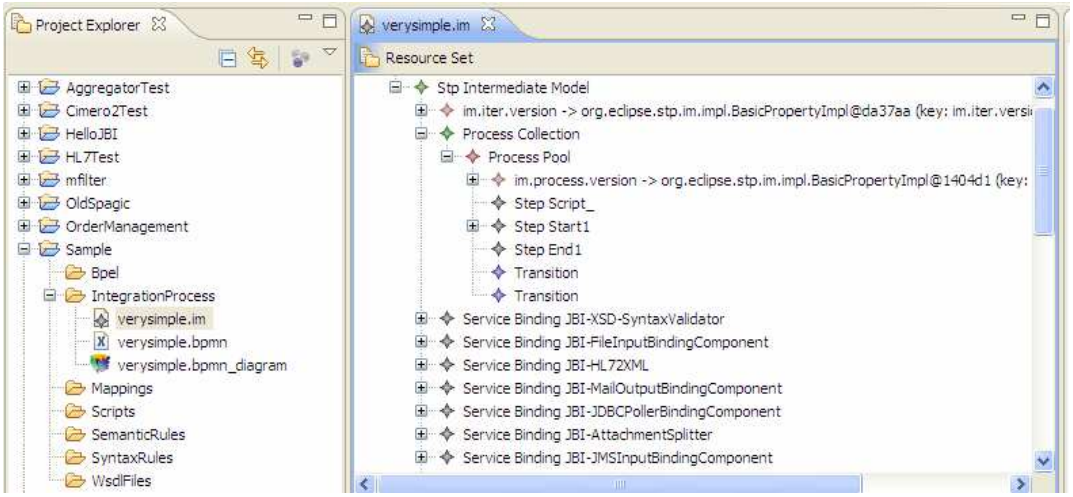
This document is about Spagic studio and not about intermediate model, but it’s important to understand that:

- The extensions of BPMN Editors are targeted to get a rich intermediate model
- Intermediate Model, is the model that will be published in Spagic Metadabase
- Intermediate Model, is the starting point for all Spagic 2 code generators (for example BPEL and JBI)

To get the intermediate model from BPMN files, use the context sensitive menu “BPMN For Intermediate Model/Generate Intermediate Model”

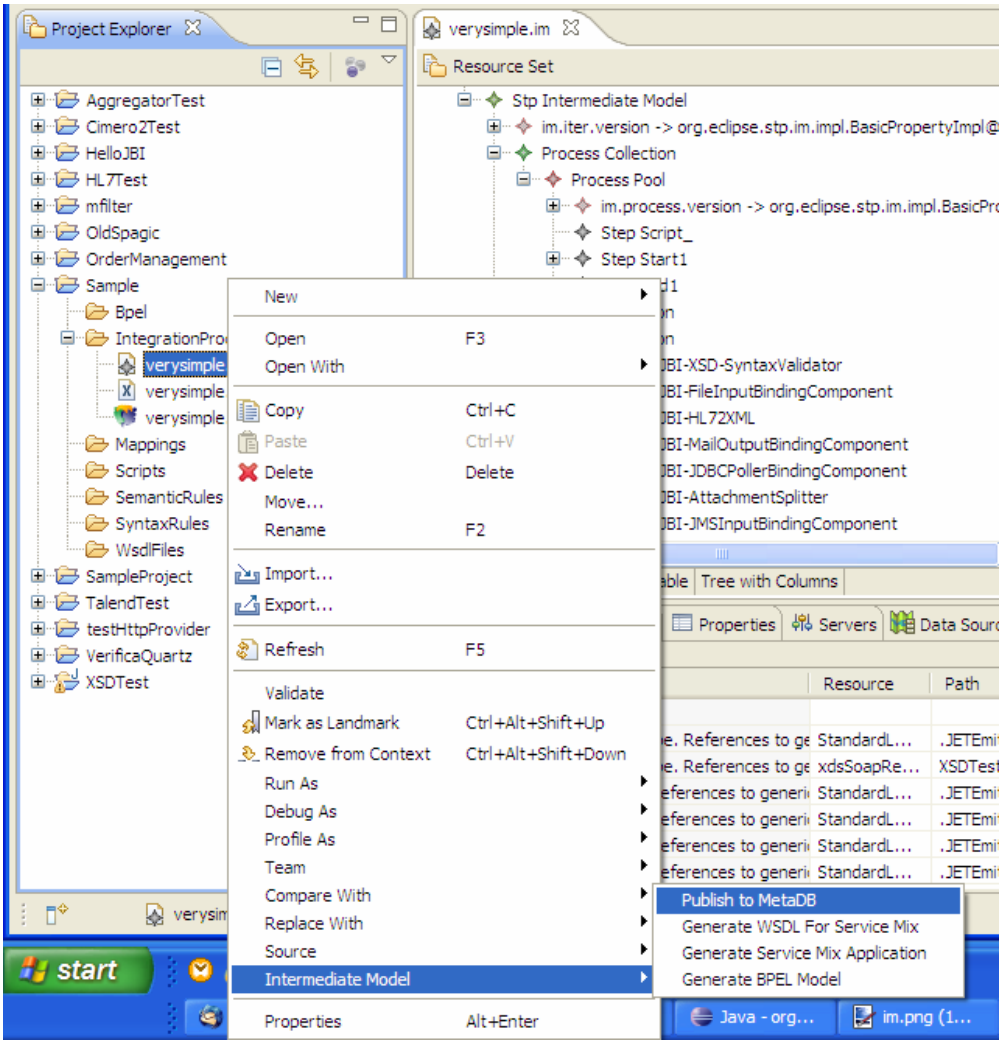


Once you’ve obtained Intermediate Model file (*.im) you could edit using Intermediate Model EMF Editor:



1.13 Publish Intermediate Model to Database

With Intermediate Model, you could publish all processes contained within Intermediate Model in the database using the context sensitive menu available on im files: ***Intermediate Model/Publish to MetaDB.***



1.13.1 Publish Intermediate Model to Database without Spagic Studio

The publishing of a process on the MetaDB can be performed also with a Spagic command line tool called ***spagic-deployer***.

This tool is available in the Spagic distribution within the spagic-service-manager${SPAGIC-VERSION}.zip, in the folder ***spagic-deploy***.

In current Spagic Studio, when the command *"Generate ServiceMix Application"* is selected, at least two files are generated:

- **<Process name>_v_<version>-sa.zip**: service assembly to be deployed in ServiceMix.
- **<Process name>_v_<version>.deploy**: deploy file necessary to publish the process on the Spagic MetaDB without Spagic Studio.

To publish a process with the command line tool, open a command prompt in the *spagic-deploy* folder and use the command:

spagic-deploy <url of file process.deploy>

This command requires that ServiceMix is running, for publishing the process.

If the publishing is successful, the following message is displayed:

Calling deploy service at URL: http://localhost:9092/deployProcess/

Deploy successfully

The following options are supported:

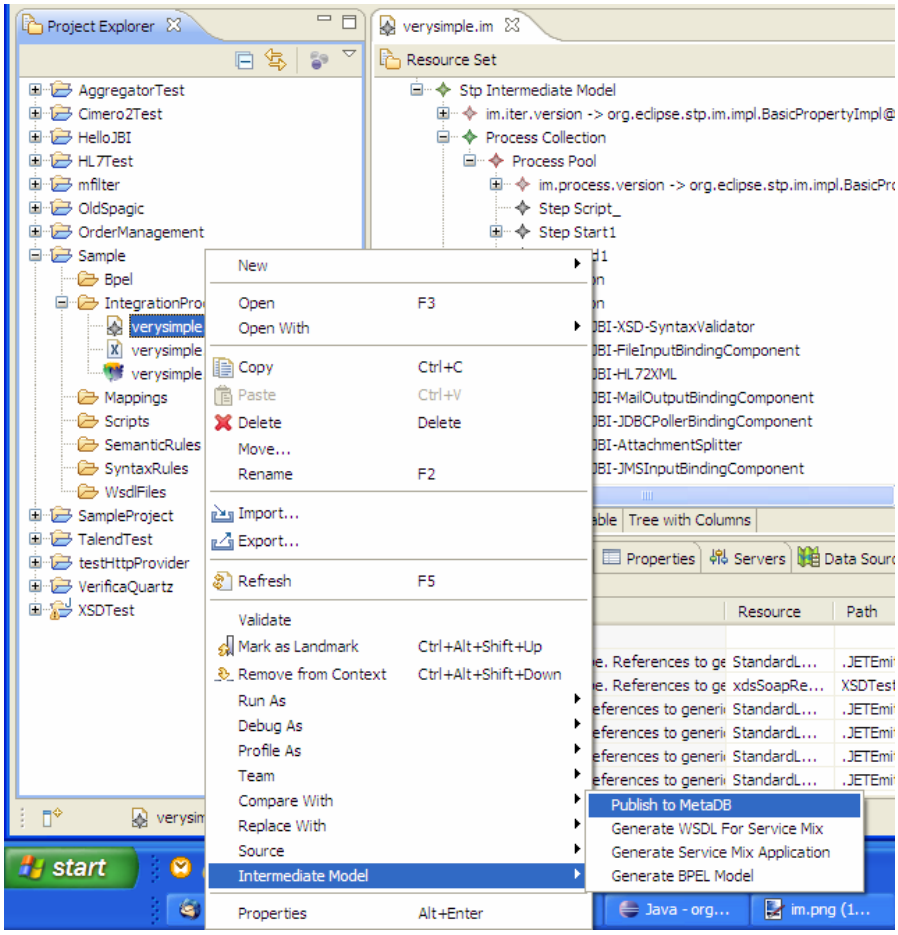
-host:<host name or IP address>	ServiceMix location (default: localhost)
-port:<port number>	ServiceMix Deploy Process port (default: 9092)
-update	Update the process information if the process structure is the same

The most important option (not yet implemented in Spagic Studio) is "**-update**" that allows to republish a process that was already published, if its structure is not changed, for example because only some configuration properties were changed.

1.14 Generate Code for ServiceMix and Process Deployment

The starting point for generating code for Apache Service Mix is the Intermediate Model.
On the Intermediate Model file using the context menu, you have two options for ServiceMix:

- **Generate WSDL For Service Mix**: This is useful if in one of the processes in the BPMN diagram, you put a *JBI-HTTPInputBindincComponent, or JBIHTTPInputOutputBindincComponent* in your process starting point. This simple means you want expose you process with a SOAP interface. The WSDL is useful for the client that need wsdl to generate client stubs.
- **Generate ServiceMix Application**: This run the code generator for ServiceMix against the intermediate model. The code generator will generate a different service assembly for each pool that belong to Apache ServiceMix Runtime and will ignore pools bounded to other technologies.



1.15 Manage Process Publication on UDDI Registries

As described from the previous section one of the operation available on integration processes is the publication on a business registry. From a high level point of view we can say that a ***business registry is a repository where we can publish services with their definitions, and provide a classification of these based on taxonomies.***

The important concepts about business registry are:

- ***Organizations***: A service published within a business registry must be associated to an organization that is the provider of the service.

- ***Taxonomies*** are finite set of values defining "domains", and classifying a service by a taxonomy means to choose one or more value from the taxonomies and assign to service. Important concepts are:
  1. A service can be classified one or more time within the same taxonomy
  2. A service can be classified one or more time in different taxonomies

- The main feature of a business registry is to provide search features based on organizations and classifications. Typical query for a business registries are:
  1. Search all services provided by "organization X"
  2. Search all services classified as "payments" by the business taxonomy
  3. Search all services provided by "organization X" classified as

Organization and taxonomies need to be defined at the beginning of project and often are stable in time, so in Spagic Studio we offer two utilities to preload organizations and taxonomies text files.

To **"Load organizations"** in Business registry use the following button:

Spagic Studio will ask you for a ".org" file. It's a simple text file like the example:

```
#OrganizationName;OrganizationDescription
Engineering;Engineering Ing. Informatica
Bank XYZ;Description for Bank
Institute ABC;Description for Institute ABC
```

To **"Load taxonomies"** in Business registry use the following button:

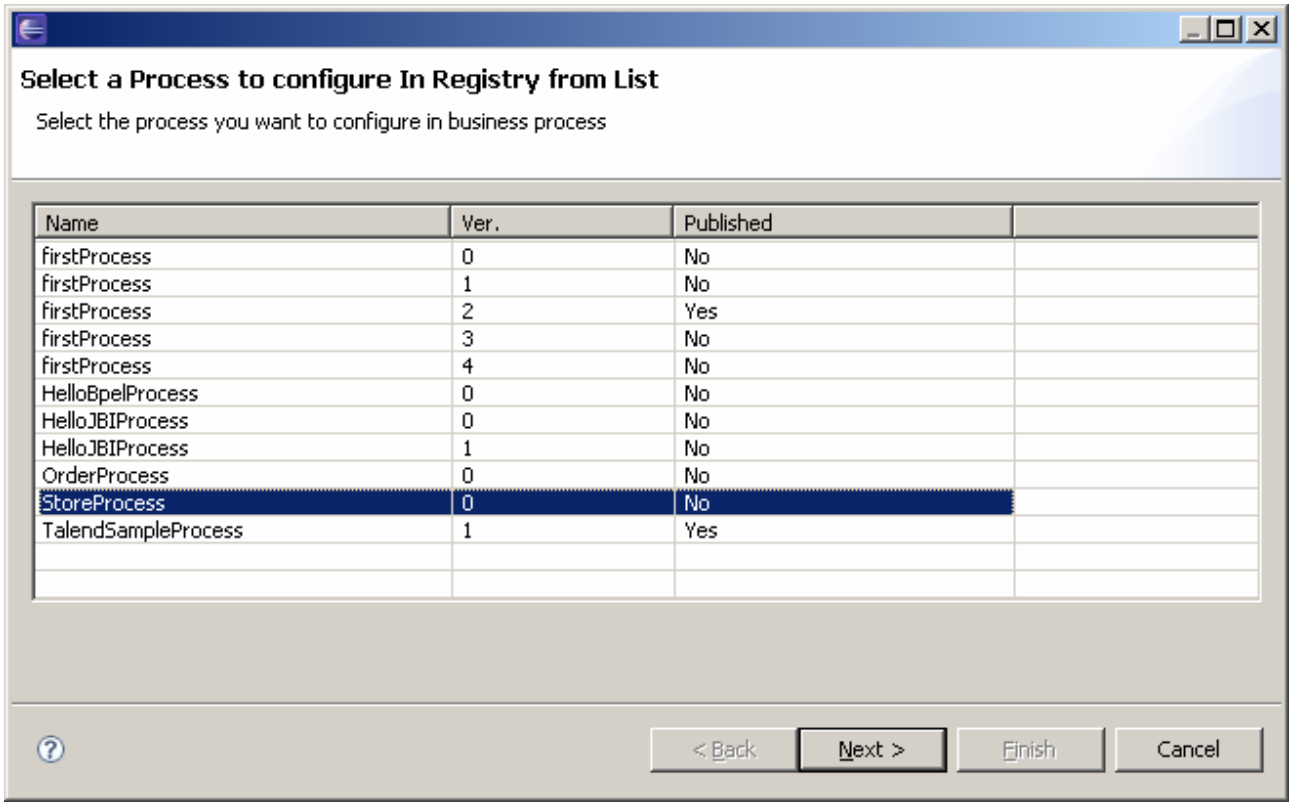
Spagic Studio will ask you for a ".tax" file. It's a simple text file like the example:

```
#TechnologyTaxonomy
->Java Services
->Web Services
->CICS Services
->Cobol Serivices
```

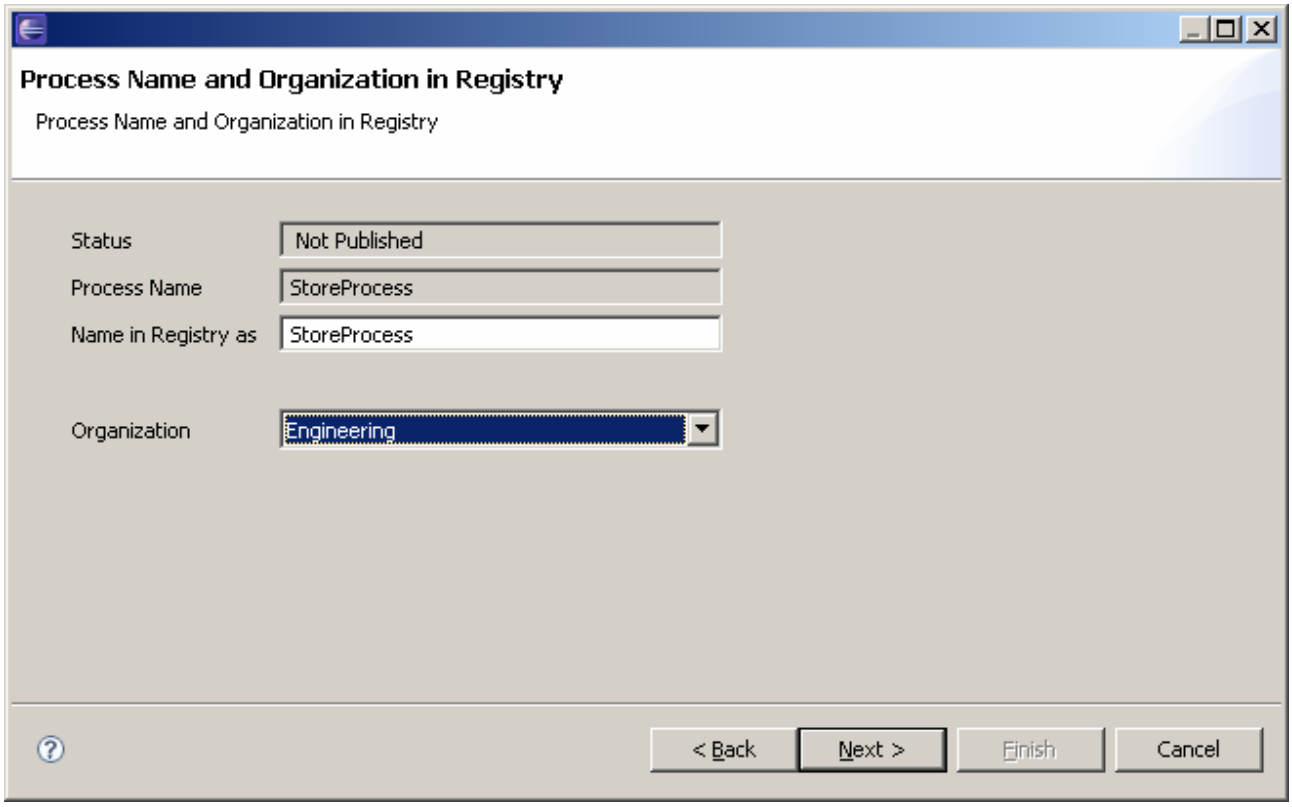
After you've loaded the data in the registry, you can configure the processes published in the metadabase in registry using the **Service Registry Button.**

Operations regarding the service registry, are done using a very simple, three wizard step:

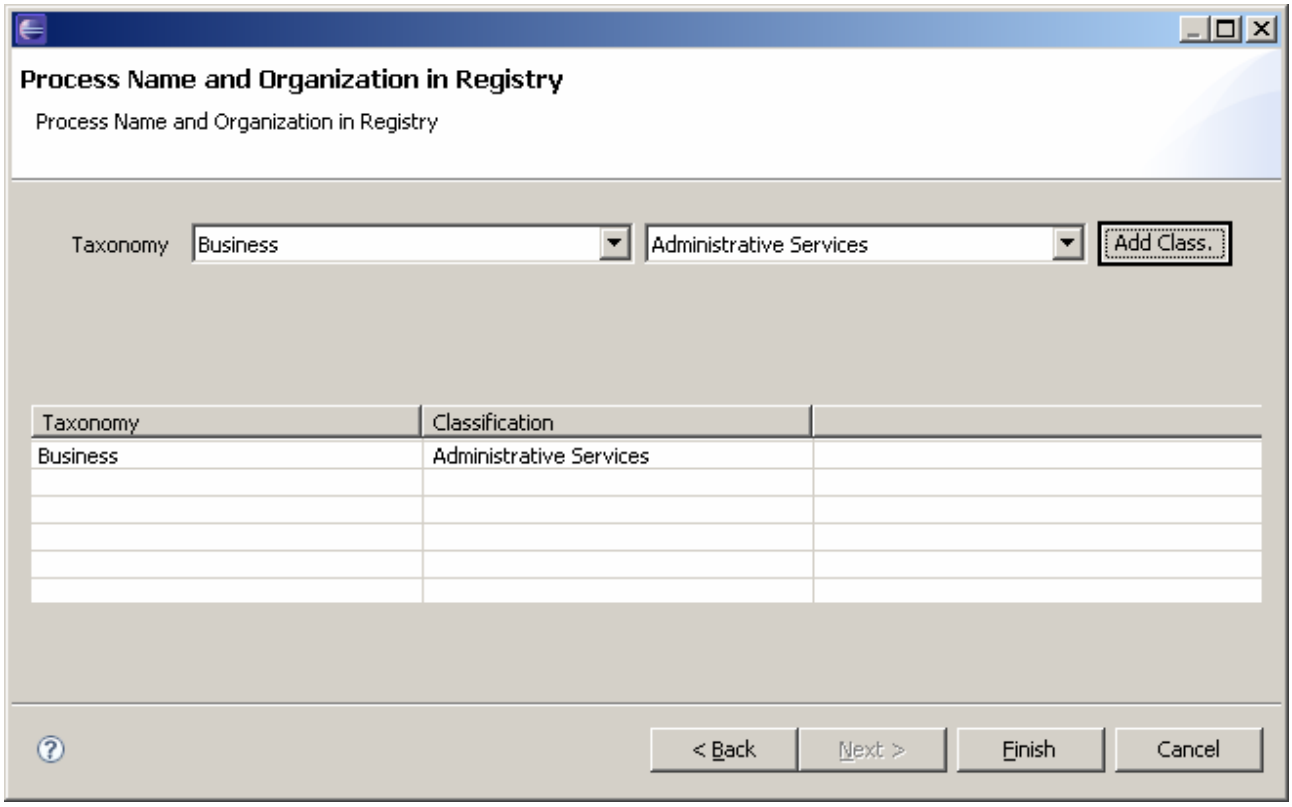
1. First aff all you need to choose a process from the list of processes published in the database.



2. Then if you selected a process that is not published in the database, anf go on to the wizard with the next button you're required to chose an "Organization" owner of your process and the name of the process in the Business Registry. If the selected process is alredy published you could do not change the information in this dialog page.



3. At this point you're required to classify your process according using taxonomy and taxonomy values. If you want to remove a classification element, simply select the element in the table and delete with "Del" key on keyboard. If your process is already published and classified, this steps simply means as a reclassification.



4. At this point you could push the finish button to terminate the wizard and save the data: