Spagic Events Guide

Author: Andrea Zoppello

Document Goal

In this document we will focus on the feature of Spagic to handle processes that need to handle external events.

Version History

Version/Release n° :	1.0	**Date**	July 17, 2008
Description/Modifications:	First Release (English version)		

1 Needs for Event Handling

When dealing with real use case, there's often the need to model situation where a process instance must wait for an external event before to proceed with it's execution.
In particular external events could be divided in:

- Event performed by Humans (Human Activities), this means that a human person perform some task that let the process to continue (for example fulfill and submit a form)
- Events generated automatically by a system when something occurs (a file has arrived in a folder, a mail has been received and so on)

1.1 Spagic Event Handling Features

For the reason explained in previous paragraph, in the version 2.2 of Spagic we've introduced some basic event handling mechanism and components that enable Spagic users to model process with event handling capabilities.

To enable this we've basically introduced:

1. **Wait Components**, this let the process analyst designer to model that a particular process must wait for an event before to proceed with its execution. This components let's the modeler to save some data associated to the process when this is in wait status.
2. **External Event Handlers Components,** to model the situation where a process with a wait component will be resumed by particular event.
3. **Mechanism to automatically resume or kill process waiting for events.**
4. **Extended the metadabase to handle data associated with a process waiting status.**

With the feature above in Spagic it's now possible to handle the following use case:

- Process waiting for human task event
- Process waiting for external event handling

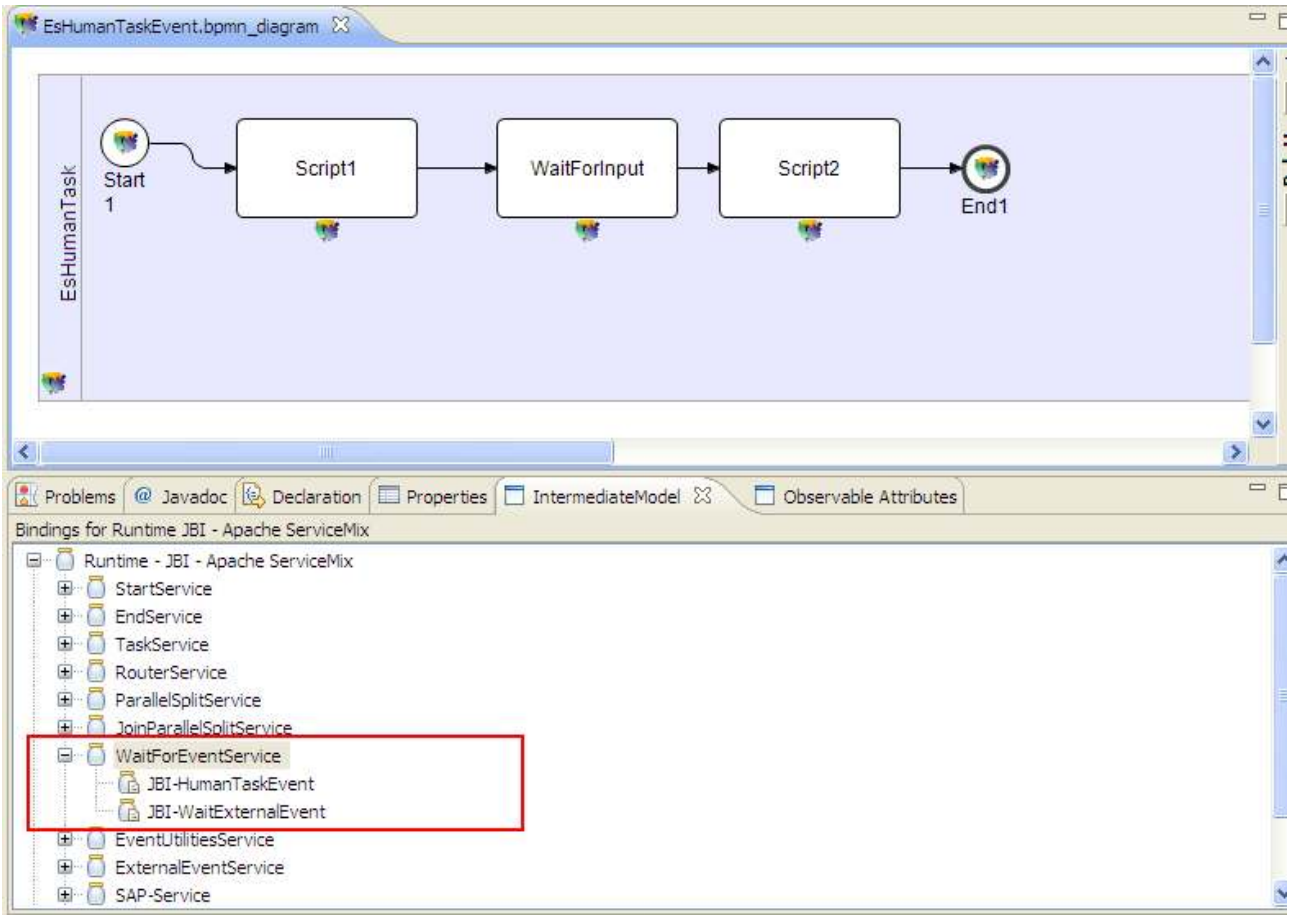
In the following of this document we'll explain in detail how to model and handle this type of process with Spagic studio and at the end we'll explain the chron job that handle expired process instances.

2 Spagic Processes With Human Task Events (Basic Feature)

2.1 Modeling Processes with Human Task Events

To model a process that needs to wait for a human task event follow these steps:

1. Model a process with a wait task
2. Associate to it the service called **HumanTaskEvent** in the **WaitForEventService** Category as you do with all other services in Spagic studio and fill the property for HumanTaskEvent



When you've associated the HumanTaskEvent Service to the task, you need to fill some properties, like the Expiration Policy, the action to do when the process instance expires, and you could associate a set of data, related to the wait instance, extracted with a set of XPath rules.

The **Expiration Policy** could be:

- **Relative** to the time in which the process instance enter in wait status (AfterTimeInterval), in that case you need to specify the delta time to calculate the expiration policy dynamically

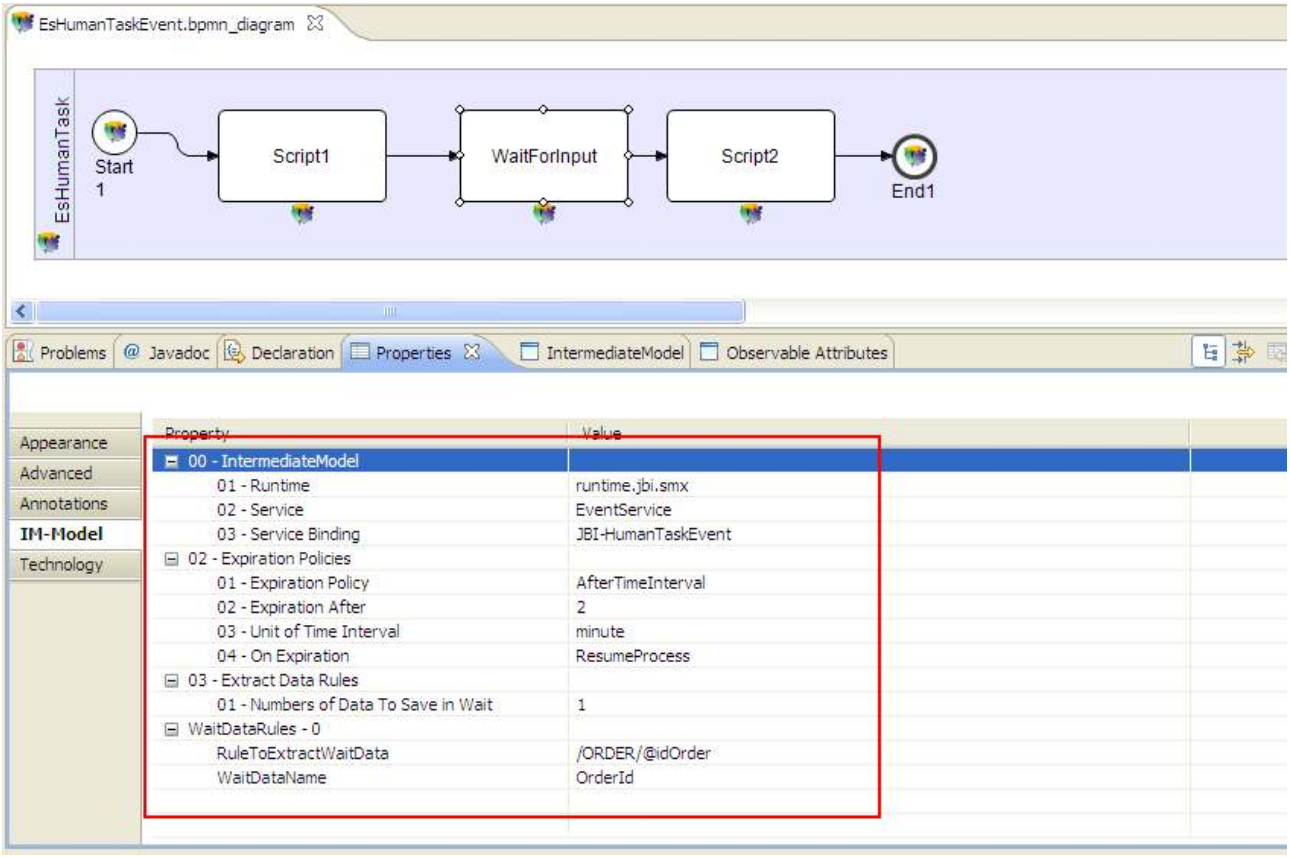
- **Absolute**, in that case you need to give a fixed date in which the process expire in the format required (DD-MM-YYYY HH24:mm:ss)

When a process expire Spagic take could do automatically two type of things:

- **Resume the process instance**, in that case the process execution continue as if the event occurs
- **Kill the process instance**, in that case the process instance data is killed and the process does not continue the execution.

You could associate a set **of data to a waiting process instance.**

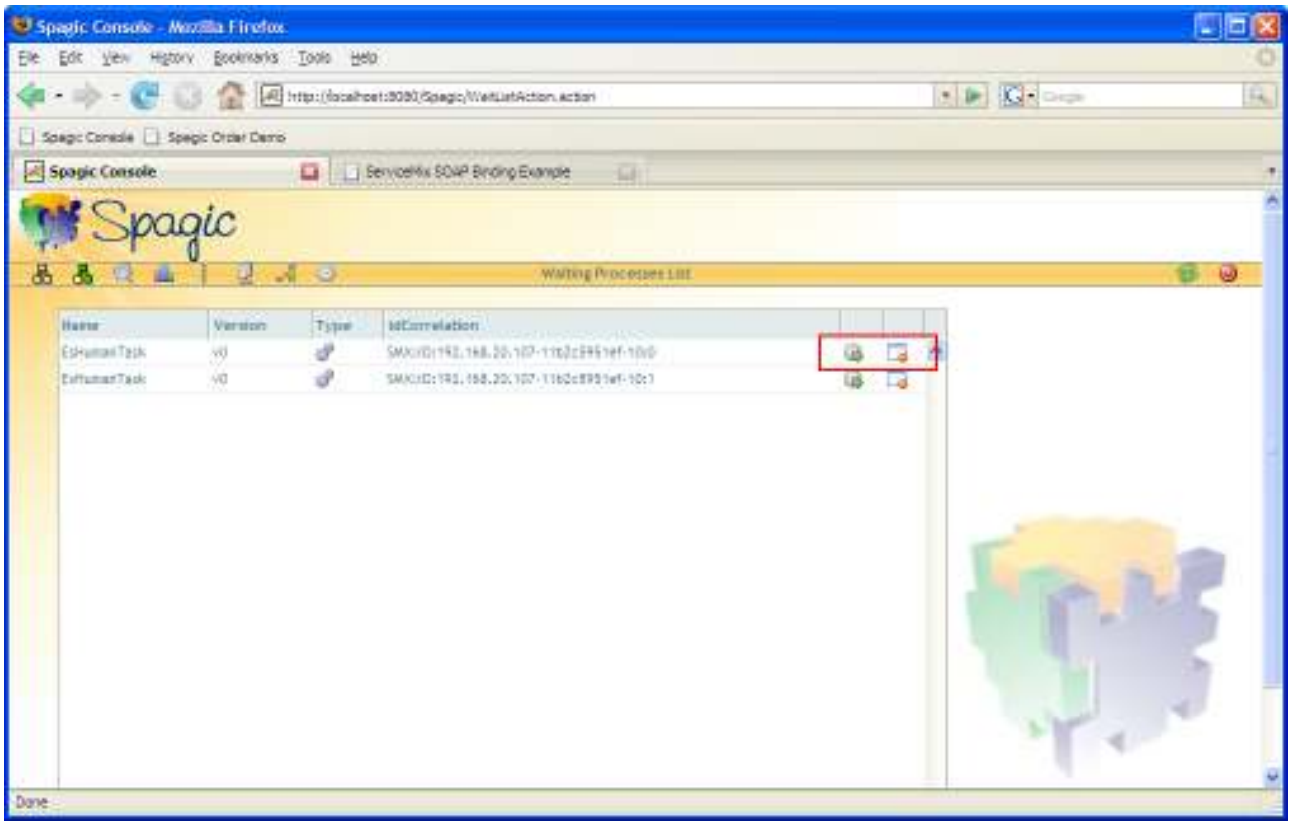
In the case of human task event this "wait instance data" are not very important because in the case of human task event it's supposed that the human action is directed to a particular process instance so we don't need to identify the instances to unlock.



2.2 Generating Human Task Event with Spagic Console

Actually there's only one type of human task event handled by Spagic, and this is the event generated on a blocked process instance with an explicit request to unlock that a user could provide on the Spagic console.

In particular in the Spagic console you've the list of waiting process and you've two buttons, the first one generate the Event that will resume the process instance, the second one will kill it.



- This is a very basic feature, but it has some advantage

3 Spagic Waiting For External Events

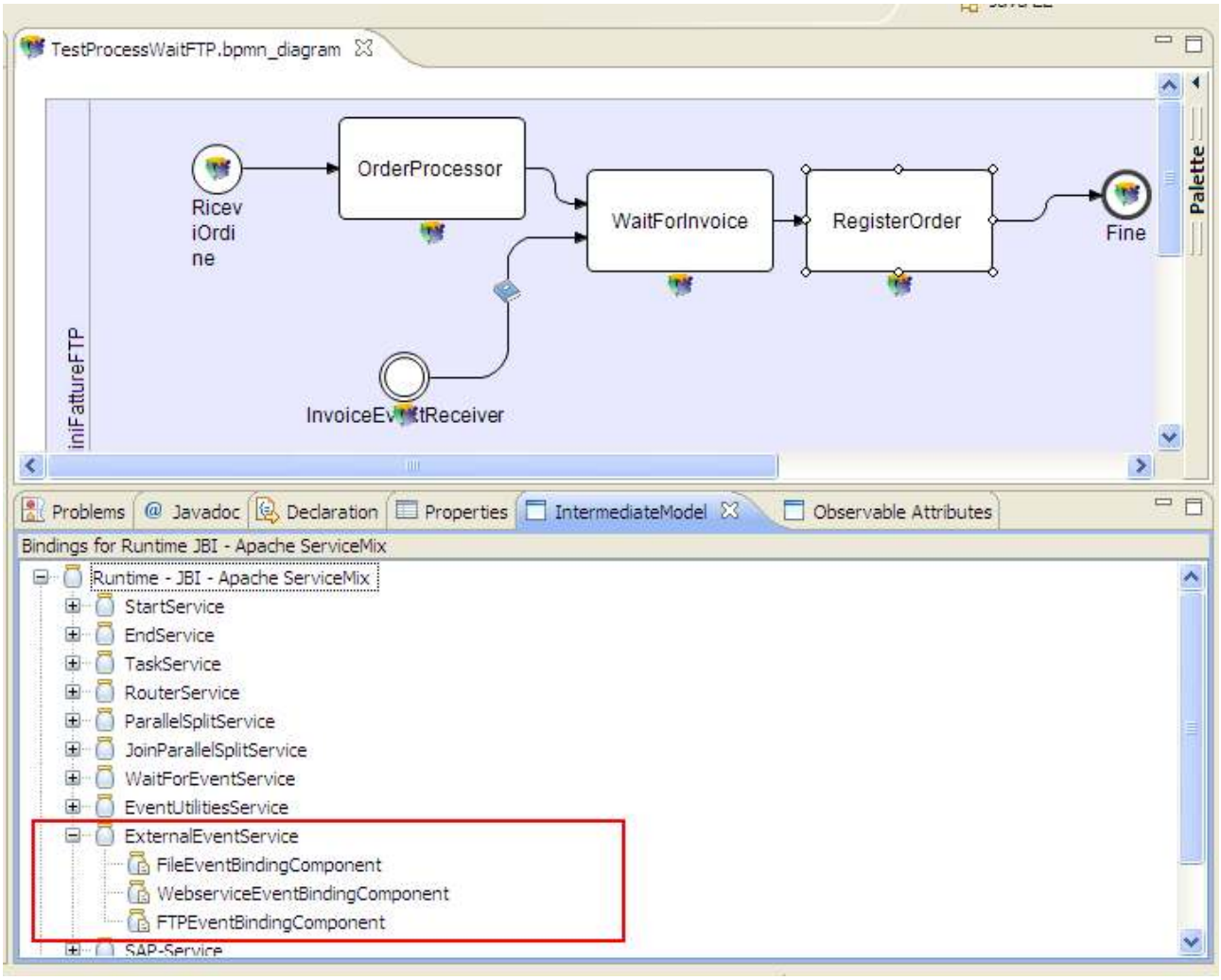
In the previous section we have seen how to model process that need to wait for Human Task Event. In this one we'll explore the case in which the process wait for event, automatically generated by external system.

The modeling phase is similar to the previous one, but in this case we must:

1. Model a process with a wait task (This is the same as the previous one)

2. Associate to it the service called **WaitForExternalEven** in the **WaitForEventService** Category as you do with all other services in Spagic studio and fill the property for HumanTaskEvent.

3. Provide to the process an **ExternalEventHandler** to let the process to receive external event. The event handler are modeled with **BPMN Intermediate Event Element** annotated with the service that act as event handler, and connect it to the wait activity, in that way we've expressed in very elegant way that event received by event handler will resume the process instance in waiting status.



4. Provide a set of rule, that let the event handler to match the event arrived with one or more wait instances. This further step is necessary because when a human task event occurs, it contains the information about its

related instances, instead when an external event occurs you've no information about process instances and you need a way to bind the data contained in the event to the instances to resume.

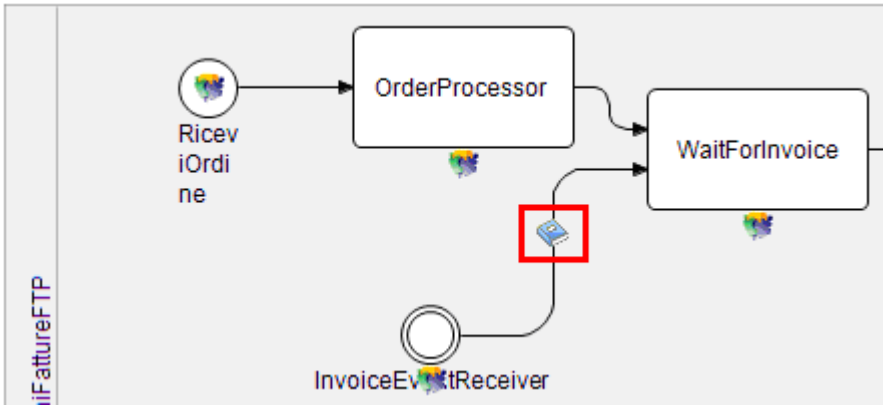
To be more clear, if you resume an order selecting it with Spagic console, you're sending to Spagic the information that you want to resume a precise instance of the order process, but if you are modeling your process as it wait the invoice in a particular folder, you need a way to match the event data (the invoice data) with the wait instance data.

For this reason, the event handler is responsible to:

a. Normalize the event to **Normalized Event Form (A set of named Header, and a xml payload)**
b. Select all the waiting instances candidates for the type of event arrived
c. From the candidate instances evaluate a set of rule, to select one or more that will be resumed, the rules are expressed matching "waiting instance data" with event attributes, or XPath rule on event data.

Taking the order example of the picture you simply save the OrderId as "wait instance data" extracted on waiting activities task, and the you configure an event matching rule saying that when an invoice file arrive to a particular directory, it will match the instance if the OrderId match with "/INVOICE/Order/@orderid" on event payload.

To put event matching rule you must annotate the link between event handler and waiting activity, using the "Matching Event Rules" from the available expression in Intermediate Model View.



5. As in the case of Human Task Event, you could choose the Expiration Policy, and the action to do when a waiting instance expires.

3.1 Available External Event Handlers

As we've seen in the previous section when you model a process with an activity waiting for external event, you must configure an external event handler.

In Spagic 2.2 there at the moment there are three type of external event handler that are enough to cover most of the use cases:

- **File Event Handler (FileEventBindingComponent).** This is to handle the event when one or more file becomes available in a particular folder, this event hander is very similar to a standard file binding component, where in addition is possible to configure some filter rules about the files to accept. The configuration properties are the same of a File Input Binding Component.

- **FTP Event Handler (FTPEventBindingComponent).** To handle the event when one or more file becomes available to a particular ftp server (in specified folder). The behavior is very similar to File Event Handler you could configure filter rules on file to accept and the configuration is the same as a FTP Input Binding Component.

- **WebService Event Handler (WebServiceEventBindingComponent).** This event handler has to be used in all the situation where external system, will use a webservice call to generate events. In that case the soap payload will constitute the event body.