Java CAPS Quick Note 003
JBI-based JMS Publisher and Subscriber

Michael.Czapski@sun.com

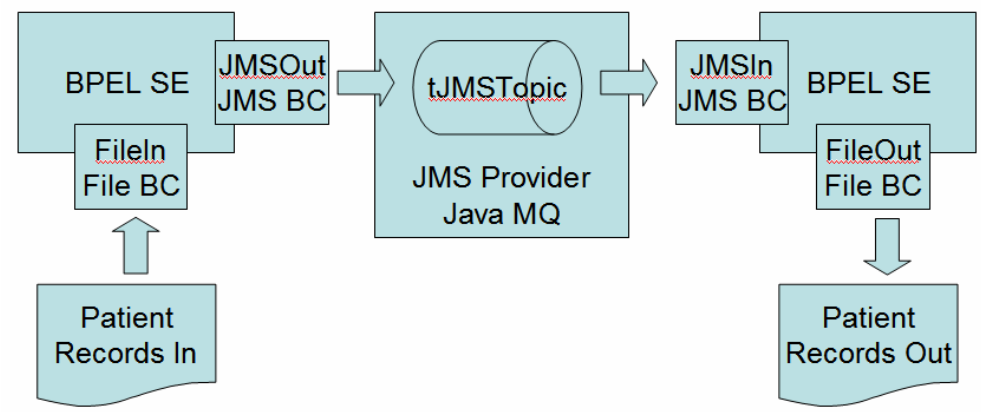
30th of June 2009

This Quick Note discusses a simple solution to the use case provided by Leonard Barkely. The question goes like this:

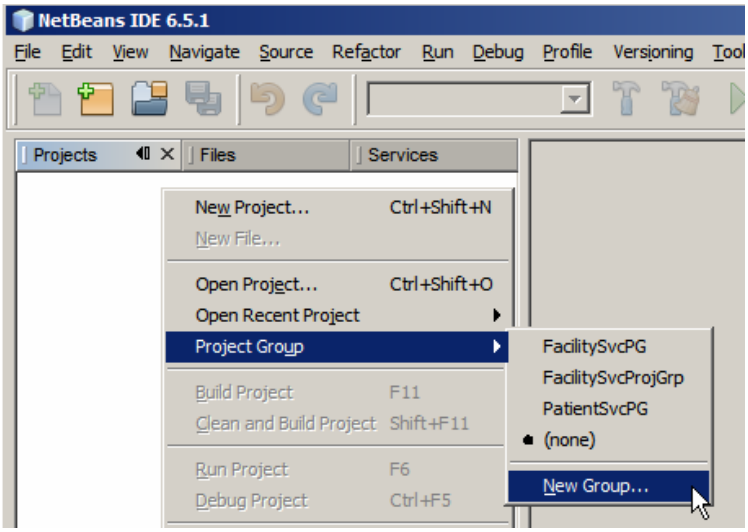
> "I dont have any idea how to implement BPEL process but the BPEL deployed as a subscriber of a topic. usually I implement the BPEL process and deployed it as web service."

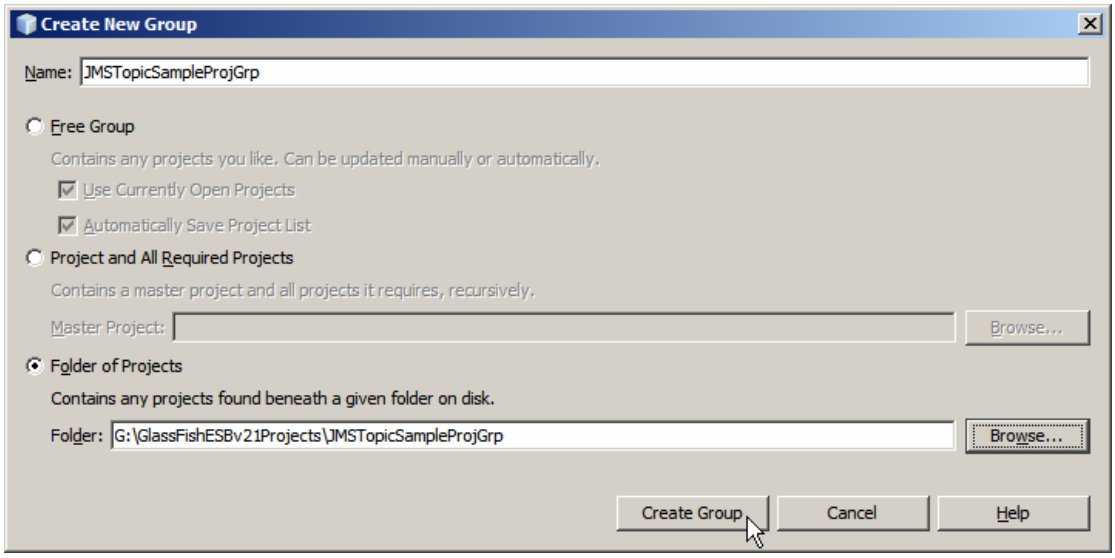
Let's produce a simple GlassFish ESB v2.1-based solution which reads a file, sends its content to a JMS Topic and another simple GlassFish ESB v2.1-based solution which subscribes to the same JMS Topic, receives the message and writes it to a file. Both solutions will use BPEL to implement the simple logic, though it is possible to implement both solutions without BPEL.

Solution schematic looks like this.

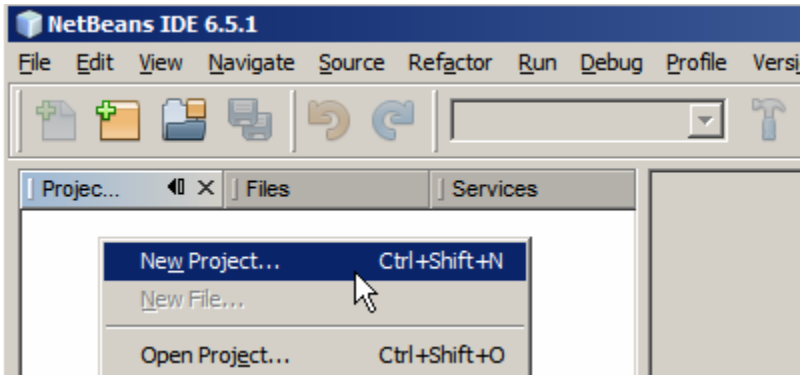


Let's create a project group, JMSTopicSampleProjGrp.

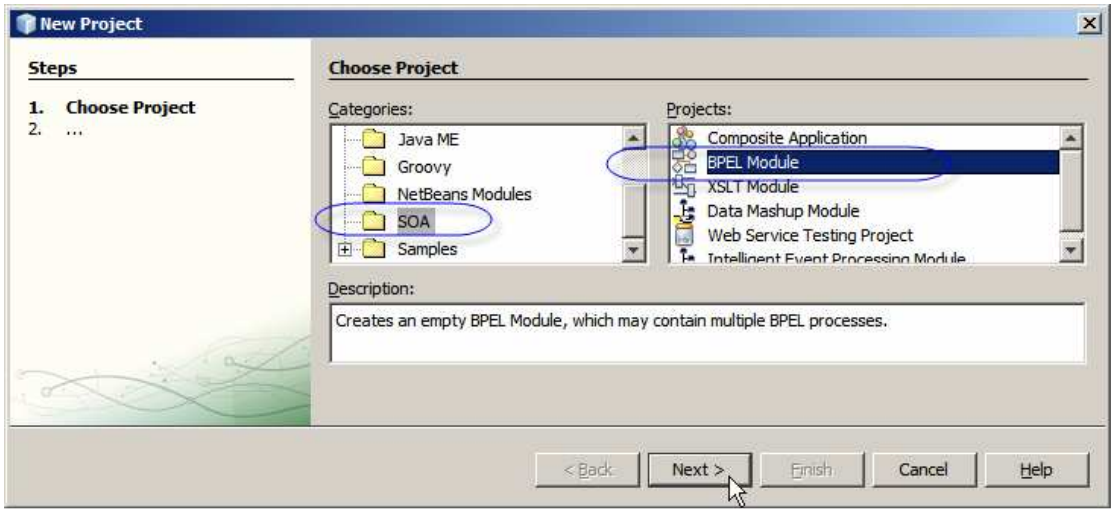




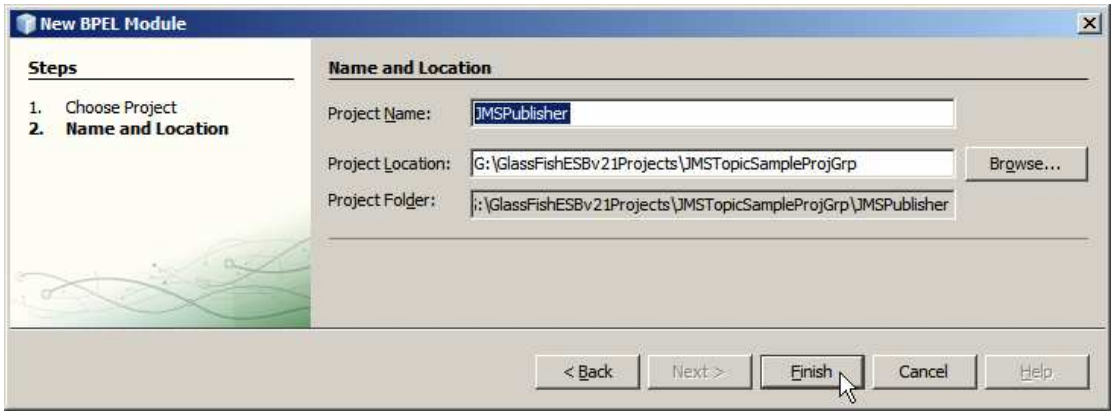
Let's create a "New Project …"



Project type will be SOA -> BPEL Module

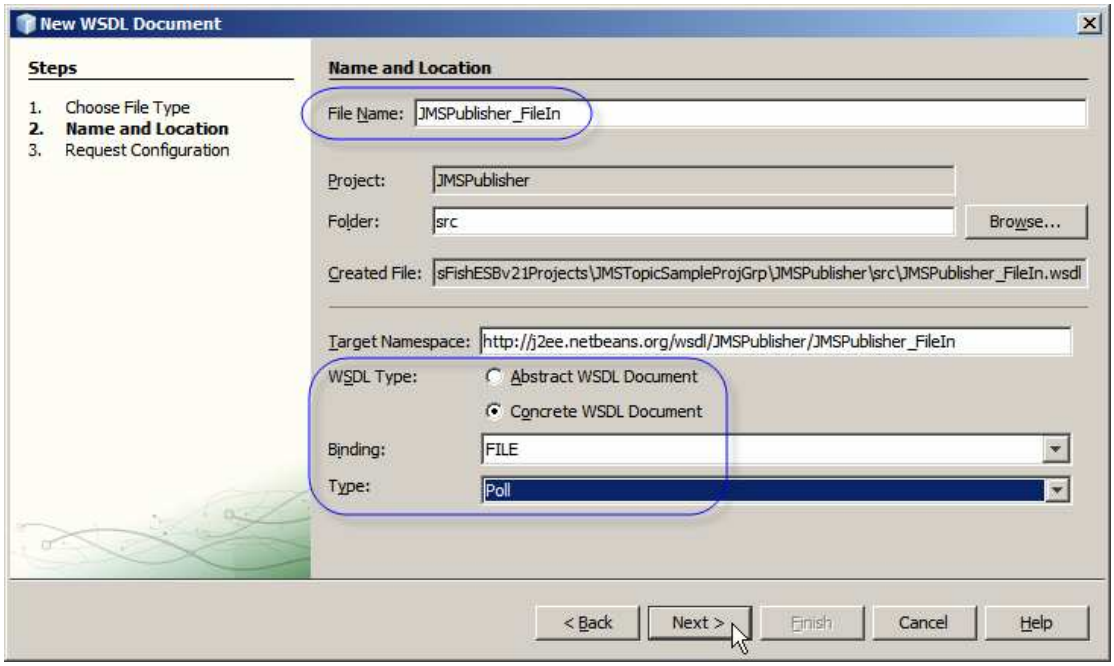


Project name will be JMSPublisher.

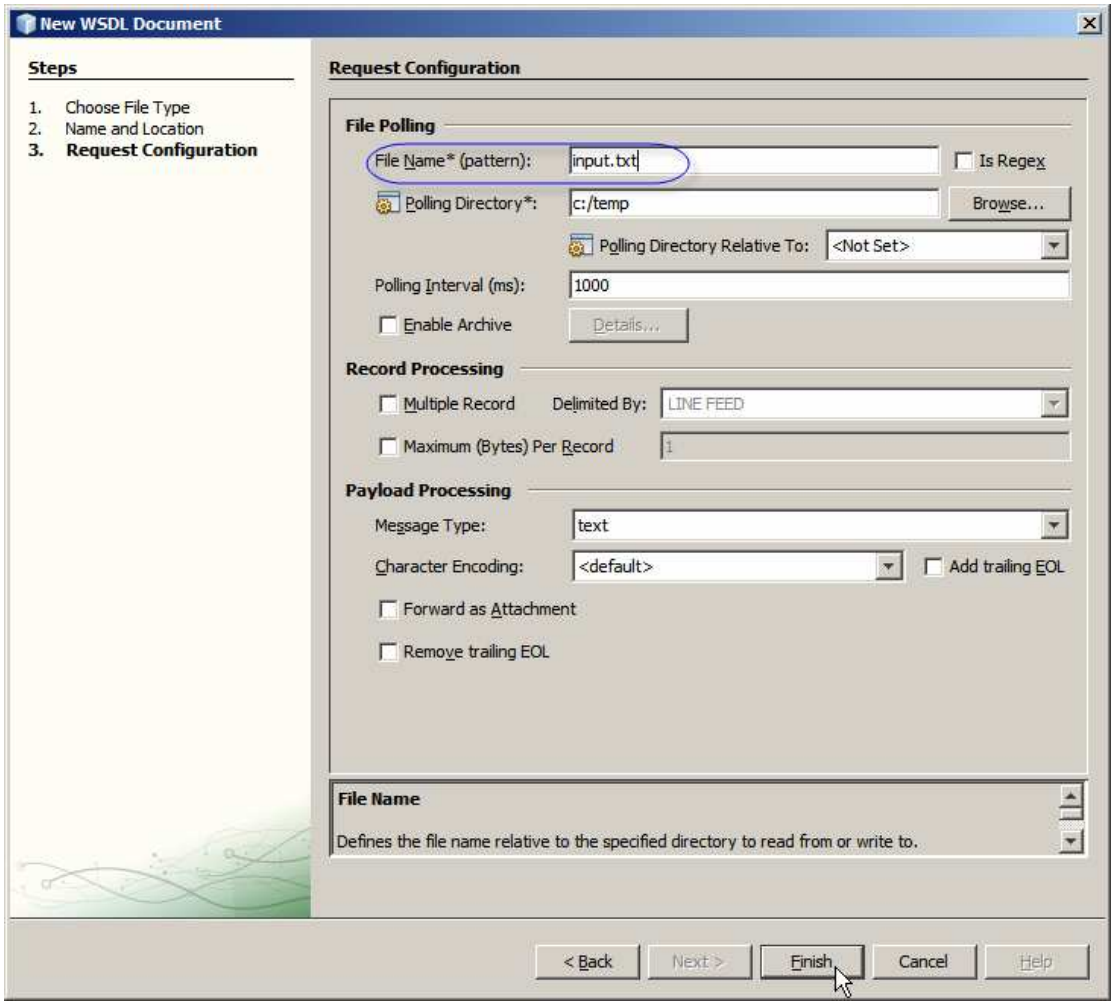


Right-click Process Files and choose “New” -> “WSDL Document …”

We need to read a file so we need a concrete WSDL to define file name, location, polling interval, etc.. Name the WSDL JMSPublisher_FileIn, choose Concrete, choose FILE binding, choose Poll Type.



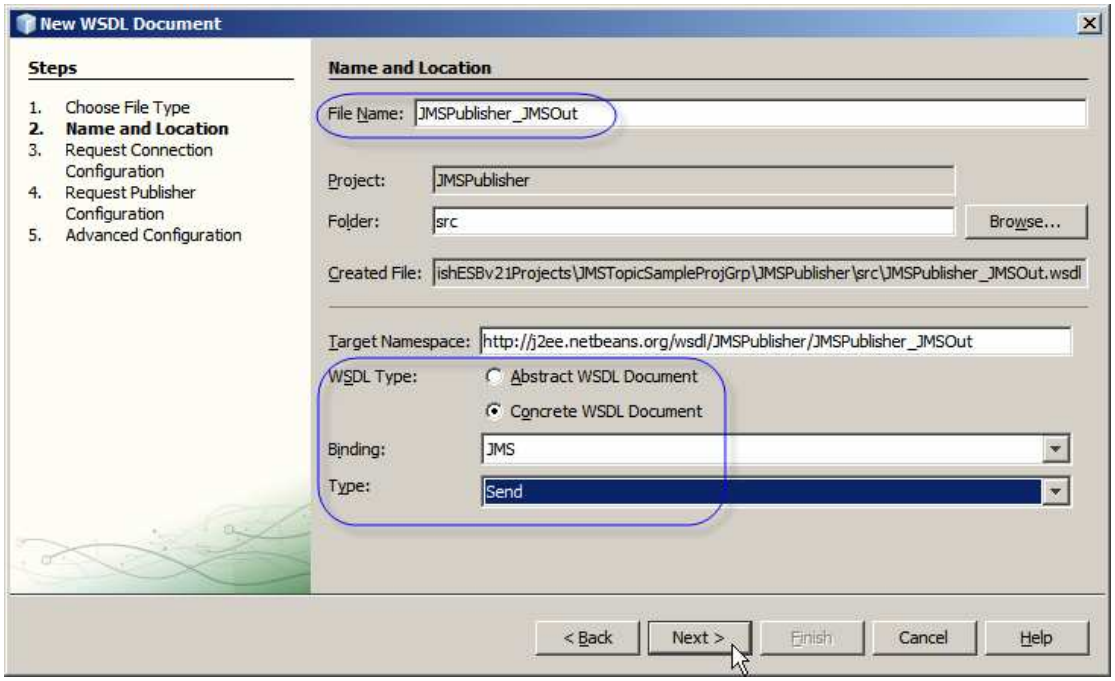
Change name of the file to input.txt and accept other defaults (if you are on Windows – if not, change directory to one of your own choosing).



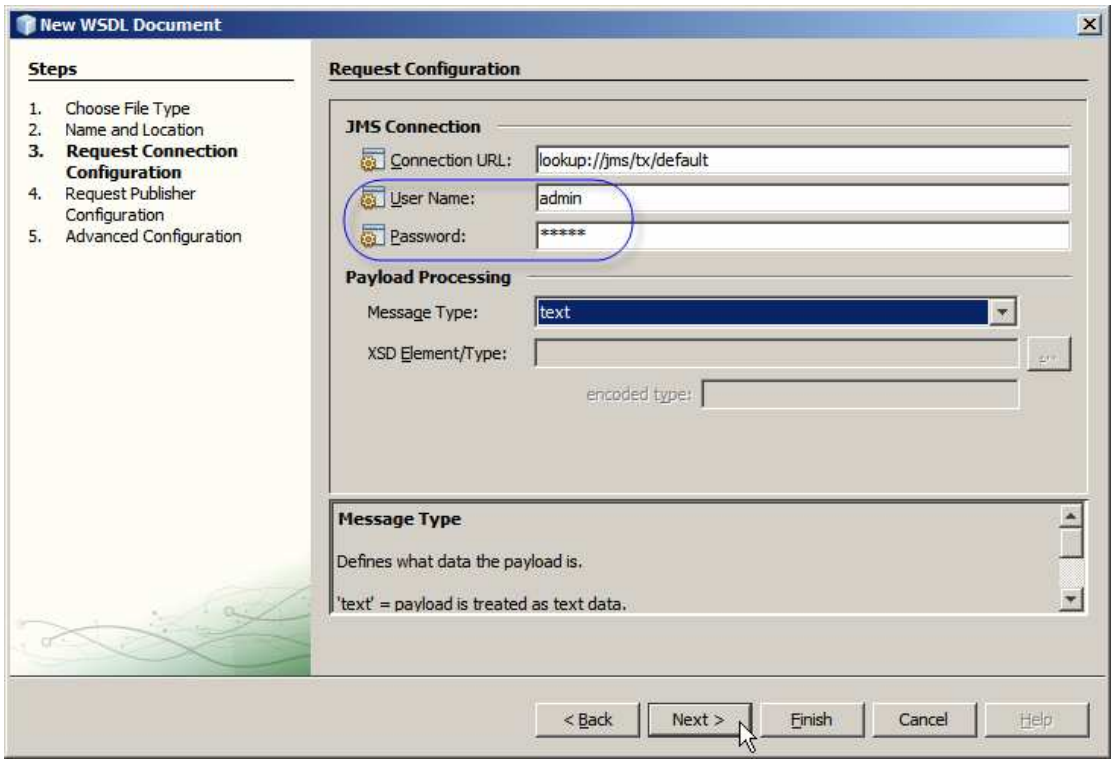
This produces a pre-configure File BC which will poll for a file c:\temp\input.txt every second.

Let's now create a WSDL to configure the JMS BC.

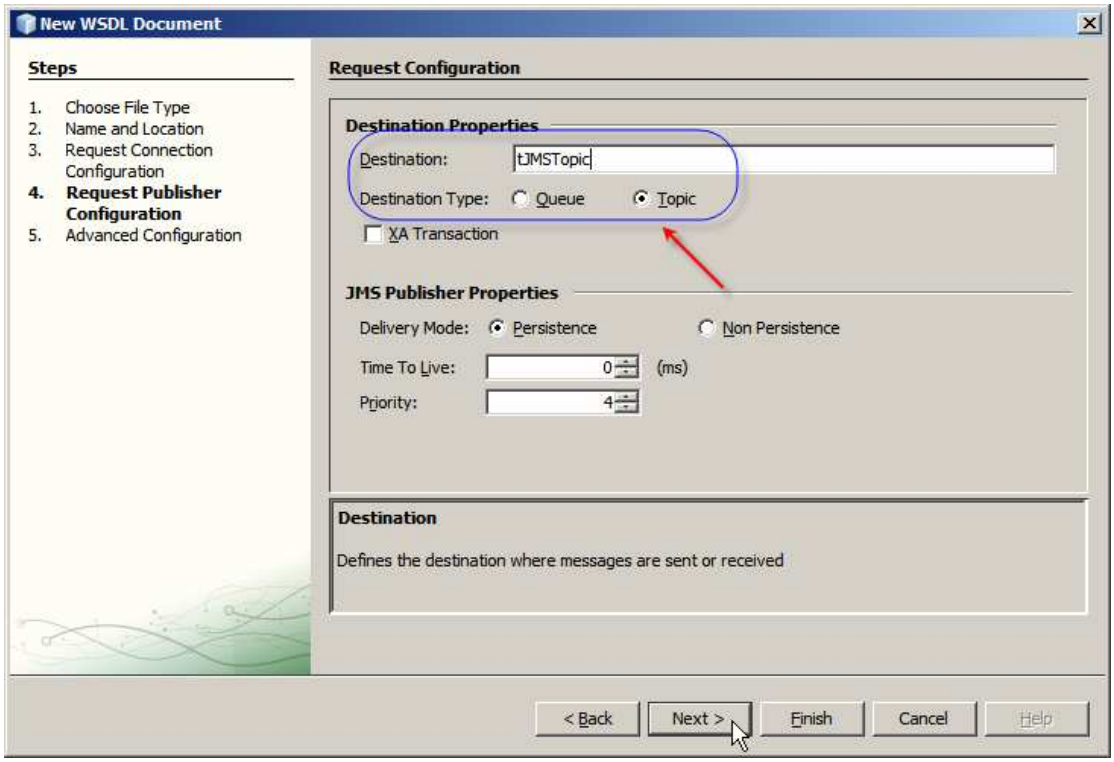
Right-click Process files, choose New -> WSDL Document. Name the WSDL JMSPublisher_JMSOut, choose Concrete WSDL, choose JMS Binding, choose Send Type.



Set the username and password – by default these will be admin and admin.

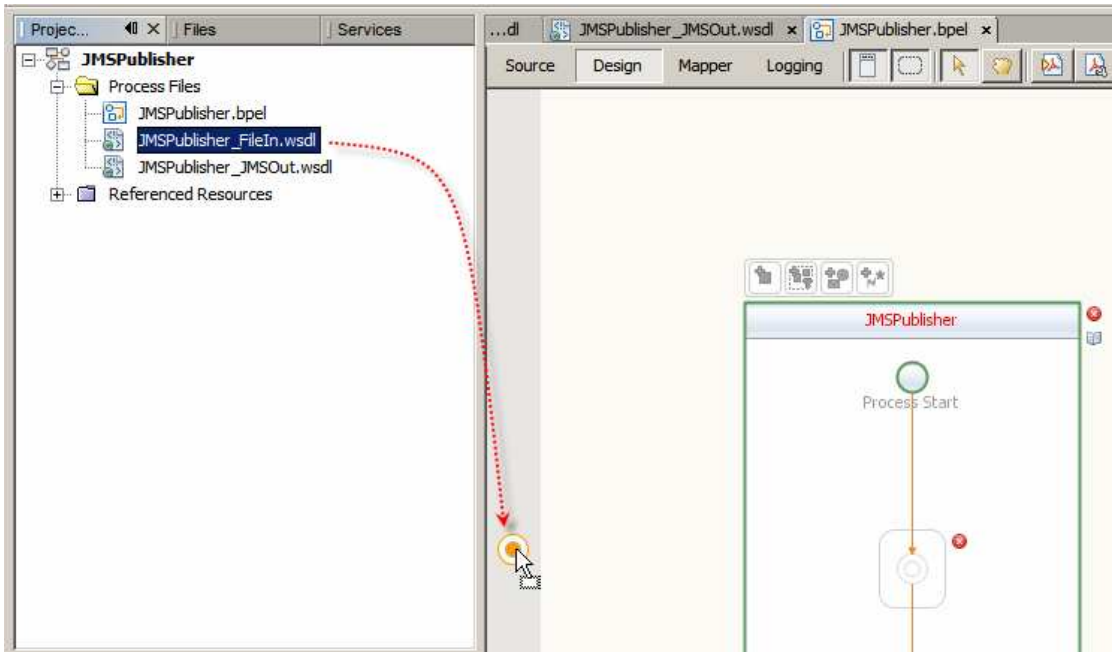


Name the topic tJMSTopic and make sure the Topic radio button is selected.



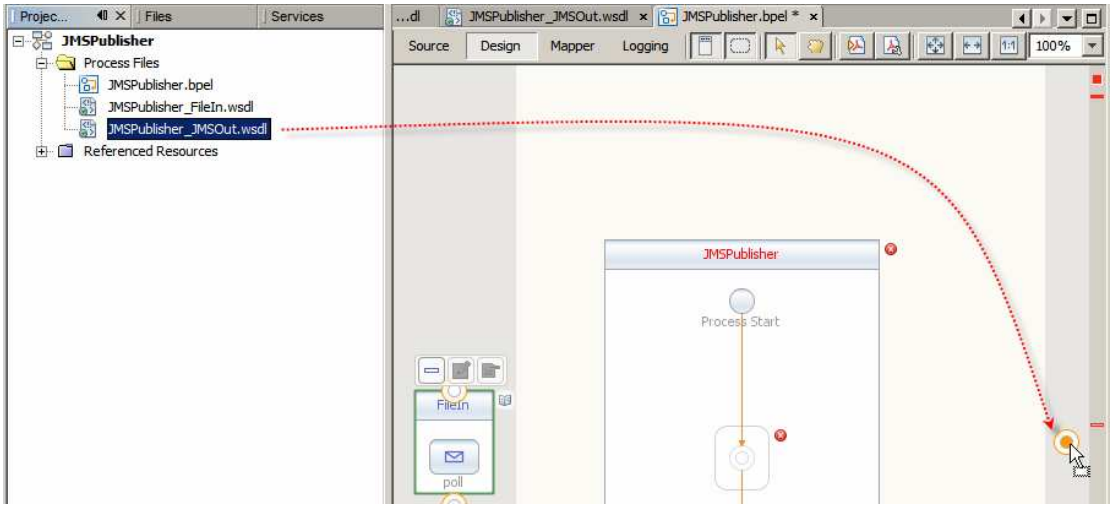
Finish.

Switch to your BPEL process and drag the JMSPublisher_FileIn WSDL onto the target marker in the left-had swim line.



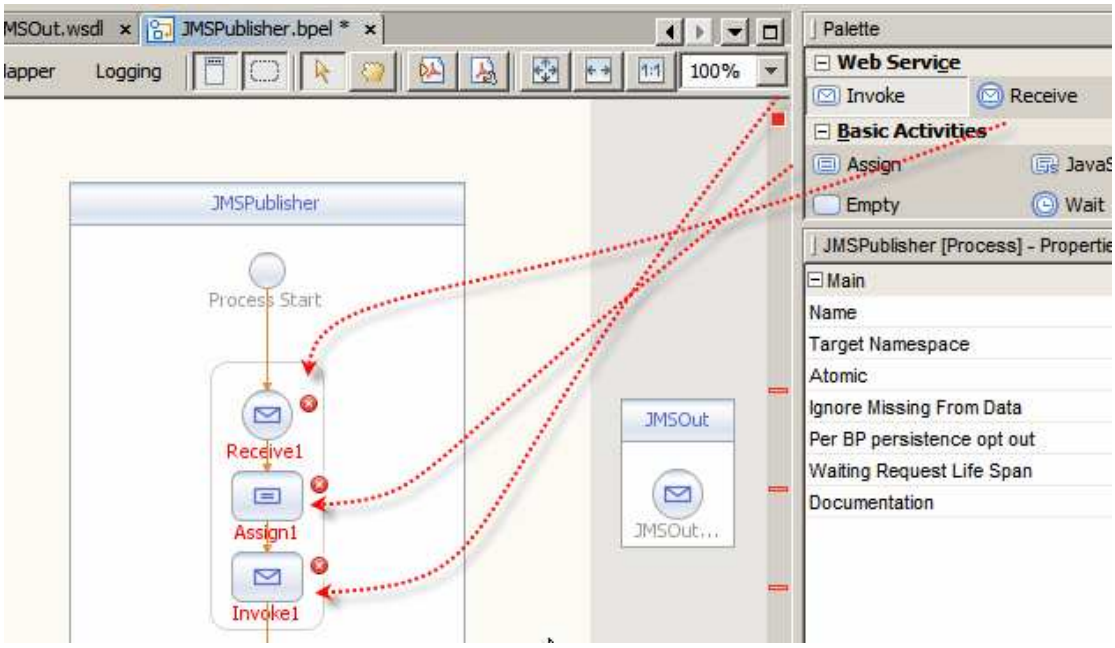
Rename the partner link to FileIn

Drag the JMSPublisher_JMSOut WSDL onto the target marker in the righ-hand swim line.

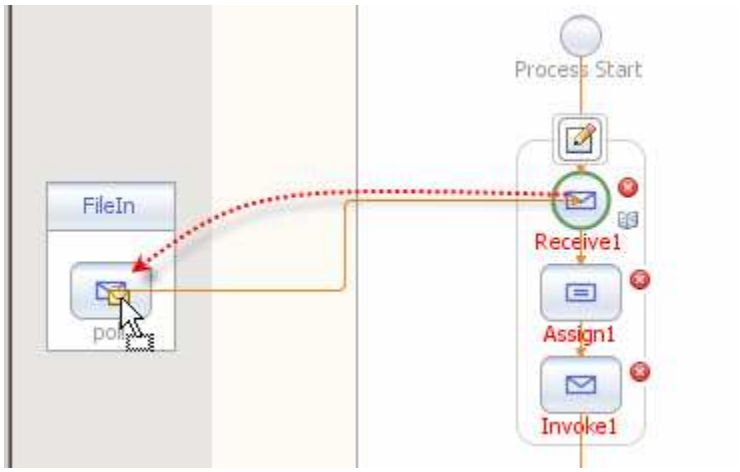


Rename partner link to JMSOut

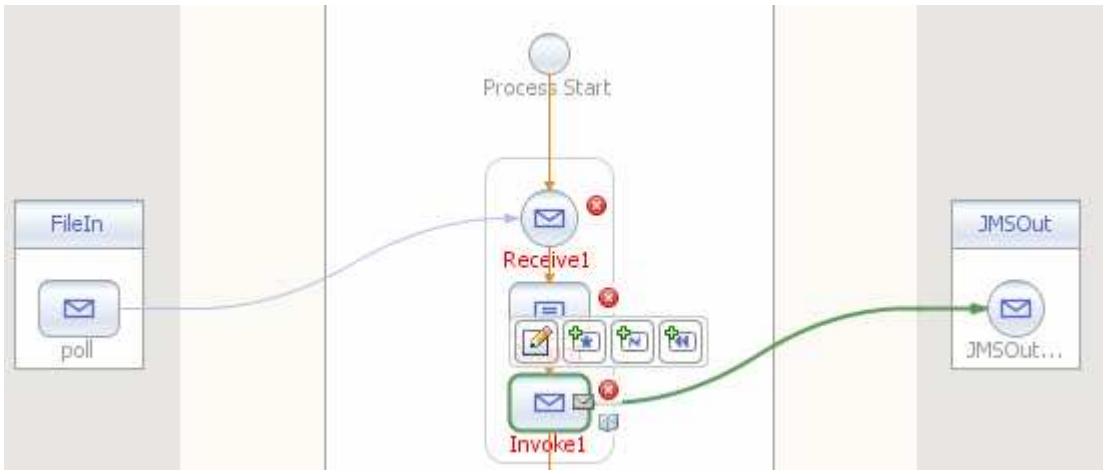
Drag Receive, Assign and Invoke activities onto the target markers inside the JMSPublisher process scope.



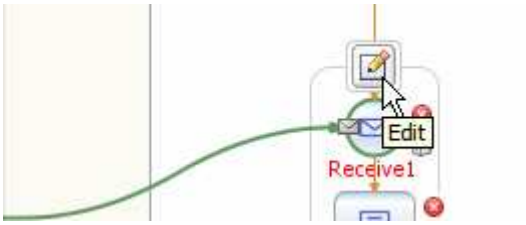
Connect Receive to FileIn partner link.



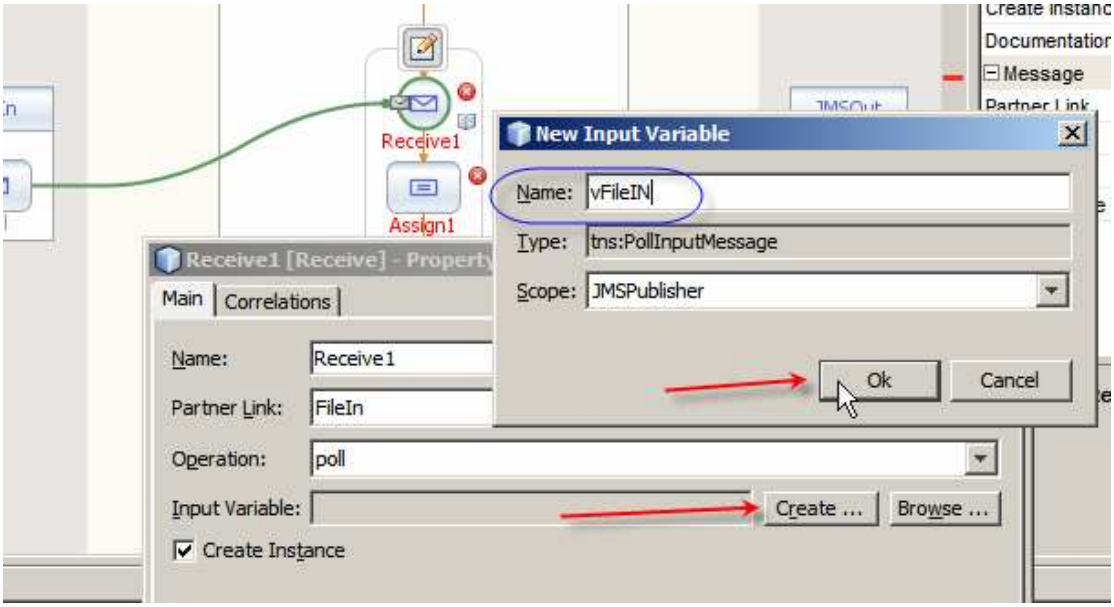
Connect Invoke to JSMOut partner link



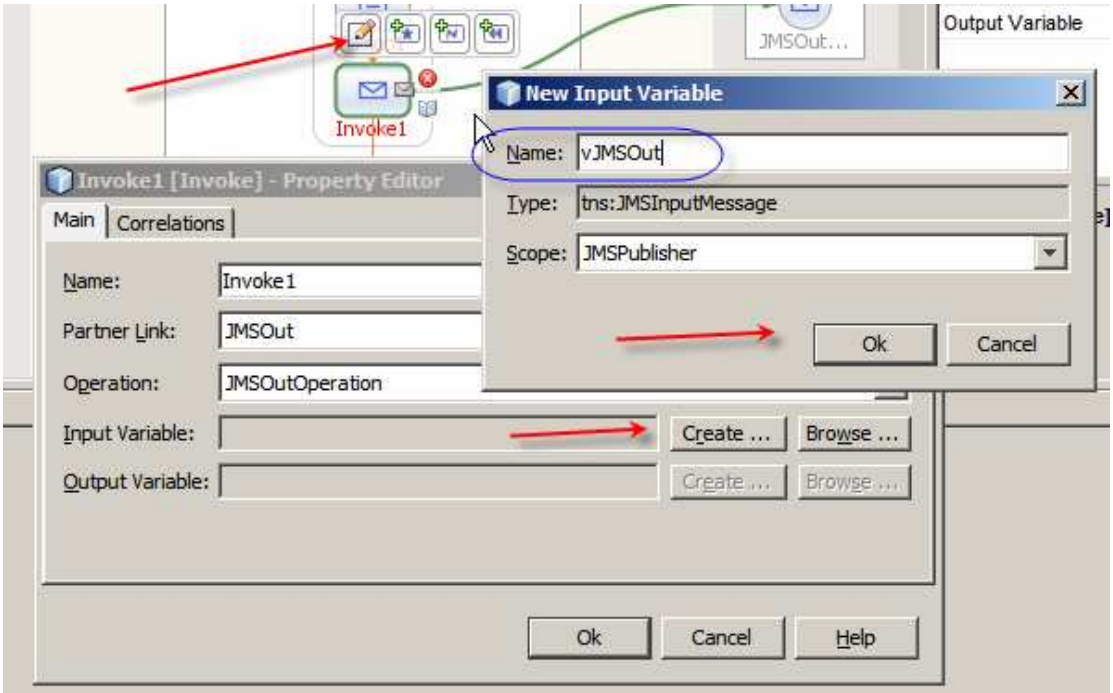
Edit Receive



Create a variable vFaileIN

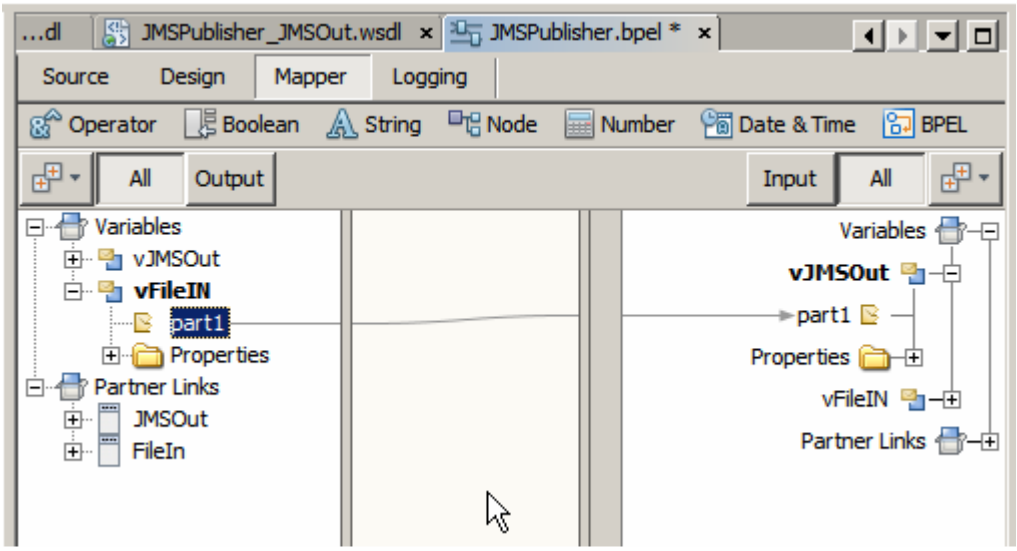


Edit Invoke, create a variable vJMSOut

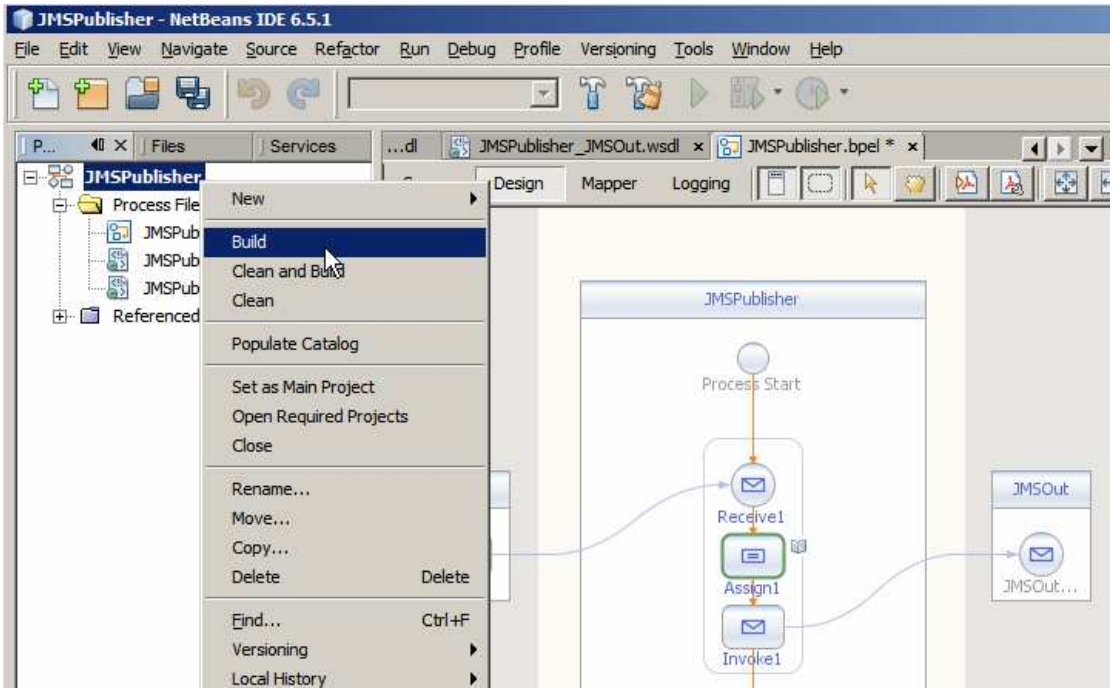


Double-click Assign to switch to Mapper view

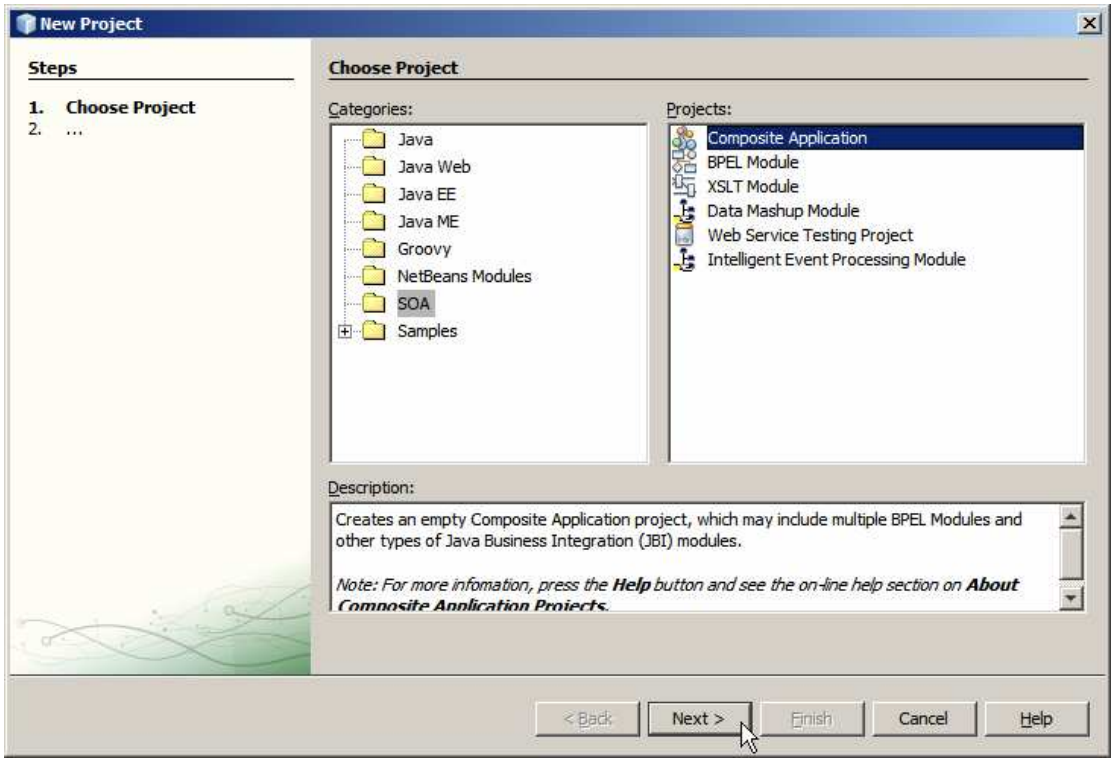
Expand variable trees and map vFileIN->part1 to vJMSOut->part1



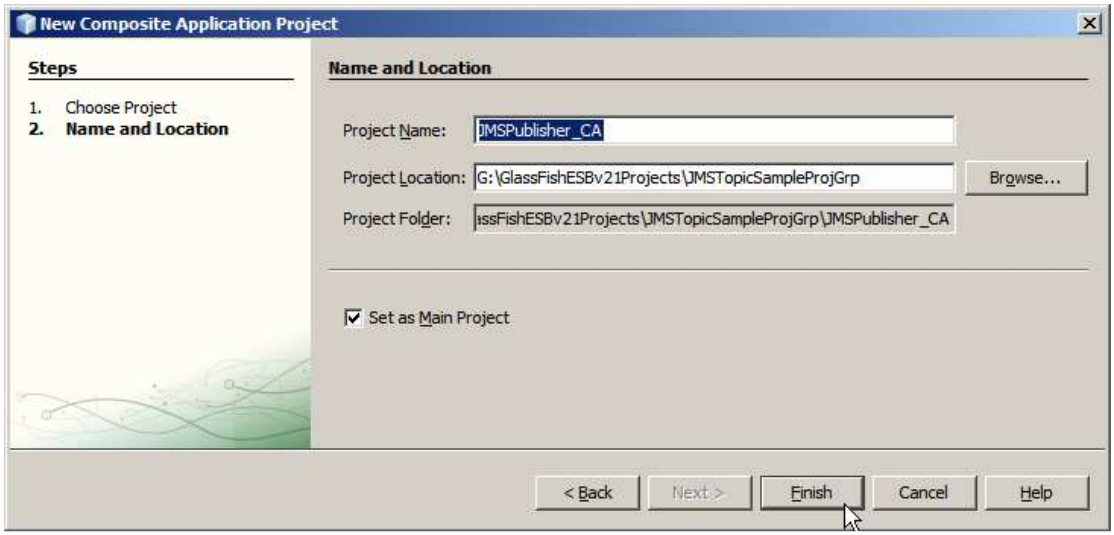
Switch back to Design view. Right-click project name and choose build.



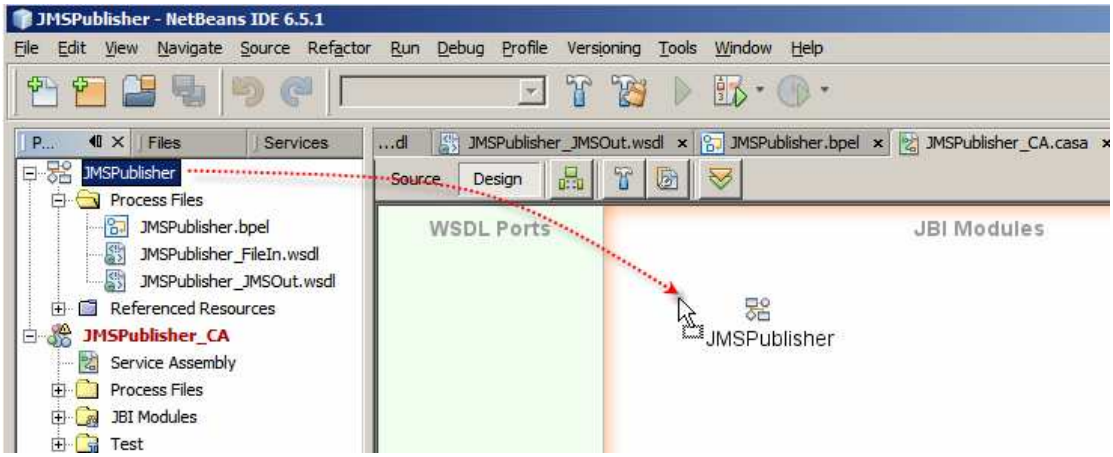
Create a New Project …, SOA -> Composite Application.



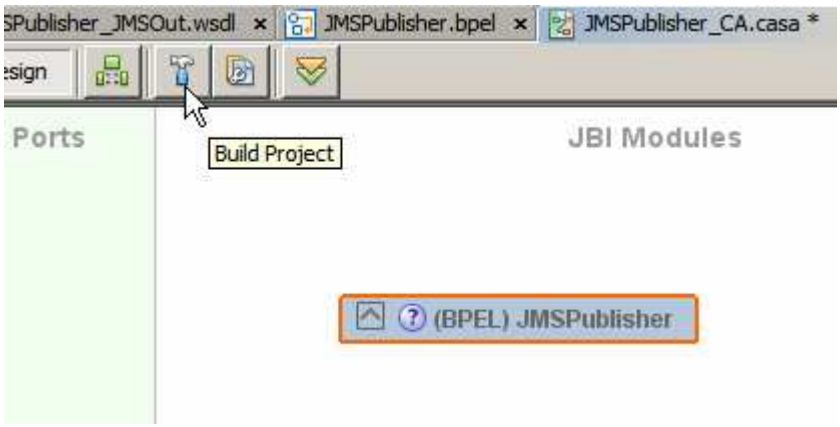
Name the project JMSPublisher_CA.



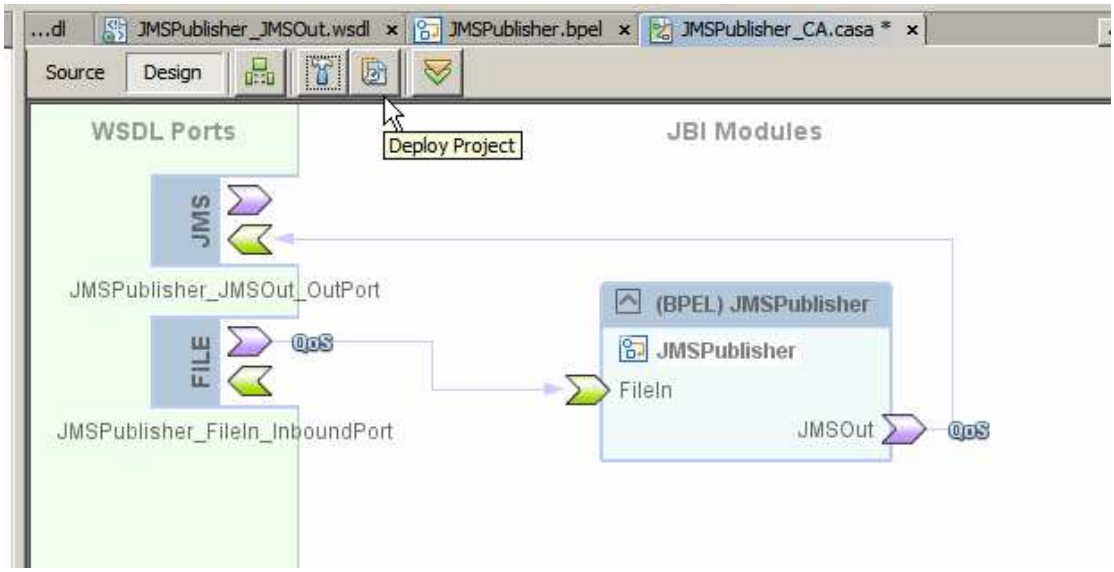
Drag the project JMSPublisher onto the CASA canvas.



Click Build



Click Deploy

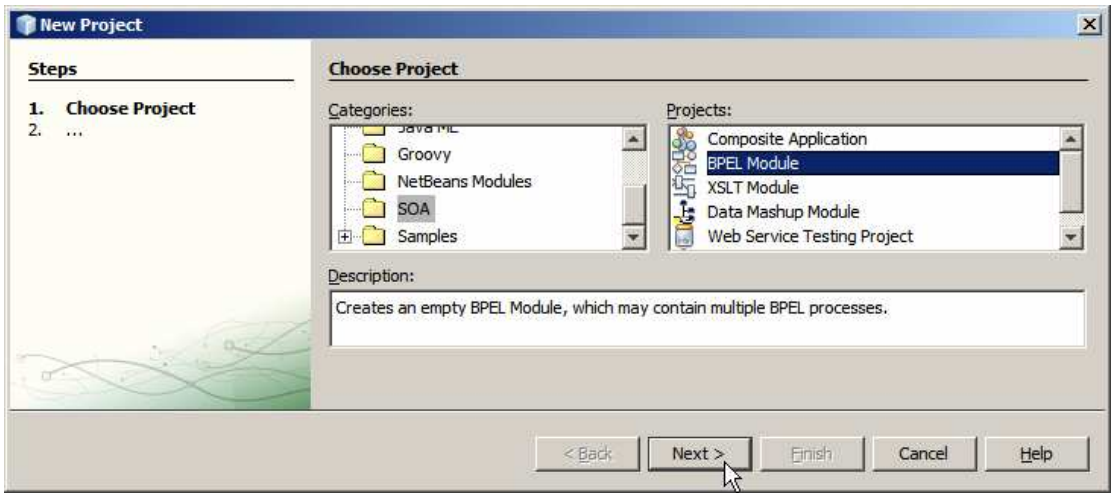


If all went well (you see BUILD SUCCESSFUL) the JMS Publisher project is ready to roll. In the absence of tools to watch messages in a JMS Topic, which would take several pages to describe how to get, install and use, we must wait for the JMS

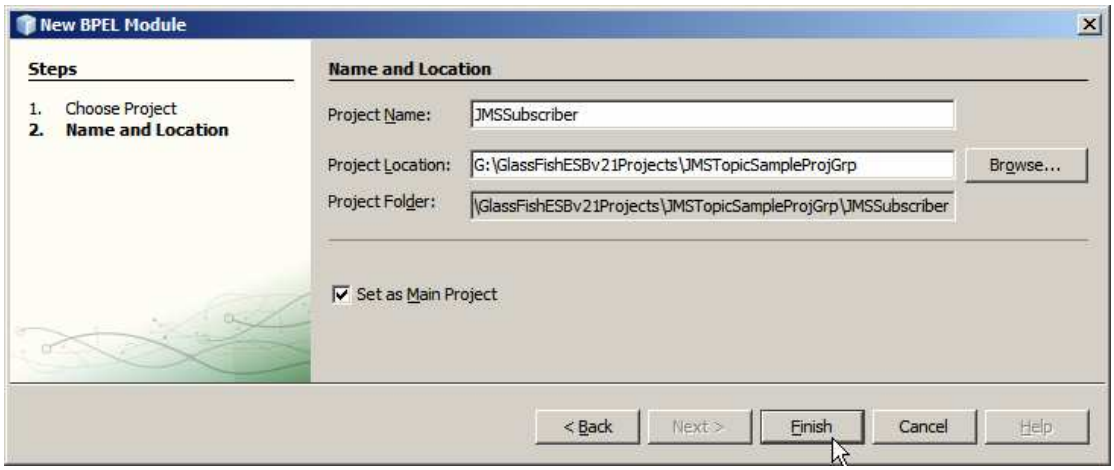
Subscriber project to be developed and deployed before we can test the solution just developed.

Let's develop the JMS Subscriber project.

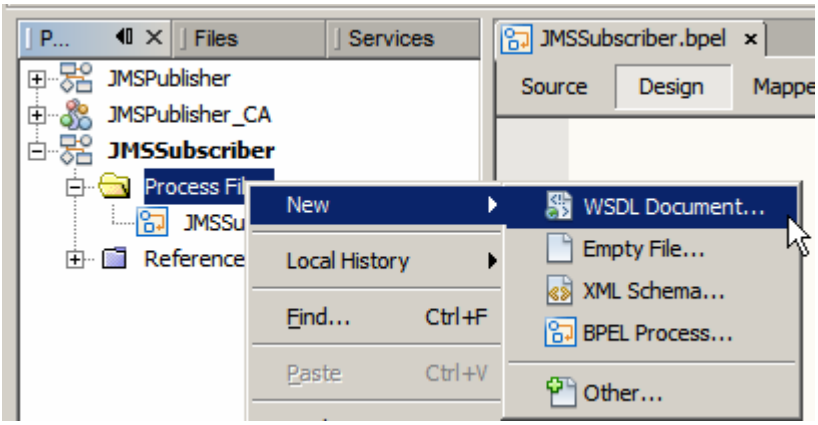
Create "New Project …", SOA -> BPEL Module.

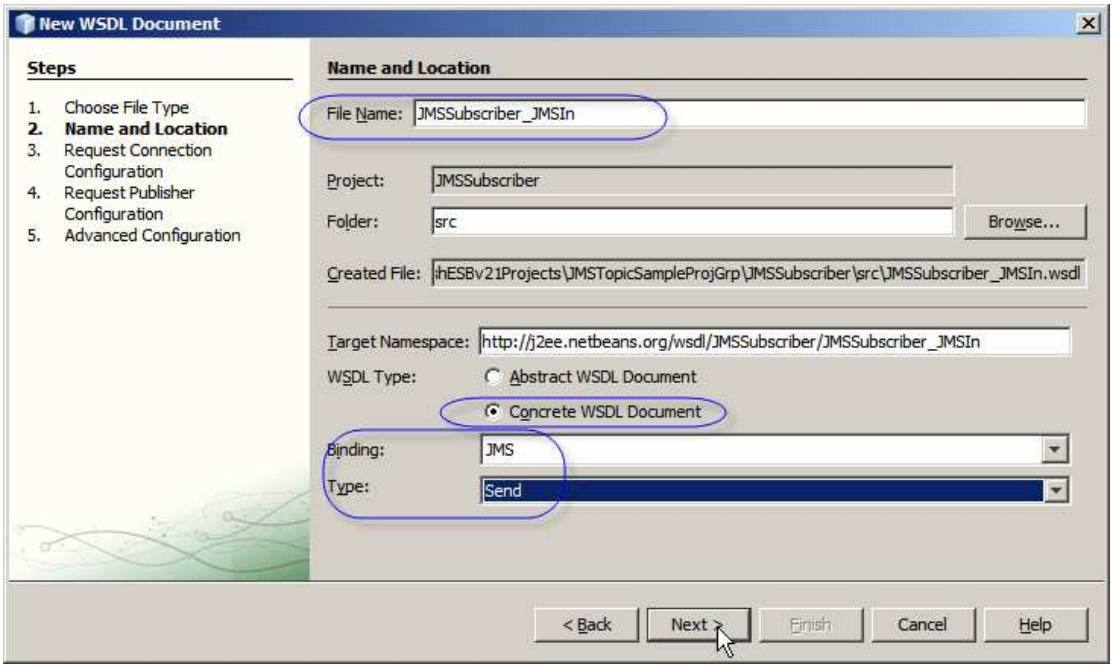


Name the project JMSSubscriber

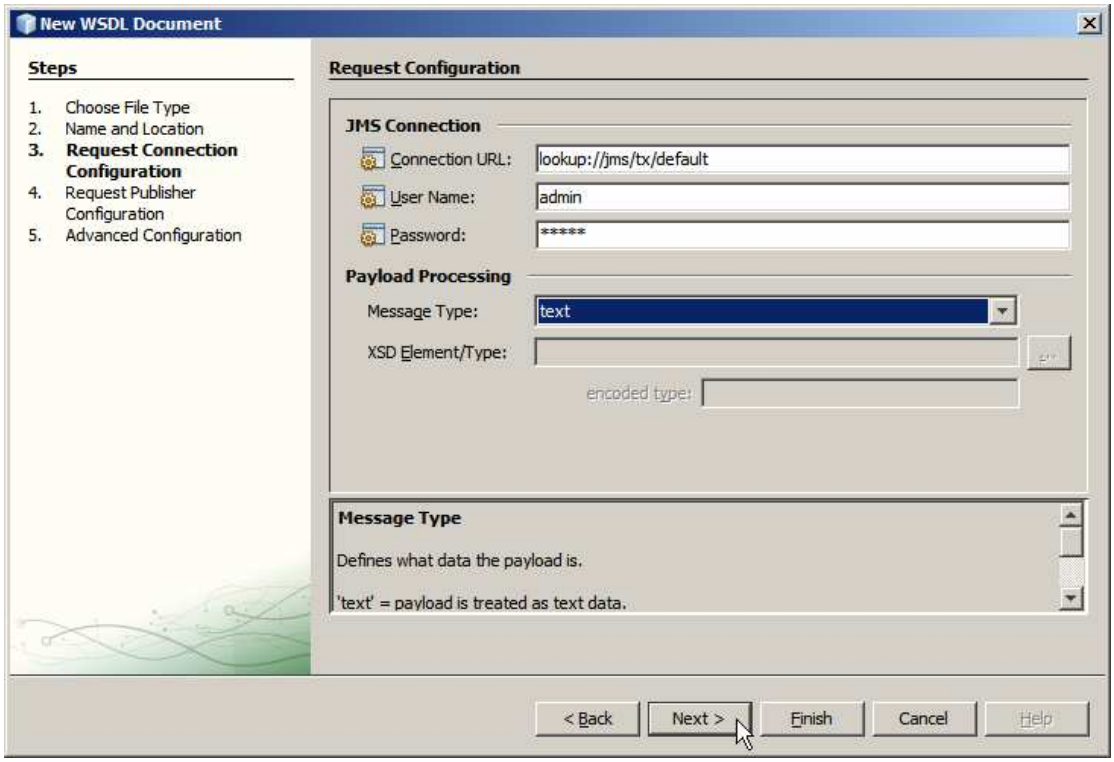


Right-click Project Files folder and choose New -> WSDL Document. Name this document JMSSubscriber_JMSIn, concrete WSDL, JMS Binding of type Send.

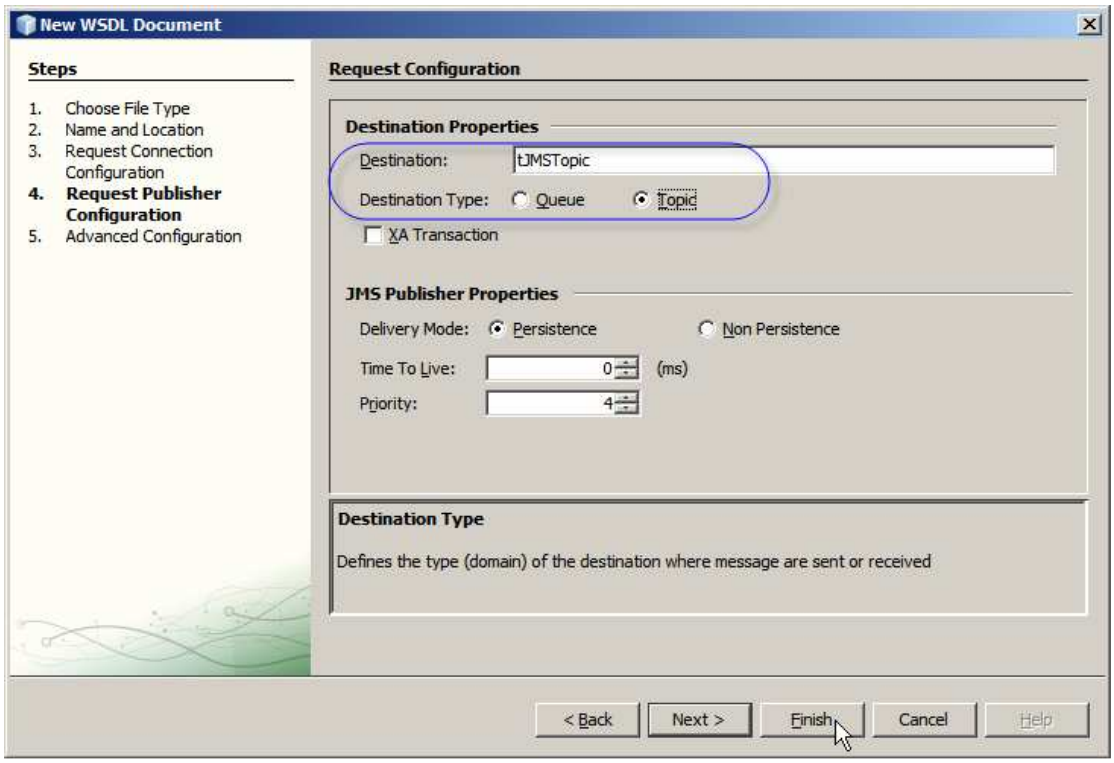




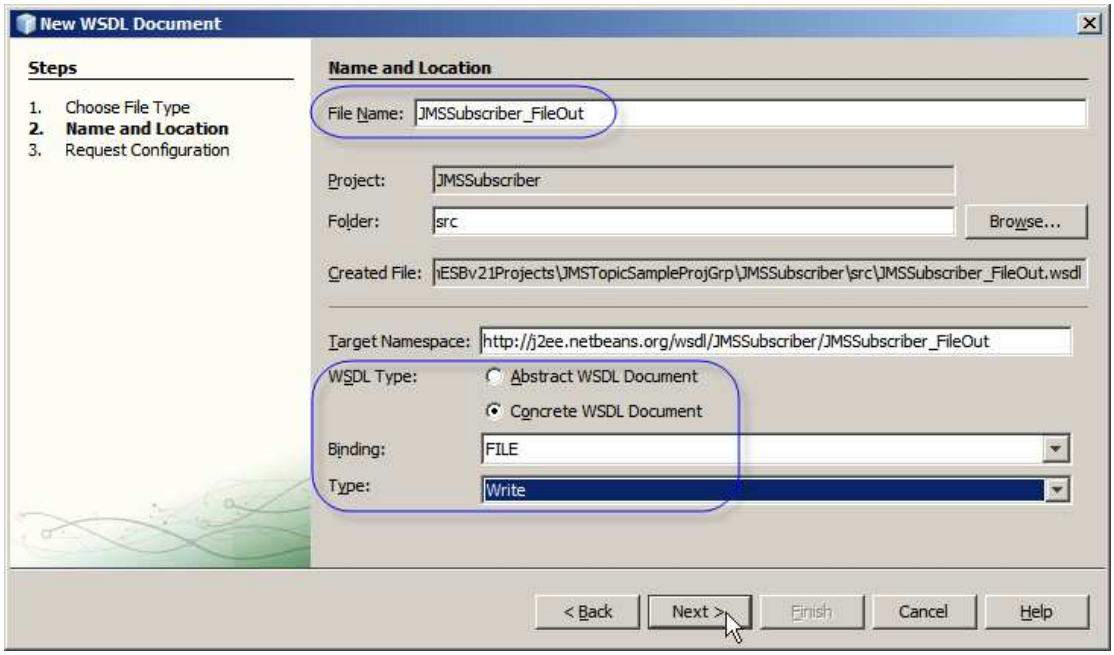
Enter admin and admin for username and password.



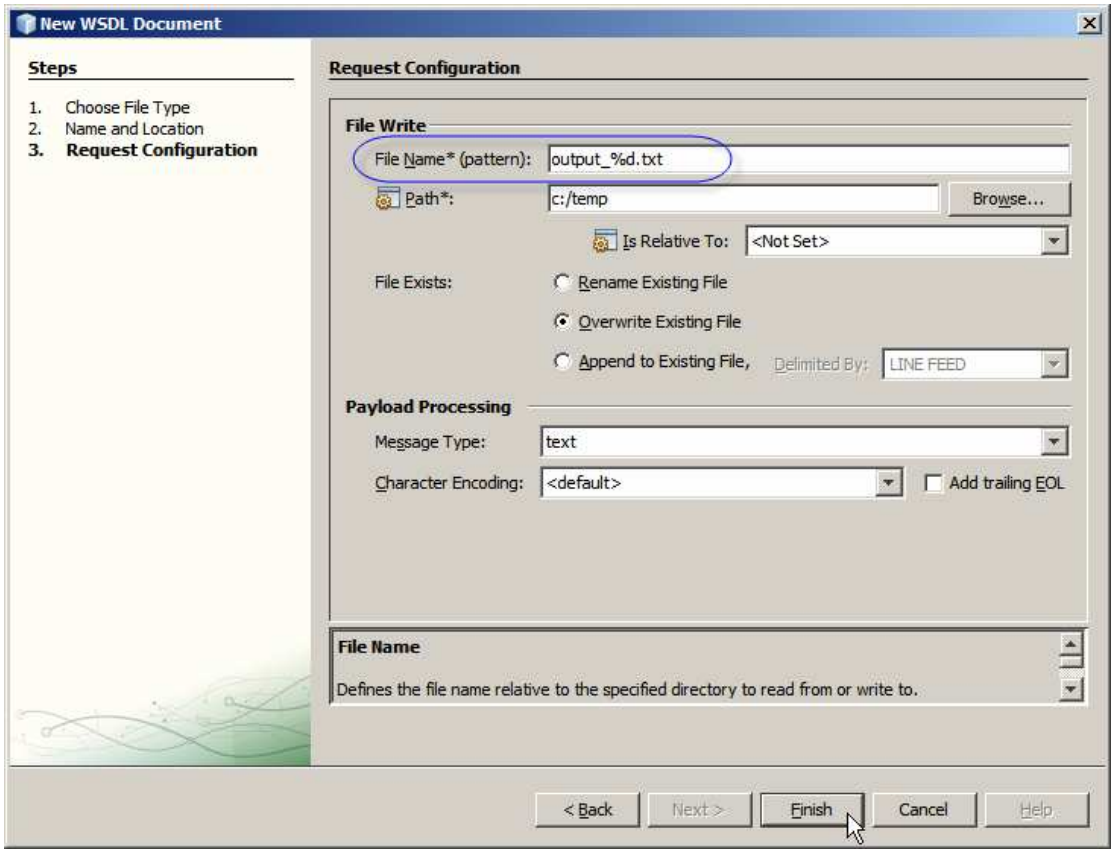
Type the name of the topic to which the JMSPublisher is publishing, tJMSTopic, and make sure the Topic radio button is selected. Click Finish to complete the wizard.



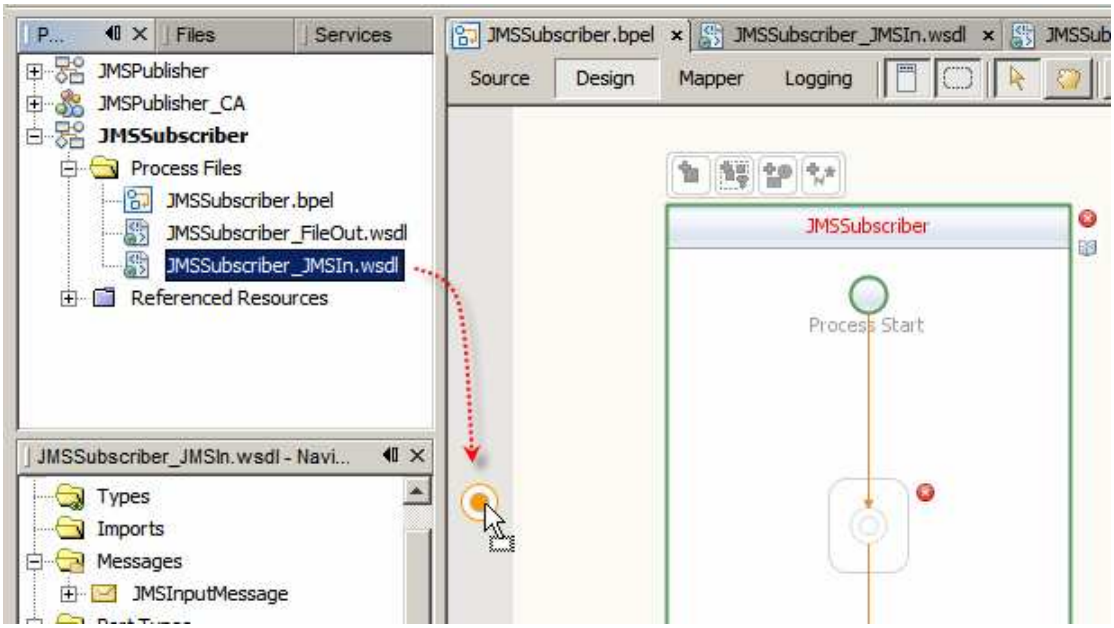
Right-click Project Files, choose New -> WSDL Document … Name the WSDL JMSSubscriber_FileOut, concrete WSDL, FILE binding, type Write.



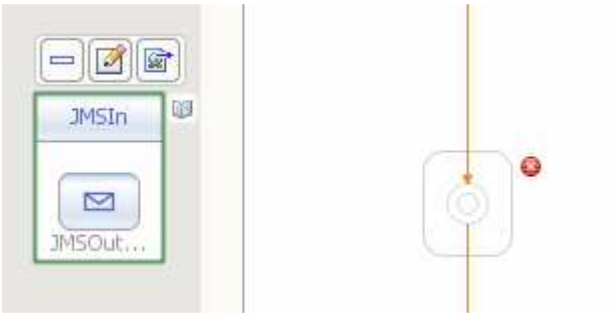
Name the file output_%d.txt and accept all other defaults.



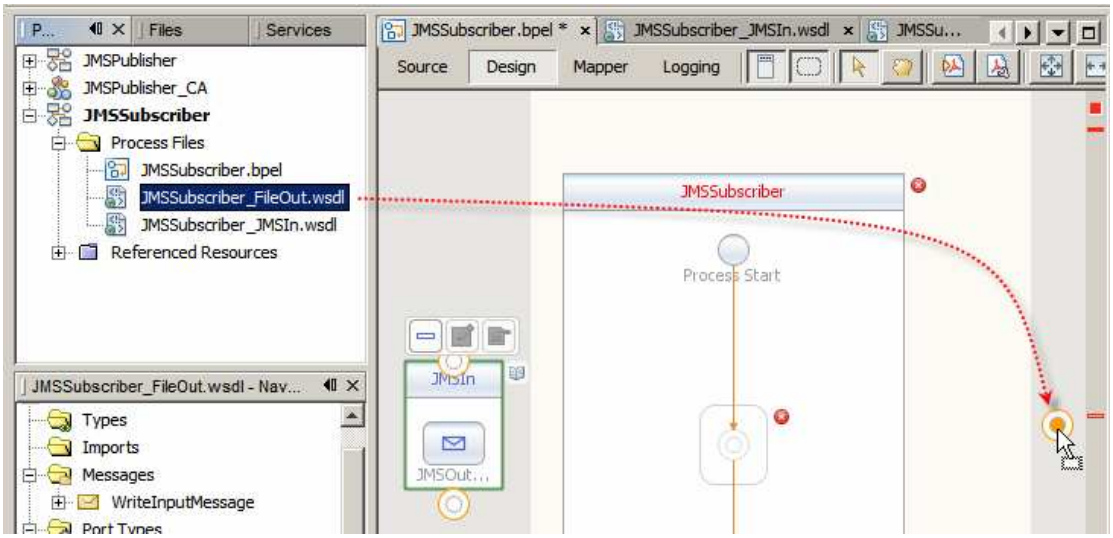
Switch to the BPEL process. Drag the JMSSubscriber_JMSIn WSDL onto the target marker in the left-hand swim line.



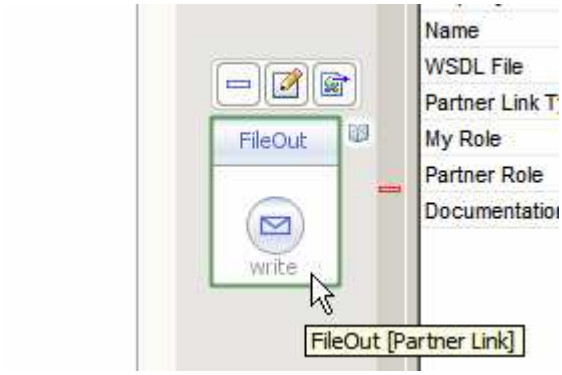
Rename the partner link to JMSIn



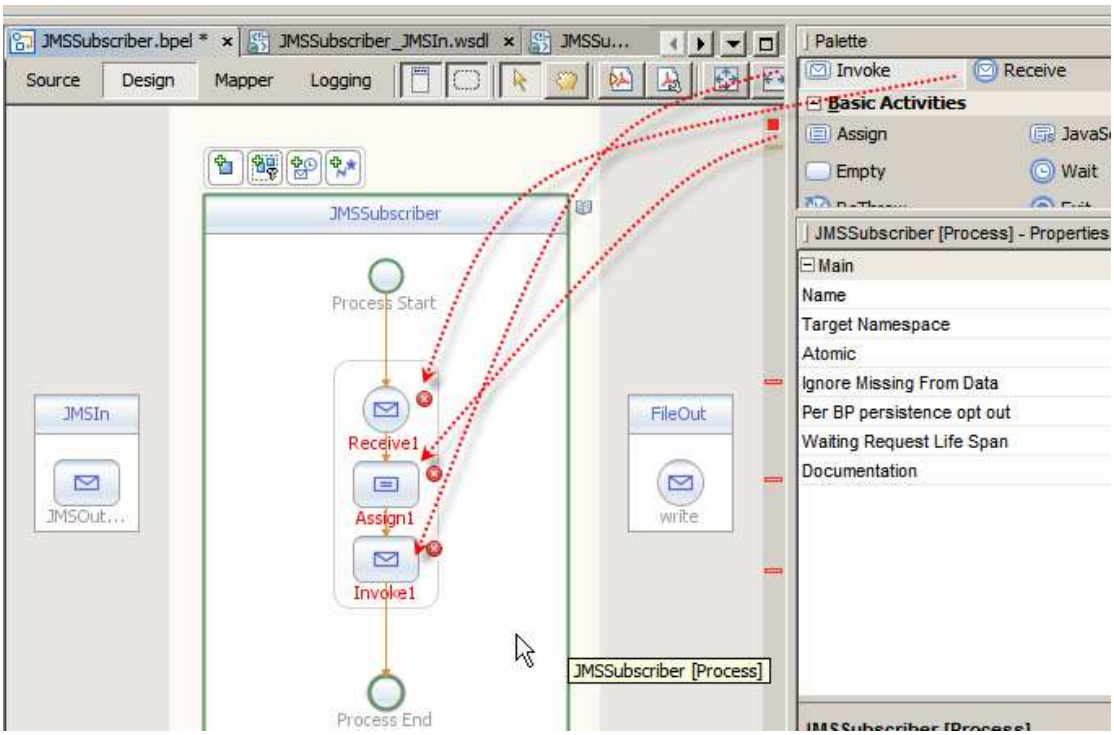
Drag the JMSSubscriber_FileOut WSDL onto the target market in the right-hand swim line.



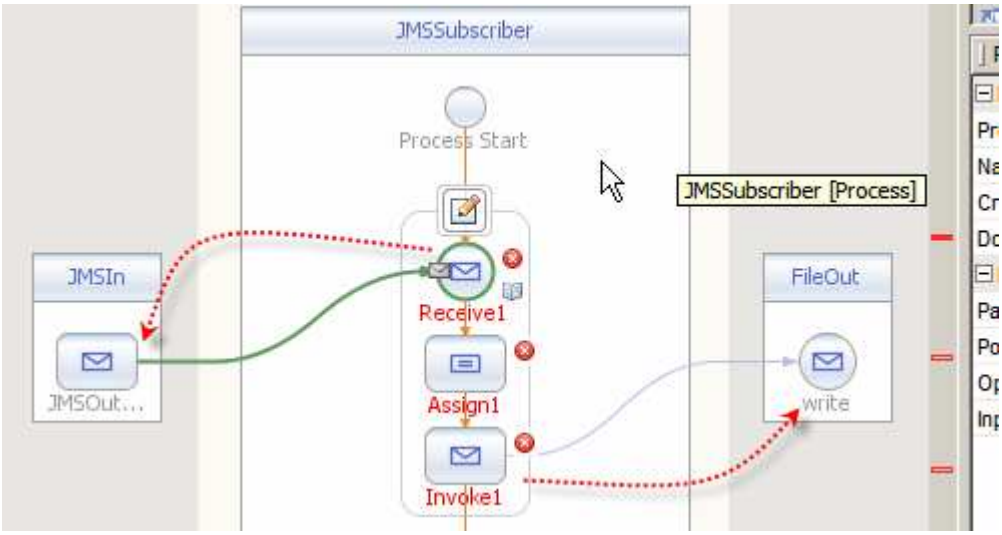
Rename the partner link to FileOut



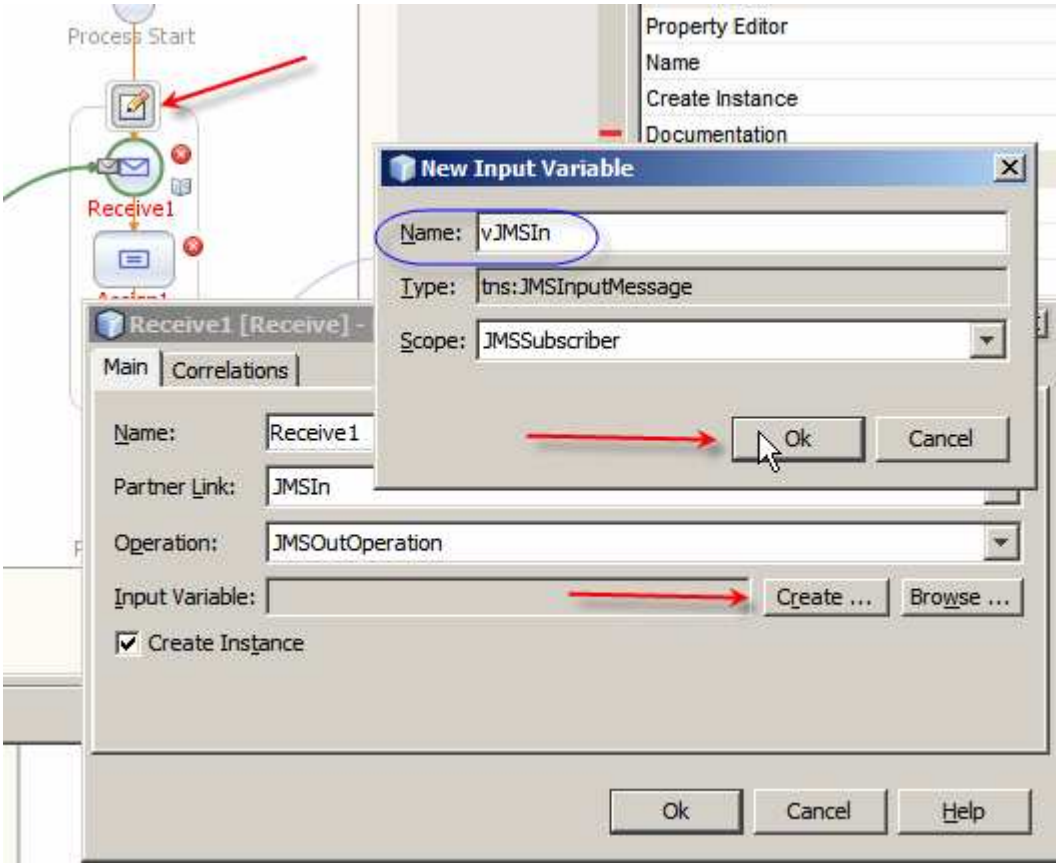
Drag the Receive, Assign and Invoke activities onto the target markers inside the process scope.



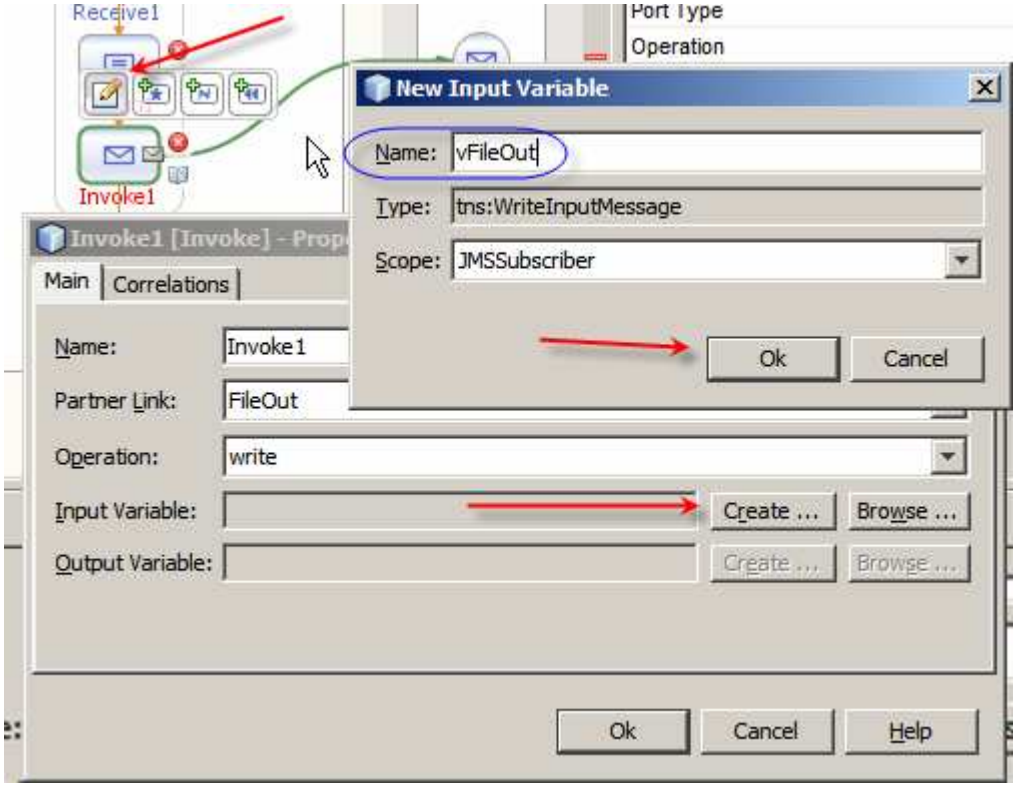
Connect receive and invoke to their respective partners



Edit Receive activity and create a variable vJMSIn

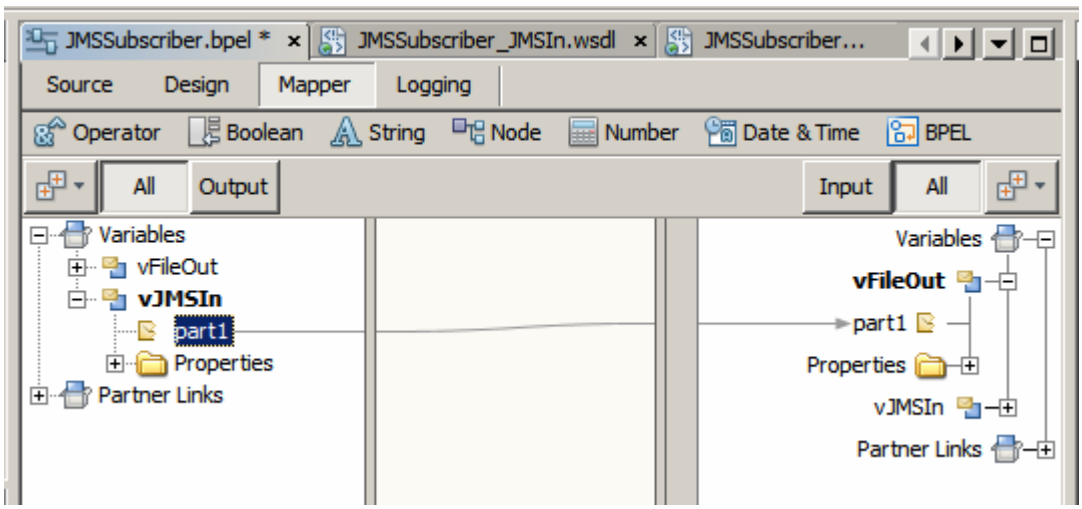


Edit the Invoke activity. Create a variable vFileOut.

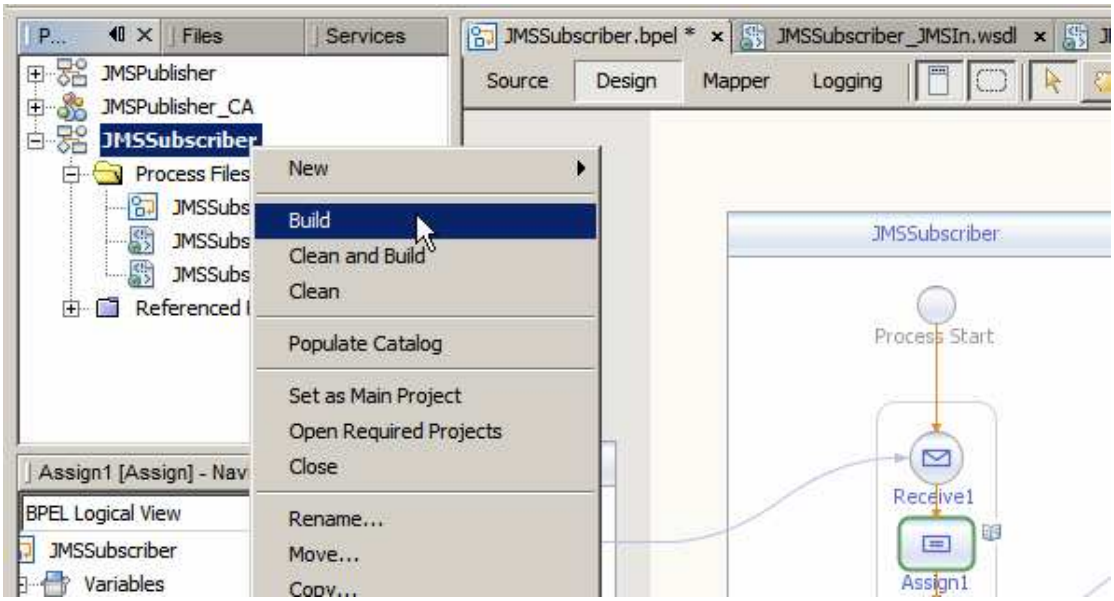


Double-click the Assign activity to which to the Mapper panel.

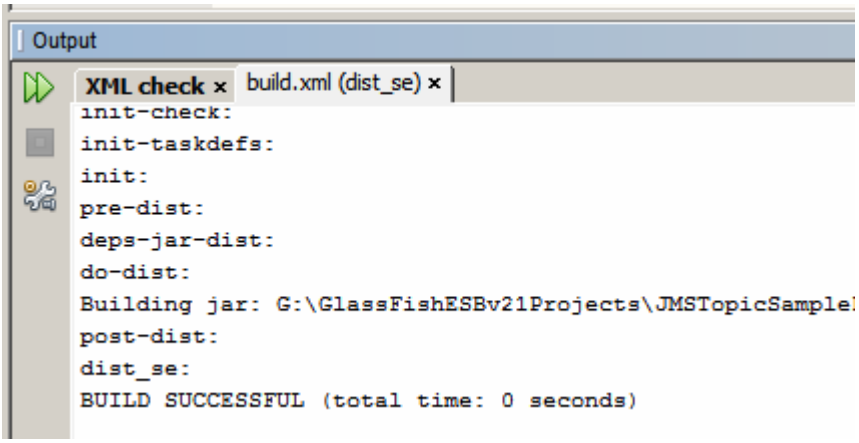
Expand vJMSIn on the left and vFileOut on the right. Connect vJMSIn->part1 to vFileOut->part1.



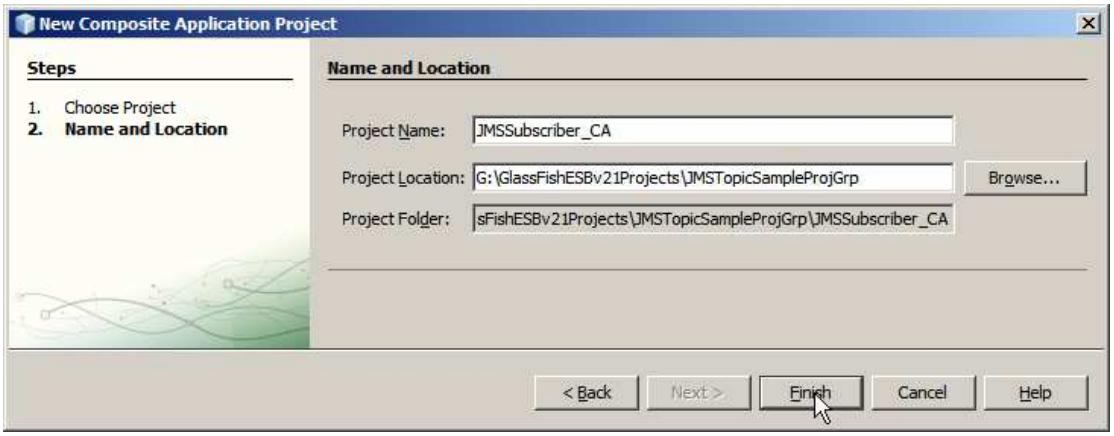
Switch to Design view and build the project.



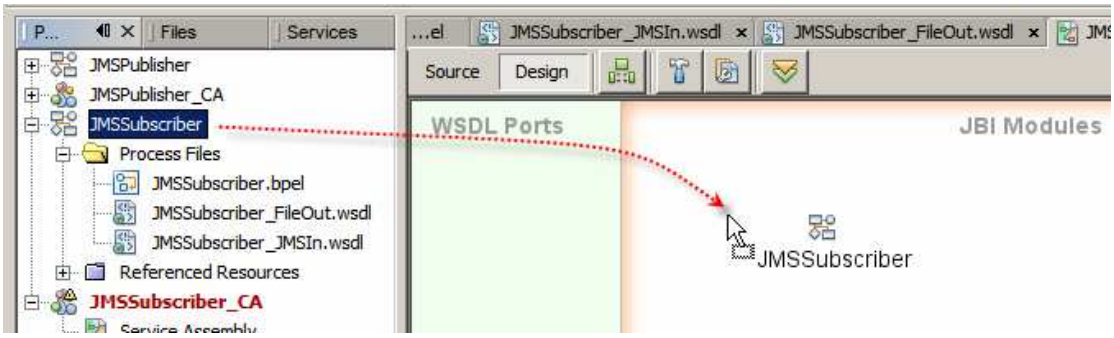
If all is well you should see BUILD SUCCESSFUL in the Output panel.



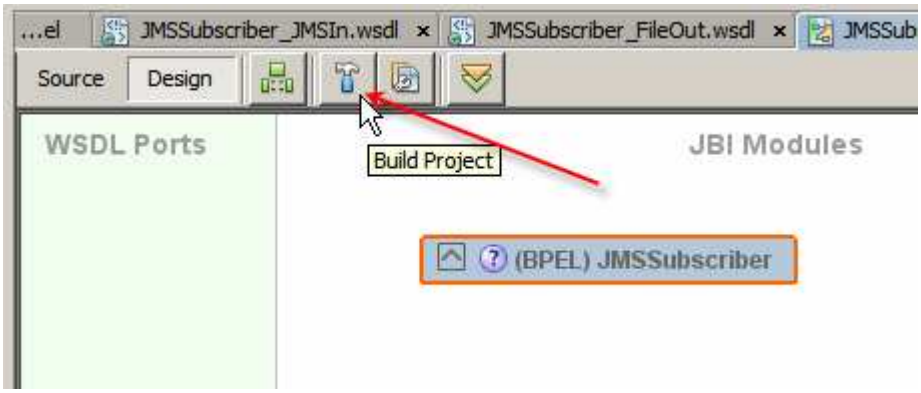
Create a New Project … -> SOA -> Composite Application. Name it JMSSubscriber_CA.



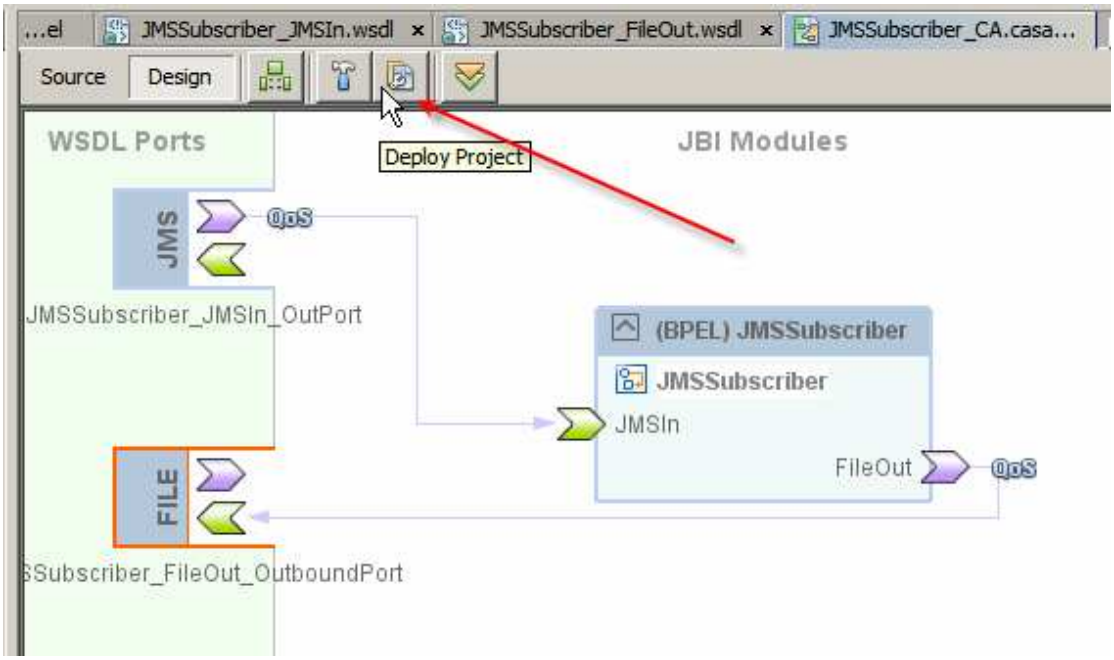
Drag the JMSSubscriber project onto the CASA canvas



Build.



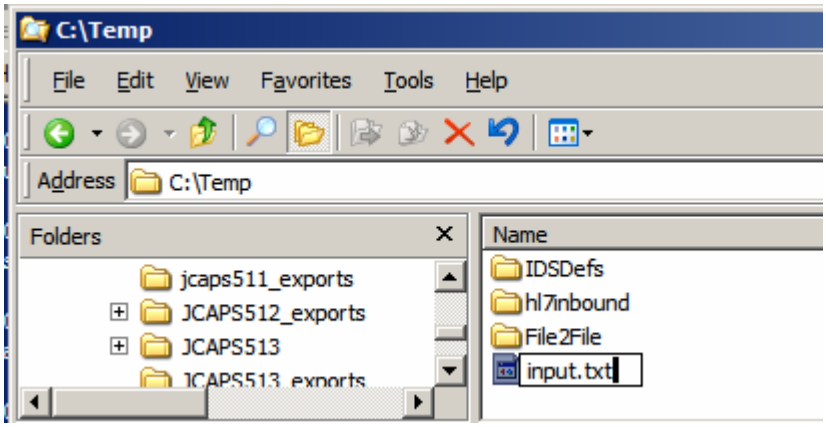
Deploy.

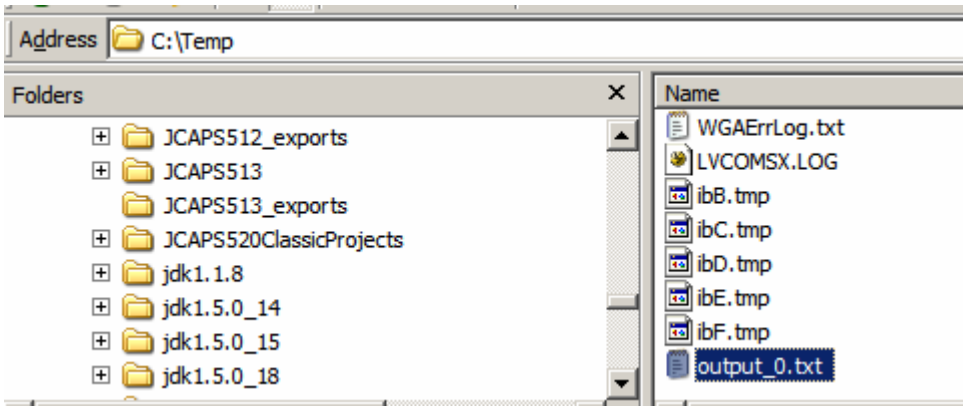


Create a text file in c:\temp (or whatever directory you specified as the one containing a file called input.txt for the File BC in the JMSPublisher project. Name this file input.txt.tmp. Edit it and add the some text, for example “Hello World”.

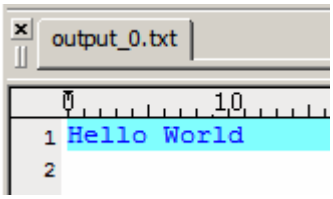
Once ready, rename the file to input.txt. Observe that the file disappears within a second or so and that a file output_0.txt appears in the same directory (assuming you

specified the same directory for the File BC in the JMSPublisher and the JMSSubscriber.





Open output_0.txt to see the content, which should be the same as the content of the input file.



This is it.

We create a solution that read a file in a file system and deposited its content in a JMS Topic. We created a solution that received a message from the JMS Topic, as soon as it was deposited there, and write it to a file system as a file. We demonstrated how to develop a simple JMS publisher and a simple JSM subscriber using JBI-based technologies.