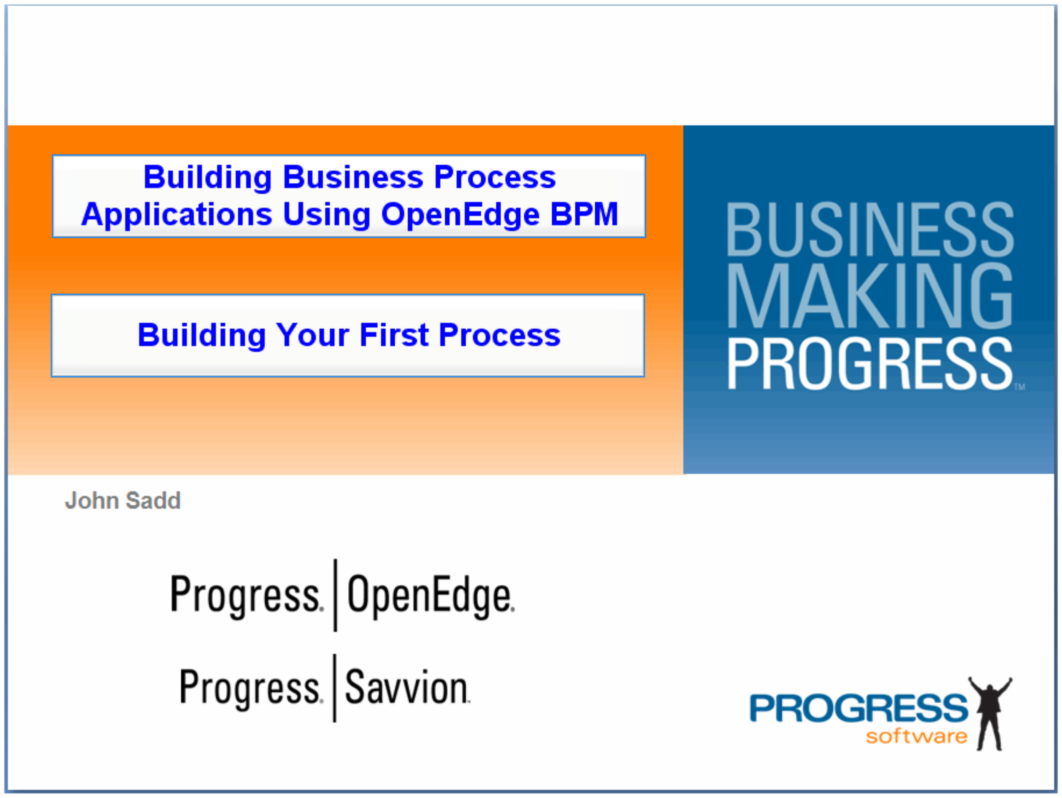
BUILDING YOUR FIRST BPM PROCESS

John Sadd
Fellow and OpenEdge Evangelist
Document Version 1.0
June 2011



Building Business Process Applications Using OpenEdge BPM

Building Your First Process

BUSINESS
MAKING
PROGRESS

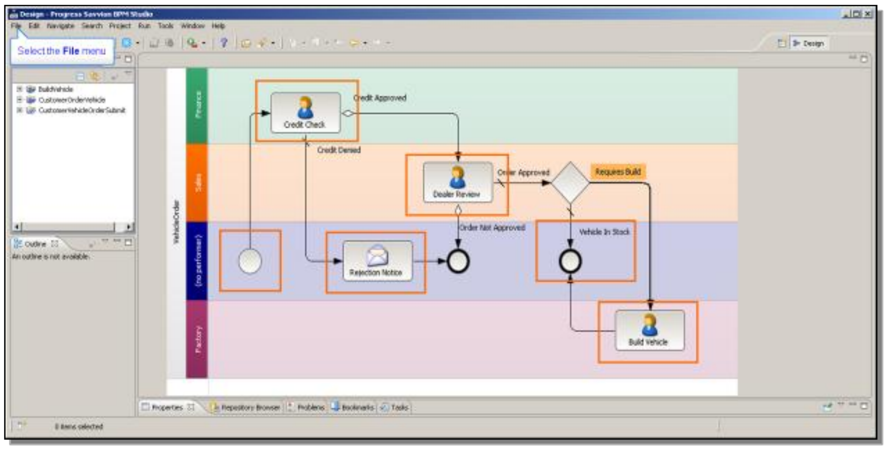
John Sadd

Progress. | OpenEdge.
Progress. | Savvion.
PROGRESS software

DISCLAIMER

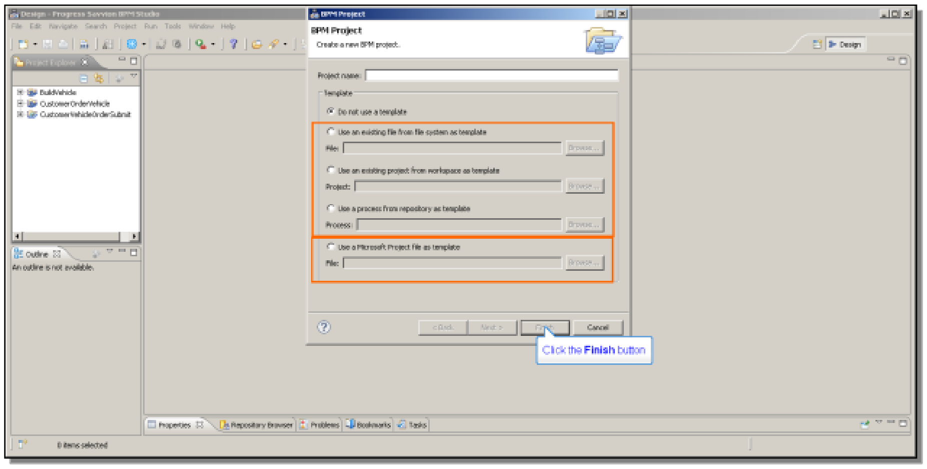
This document accompanies the three-part video series on building your first BPM process using Progress Savvion. The goal is to give you a basis for understanding how to integrate that process with an OpenEdge application. The major design tool in the Savvion toolset, called **BPM Studio**, is where you design the kinds of processes you saw in the video and document ***OpenEdge BPM Overview***. Be aware that there's a complementary Savvion product called **Process Modeler** in which you can do most of the things that I'll show you here, except for deploying the process so that it can be tested and put into production. Process Modeler is intended more for the business analysts who don't handle deployment and other programming tasks, so I do my work in BPM Studio.

To give you an idea of where I'm headed, here's a diagram of what the finished process will look like, using the standard visual notation called **Business Process Modeling Notation**, or **BPMN**.

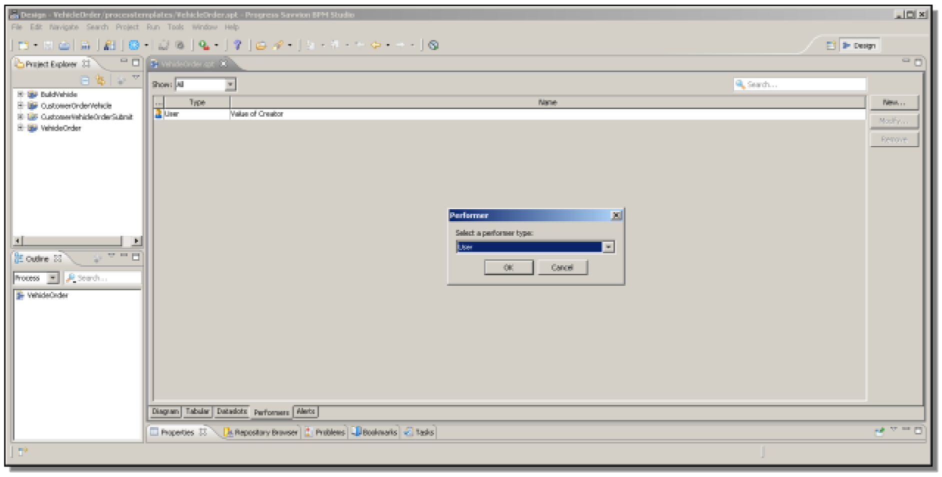


This is a very simplified application that lets a customer order a new car. There is a **Start** step represented by a circle, which lets the customer place the order. Then there's a task that will be carried out by an employee in the Finanace group, to do a credit check. If this fails, the customer gets an email telling them so. Otherwise a salesperson at the car dealer has to review and approve the order. If the vehicle is in stock, we're done. Otherwise the application goes through a step where the order to build the car is made. OK, let's get started.

I need to create a **New BPM Project** to hold my work. I name it **VehicleOrder**. It will be a subset of the **Factory** application that is the **AutoEdge** sample for showing Savvion. You can base a project on another existing one, which is then referred to as a template, or even on something built in Microsoft Project, but I start from scratch.


BPM Project
Create a new BPM project.
Project name:
Template
(●) Do not use a template
( ) Use an existing file from file system as template
File:
( ) Use an existing project from workspace as template
Project:
( ) Use a process from repository as template
Process:
( ) Use a Microsoft Project file as template
File:

One of the essential aspects of a business process, or what we've sometimes called business workflow in our **OpenEdge Reference Architecture** materials, is that a process can involve many separate tasks executed over an extended period of time by multiple people or groups of people, as well as by other software. So the first thing I need to do is define the people or groups, the **Performers**, who have a role to play in this process. These will correspond to system users defined for the runtime environment, but I need to define them here as part of the project source. **User** just means that this performer represents a person rather than a software task, which I'll get into later:


Show: All

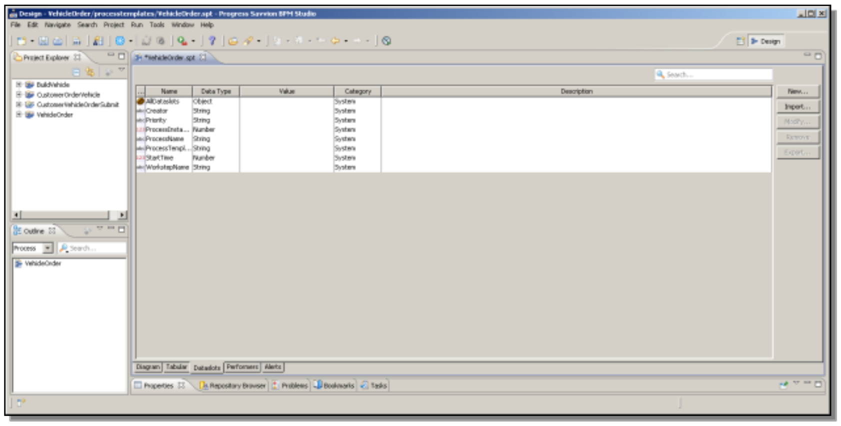
Type	Name
User	Value of Creator

Performer
Select a performer type: User

I call the first performer **Sales**. Ordinarily I would likely define Sales as a group of users, a category of people who do the same kind of job, and assign tasks to the people in that group on some basis, but I'll keep it simple and just view Sales as a single user. I need a second user representing a group I'll call **Finance**.

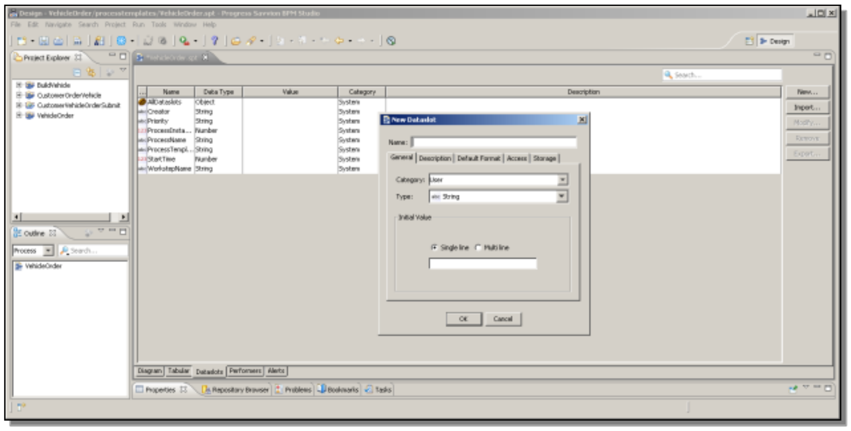
Next I need to define some data values. You can read and write data from an OpenEdge application or anywhere else through the calls you make out to the application, but there are also key data values that can actually control the flow of the process itself. You can also define values within the process that are independent of data in an external application. These are defined somewhat like ABL global variables in that they are accessible from all steps in the process, though you can limit where they can be edited and which ones each task sees. These are called **Dataslots**, and I define a few here to give the process some data to display and to use for decisions.

Here you can see that you start out with some built-in Dataslots in the System Category. I'll be adding application-specific dataslots in a different category.



Name	Data Type	Value	Category	Description
AllDataslots	Object		System	
Creator	String		System	
Priority	String		System	
ProcessInsta...	Number		System	
ProcessName	String		System	
ProcessTempl...	String		System	
StartTime	Number		System	
WorkstepName	String		System	

The first new dataslot is the **CustomerName**. The default application category for dataslots is just called **User**. You can define more specific categories to help organize your dataslots as well, so you can share dataslots between applications by category, for instance. Anyway, my VehicleOrder process represents a piece of an application that controls, in a very simplified way, how someone orders a new car, and this dataslot holds the name of the customer buying the car:



New Dataslot

Name:

General | Description | Default Format | Access | Storage

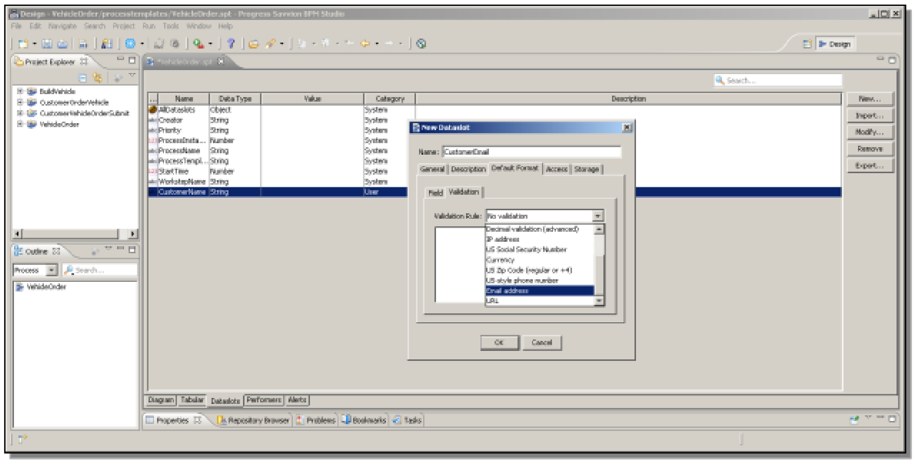
Category: User

Type: String

Initial Value

[✓ Single line] Multi line

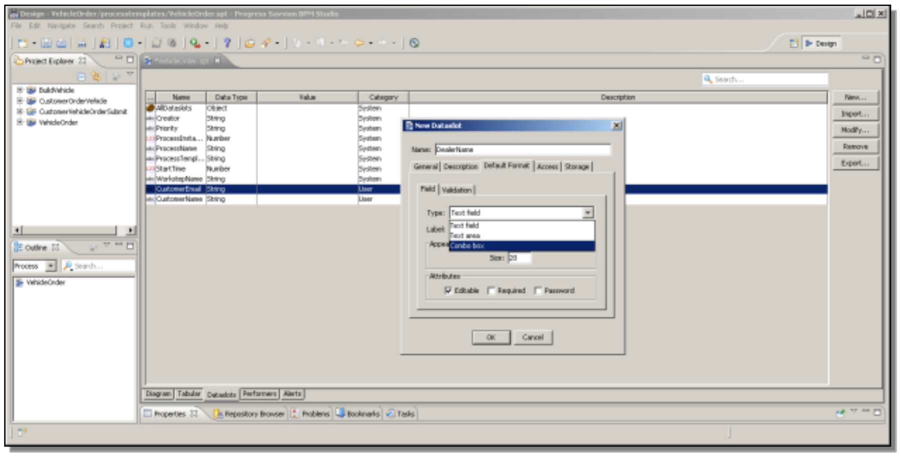
The second new dataslot holds the customer's email address. This is also a string, but I want to point out to you some of the format checking that Savvion can do. There are a number of validation types for different datatypes, and for a string, one of them is to verify it as a valid email address. BPM Studio also uses the understanding that this string is an email address in another way, which you'll see later.


New Dataslot

Name: CustomerEmail

Validation Rule: No validation

The next dataslot is another string, but in this case, I'm going to change the default display format to a combo box, because I want to define a list of possible values for the car dealer name. You can define data entry forms right in Savvion if you want, and this setting determines the default display format for them. You can also indicate which values are required, so I do that here, by checking the **Required** checkbox:


New Dataslot

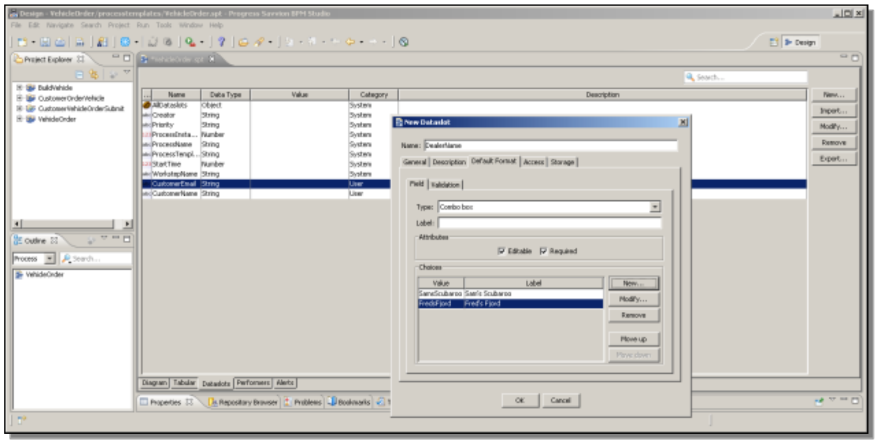
Name: DealerName

Type: Text field

Size: 20

Attributes: [✓ Editable] Required Password

I just add a couple of values to the list, one for a dealer who sells Fjords, and one for a dealer who sells Scubaroos:


New Dataslot

Name: DealerName

Type: Combo box

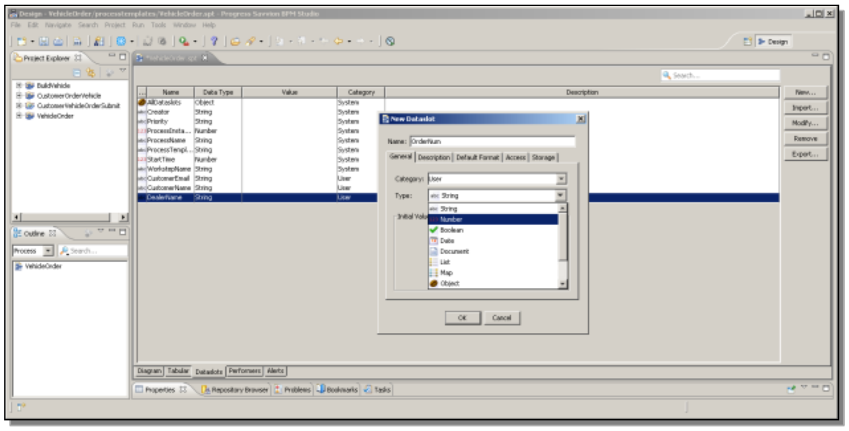
Label:

Attributes: [✓ Editable] [✓ Required]

Choices

Value	Label
SamsScubaroo	Sam's Scubaroo
FredsFjord	Fred's Fjord

Now I define a dataslot whose value isn't a string, the order number. In a later presentation I'll get into more detail about how you can map dataslot values to OpenEdge datatypes, which isn't always a one-to-one match, but here I'm just defining values to be used within the process:



Name	Data Type	Value	Category
AllDataslots	Object		System
Creator	String		System
Priority	String		System
ProcessInsta...	Number		System
ProcessName	String		System
ProcessTempl...	String		System
StartTime	Number		System
WorkstepName	String		System
CustomerEmail	String		User
CustomerName	String		User
DealerName	String		User

New Dataslot

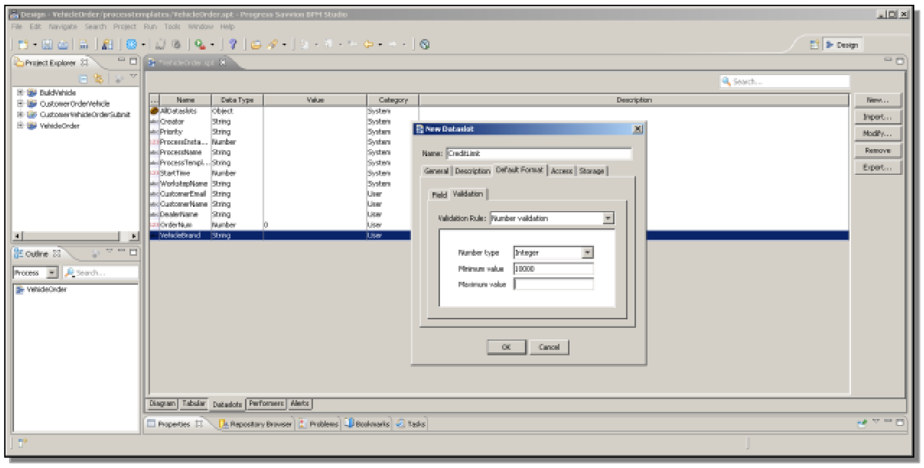
Name: OrderNum

Category: User

Type: String

Now I want the brand of vehicle being bought. I make this another combo box with two values, to match the dealers I defined. In a complete application, lists like these would normally be populated out of the database through calls back to OpenEdge, but that's more advanced than I have time for in this simple example.

Next is the customer's credit limit, which can determine whether the customer's order is going to be approved. Here I can show you some more validation rules you can apply within the process, in this case that the initial value is 20000, and that the value is required to be set, and that the value has to be within a certain range, in this case between 10000 and 50000. I'll show you when we get to actually running the application how these rules are enforced.



Name	Data Type	Value	Category
AllDataslots	Object		System
Creator	String		System
Priority	String		System
ProcessInsta...	Number		System
ProcessName	String		System
ProcessTempl...	String		System
StartTime	Number		System
WorkstepName	String		System
CustomerEmail	String		User
CustomerName	String		User
DealerName	String		User
OrderNum	Number	0	User
VehicleBrand	String		User

New Dataslot

Name: CreditLimit

Validation Rule: Number validation

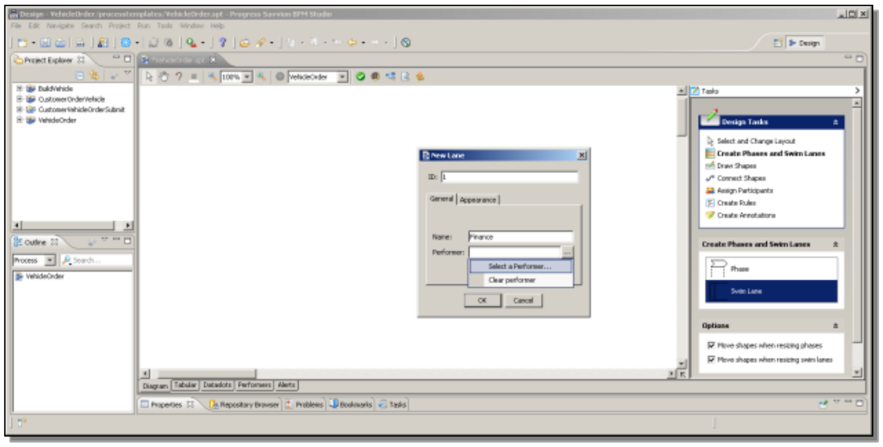
Number type: Integer

Minimum value: 10000

Maximum value:

Those are all the dataslots I need to get started. If I click on the **Diagram** tab, I'm returned to the BPMN diagram, where I can draw the boxes and arrows that define the process visually.

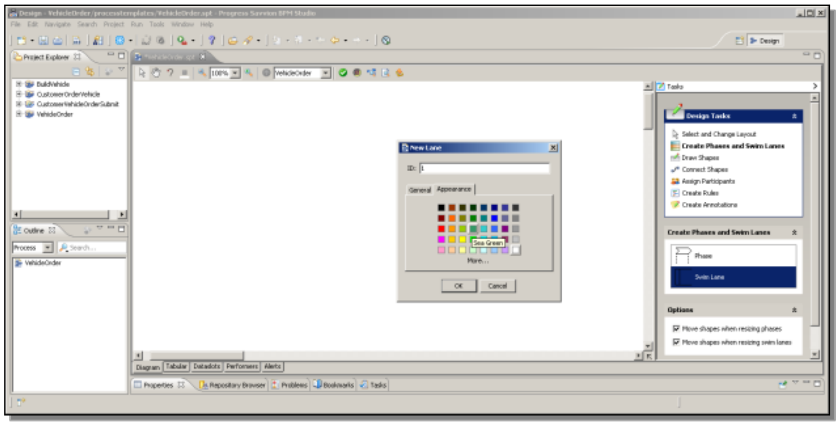
One of the nice ways to organize all the tasks in a process visually is by defining what are called **phases and swimlanes**. A phase is a subset of the total time that the process can take, organizing the tasks into a sequence. If the diagram is laid out so that the tasks proceed from left to right, then it can be divided into vertical phases. A swimlane goes the other way, horizontally, to separate visually tasks that are assigned to different performers. My process isn't extensive enough to require multiple phases, so I'll define some swimlanes here. The first will be for Finance tasks. So I select Finance as the performer, from the Performers I defined a moment ago:



ID: 1

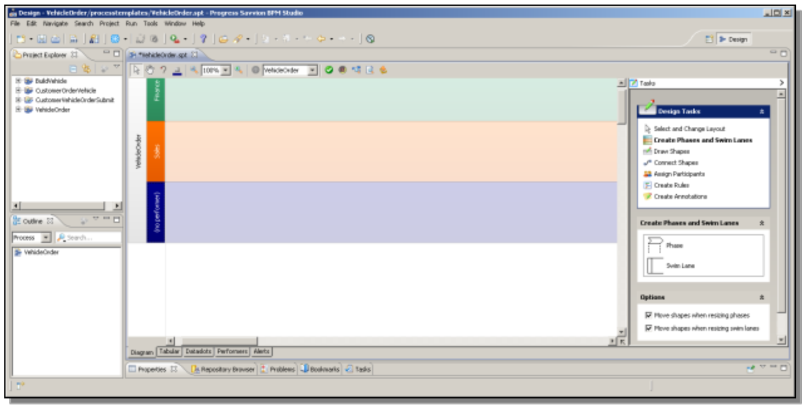
Name: Finance

I can also pick a distinctive background color for the swimlane. Finance people are supposed to see green, so I'll pick Sea Green as the color (sorry…).

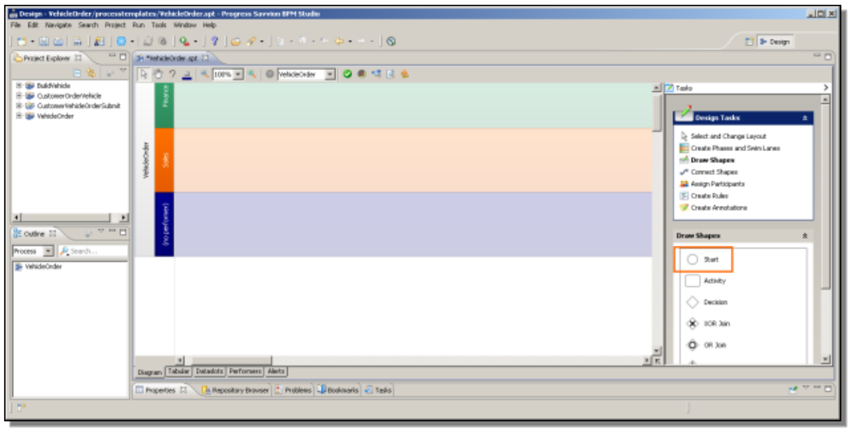


ID: 1

I then define two more swimlanes, one for the Sales tasks and one for tasks not performed by either of those. Now I've got a canvas to lay out tasks on.

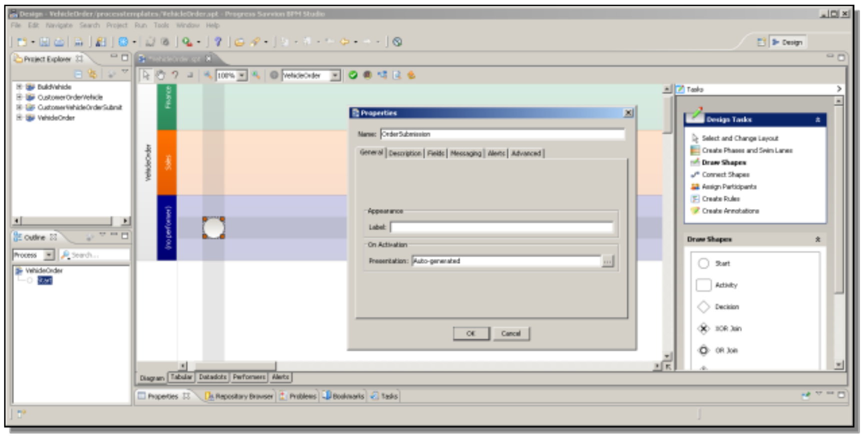


I select the **Draw Shapes** group over under **Design Tasks**. The first task I select is a **Start** task, just represented as a circle. Every process has a Start step, and just one Start step, so the Start step effectively defines the overall scope of the process.



I drag that over to the diagram in the **No Performers** swimlane. The customer will actually perform the Start step, by accessing this task from a website or in some other way to initiate the process, but isn't really considered a performer in the process itself.

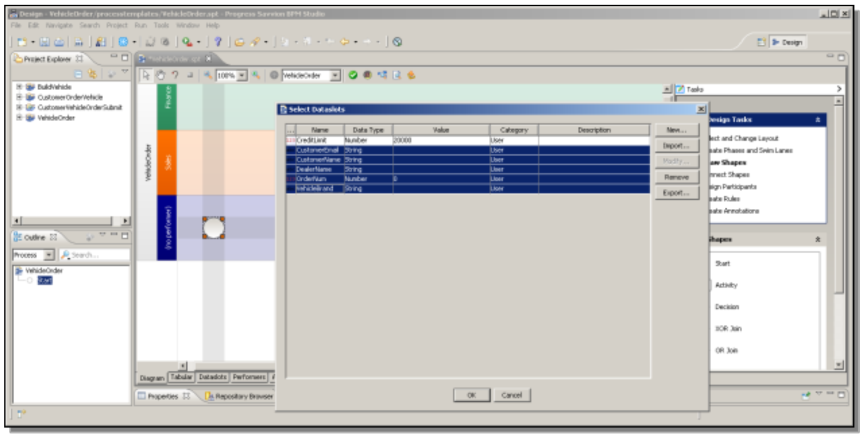
I can then right-click on the step, and select **Properties**. I can give every step a name, and I'll call this **OrderSubmission**:


Properties

Name: OrderSubmission

Presentation: Auto-generated

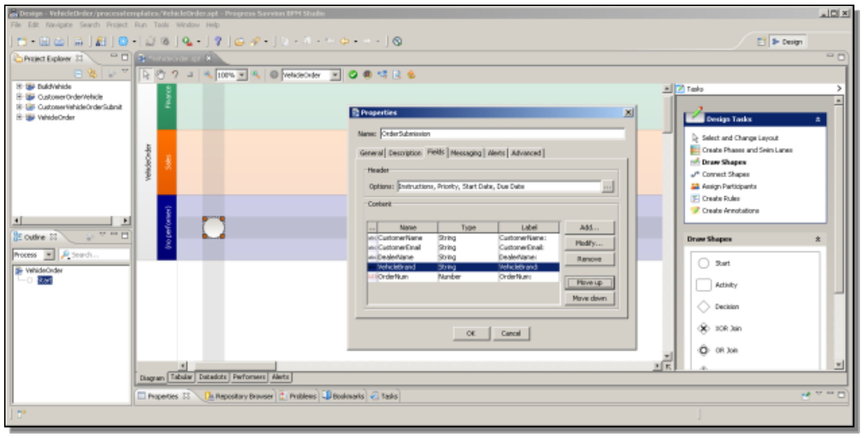
The customer will have a form to fill out, a form that is defined as part of the process. You can also integrate forms defined in OpenEdge using GUI for .NET or as Web forms or whatever else as well. This is going to be a very simple auto-generated form, which will come up just as a list of fields to fill in, so I select the **Fields** tab. Fields are the user interface representations of Dataslots, so this tab lets me pick some dataslots to display in the form. I **Add** dataslots to the form so the customer can fill them in, selecting all the existing User dataslots except for the CreditLimit, which someone else has to set:



Select Dataslots

...	Name	Data Type	Value	Category	Description
	CreditLimit	Number	20000	User	
	CustomerEmail	String		User	
	CustomerName	String		User	
	DealerName	String		User	
	OrderNum	Number	0	User	
	VehicleBrand	String		User	

I can rearrange their display order, but other than that I'm accepting a default display format:



Properties

Name: OrderSubmission

Header

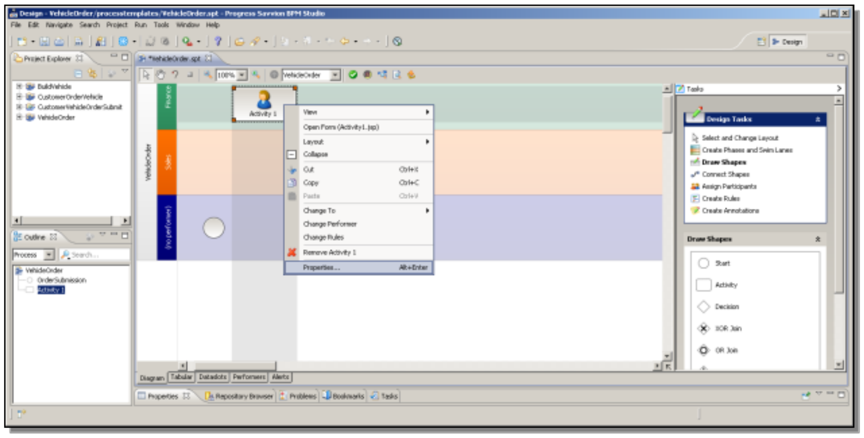
Options: Instructions, Priority, Start Date, Due Date

Content

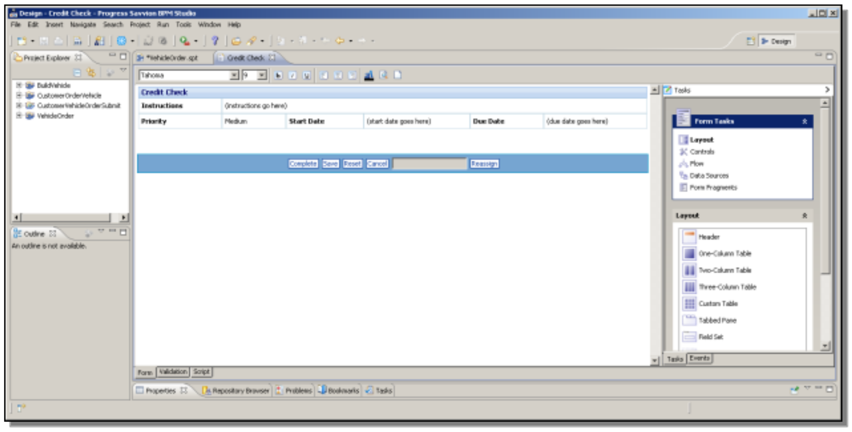
...	Name	Type	Label
	CustomerName	String	CustomerName:
	CustomerEmail	String	CustomerEmail:
	DealerName	String	DealerName:
	VehicleBrand	String	VehicleBrand:
	OrderNum	Number	OrderNum:

Now I can define the first activity task following the start step, something one of my performers has to do. I can drag the rectangle activity shape onto the diagram, and drop it onto the Finance swimlane. Dropping it onto this swimlane automatically sets the activity's performer to Finance.

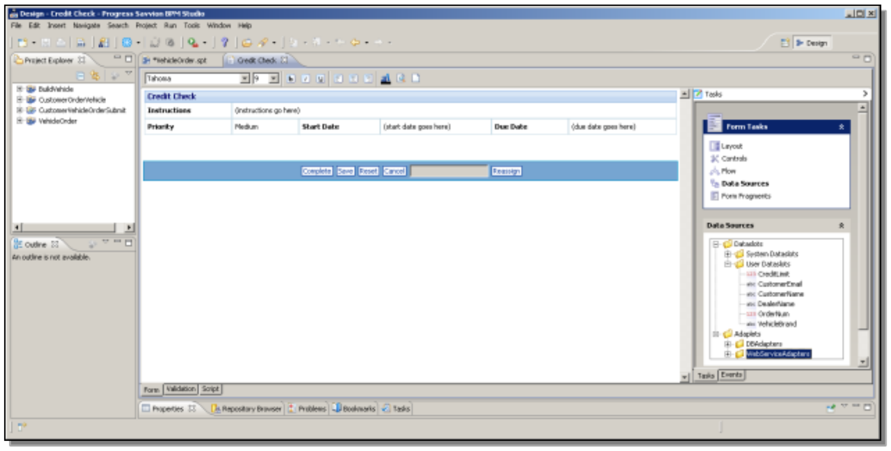
Now I right-click to set some other properties:



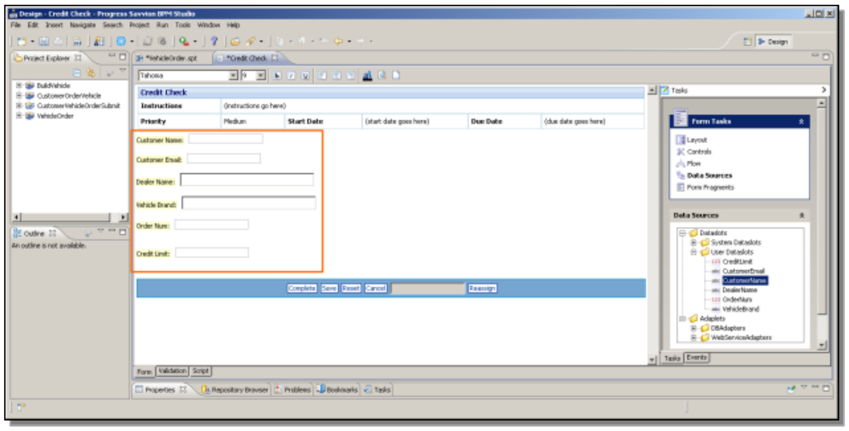
I'll name this task **Credit Check**. You can see the Performer has been set to Finance. Someone from Finance will review the order and determine whether the customer is creditworthy. BPM Studio is prepared to associate a form with every workstep that's performed by a person, and when I rename the workstep I am prompted to give the same name to the form. I can then go into the form, and show you how you can lay out the form to your specifications, instead of just using an auto-generated form. Here is the form skeleton:



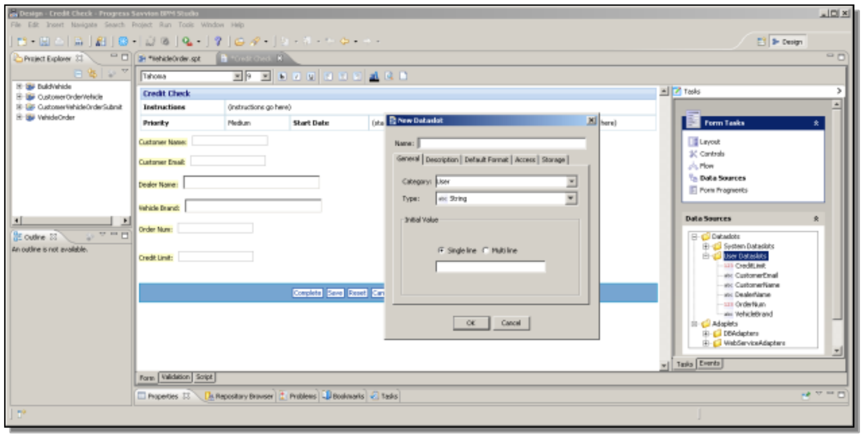
It's got a default header and footer area, which you can change as well. I can drop individual controls like text fields onto the form, or I can select specific dataslots by clicking **Data Sources** under the **Form Tasks**. The dataslots I defined are my data sources. I can expand the User dataslots -- remember that there are also predefined dataslots called System dataslots -- and select my CustomerName value:



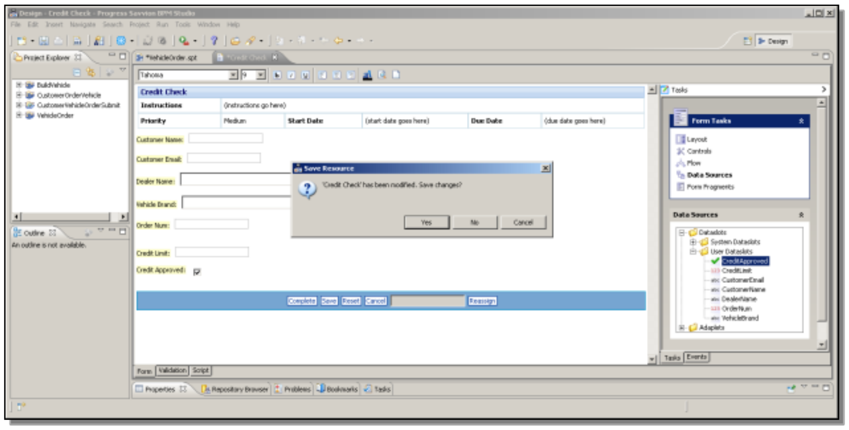
I then drag that onto the form, and drop it between the form's header and footer sections. Below I've done the same for the other fields the Finance person needs to see:



I realize now that I need another value for the process's decision making, a flag that signals whether the customer's credit was approved or not. All I have to do is right-click on the User Dataslots, and select Add. I define a Dataslot named **CreditApproved**:



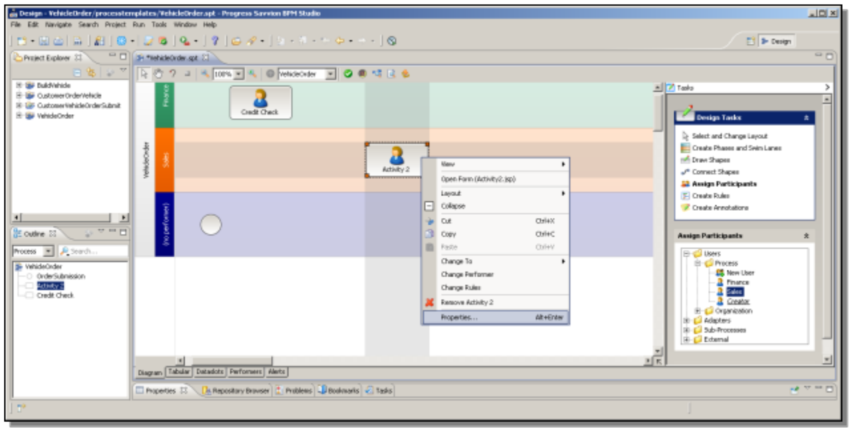
It’s of type Boolean, so by default it will be displayed as a checkbox. Now that new Dataslot is in my list, so I can select it, and drag it onto the form like all the others. In this case I’d like the checkbox to be to the right of the label, so I drag it over to the right and the label stays where it was. Once I’m done defining this form, I just tell BPM Studio to save the source for the jsp file that it will generate for it:


Save Resource

'Credit Check' has been modified. Save changes?

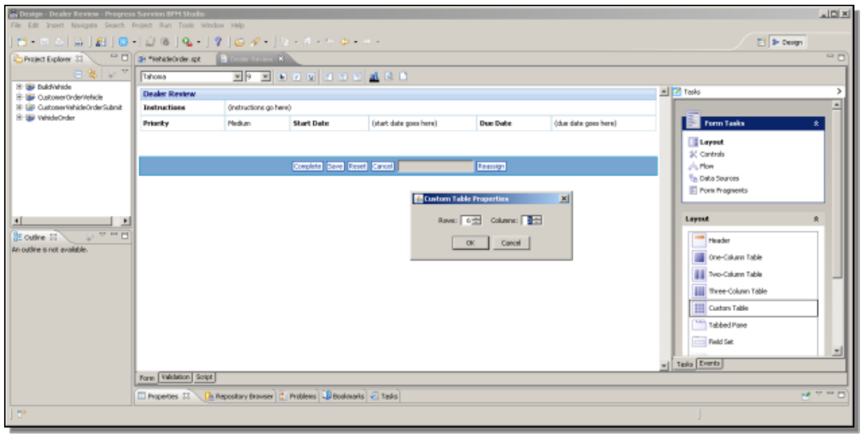
Yes No Cancel

Now I need to define the next Activity workstep, which is going to be for a salesperson at the dealer to approve the order and determine whether the vehicle is in stock at the dealer or not. I define the activity in a different way from the first. Instead of just dragging the activity rectangle onto the diagram and then defining the performer, I start by selecting the performer from the **Assign Participants** task group, which in this case is Sales, and effectively drag the Performer onto the diagram. As I showed you before, the swimlanes actually assign the performer for you as well, so either way, when I open the Properties, the performer is already defined.



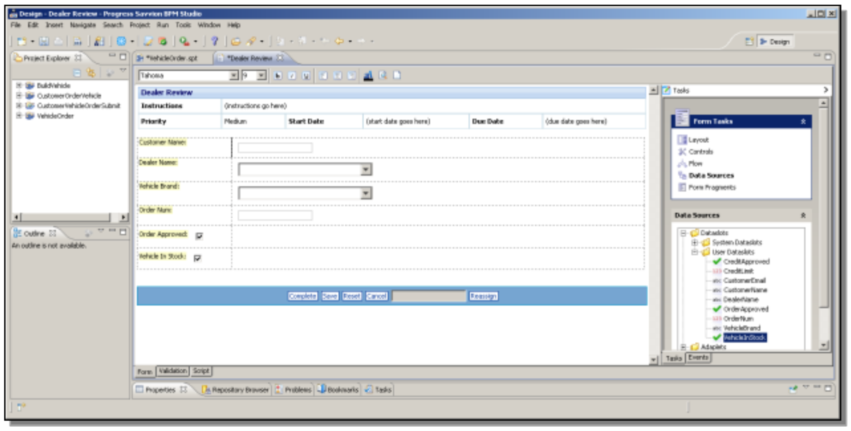
I name this task **Dealer Review**, confirm the name of the form, and right-clicking again on the Activity, I open the form. This time I’ll show you another variation. First I need to make sure that I’ve clicked in the form between the header and footer. Otherwise any controls I select will wind up under the footer. Then under the Layout options, I select **Custom Table**.

I can define a spreadsheet-like grid of labels and controls with one, two, or three columns, or I can set the number of columns and rows in one step. In the Custom Table, I say I want six rows of two columns:

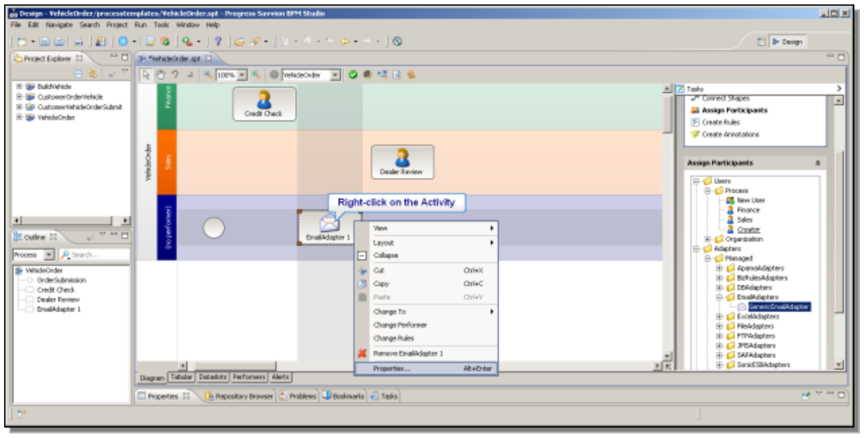


Rows: 6 Columns: 2

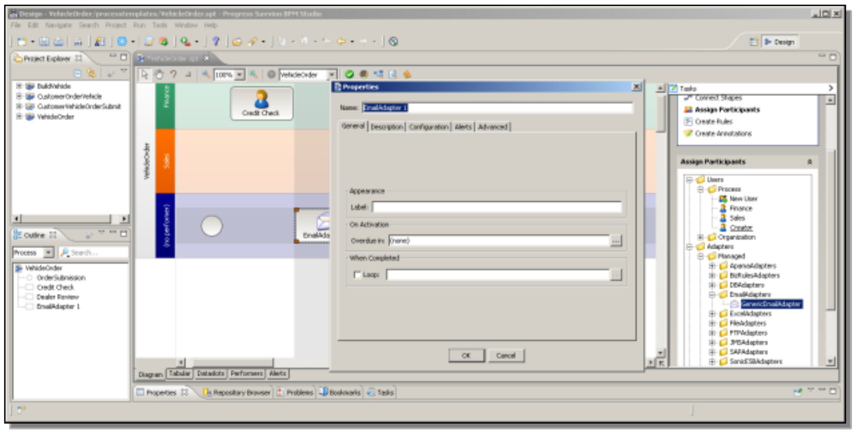
This provides me with a grid that I can now drop other controls onto. Under my user dataslots, I select the CustomerName as I've done before, and drop that onto the form. If I want the label in one column and the control in another, I can select the control and drag it into the second column, and the label stays behind. After I've done that for all the fields, once again I realize I need a couple of new Dataslots for my process decisions. The first is a flag the salesperson sets as to whether the final order is approved or not. That's another Boolean value. The second is another Boolean to flag whether the vehicle is in dealer stock or needs to be built to order. So with these two new dataslots defined, I drag them onto the form as well, and move the checkbox to the right of the label. Below I've done that for both; I can also drag the divider between columns to move the controls over some, and I'm done with the Dealer Review form:



Next, under the **Assign Participants** task type, I select a different type of participant, not a person, but an **Adapter**, which is a software interface to a specific type of behavior defined outside the model. Savvion has a number of predefined adapters, identified as **Managed Adapters**. There are quite a number of them. I want an email adapter so that I can have the process send an email message. The built-in adapter is the one identidied as *generic*. You can also create custom variations of an adapter, as well as entirely new adapters of your own. I drag the generic email adapter onto the diagram like any other performer, and open its properties:

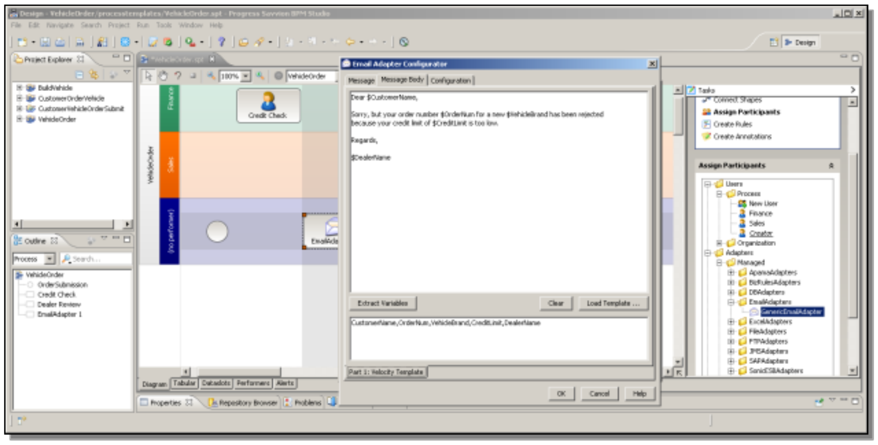


In its properties, I name it **Rejection Notice.** This sends an email if the customer's credit is not approved. There's a **Configuration** tab for an adapter:



Under that, I click **Configure** to define the different parts of an email message:

I hardcode the From email address to be the order department at my fictitious car dealer. I type in a standard subject, and in the message body, I enter some boilerplate text. The email adapter lets me enter placeholders for values defined at runtime. I just enter any value to be replaced with a dollar sign in front. Once my message is complete, I click **Extract Variables**, and the adapter generates a list of variables to be replaced by Dataslot values:


Dear $CustomerName,

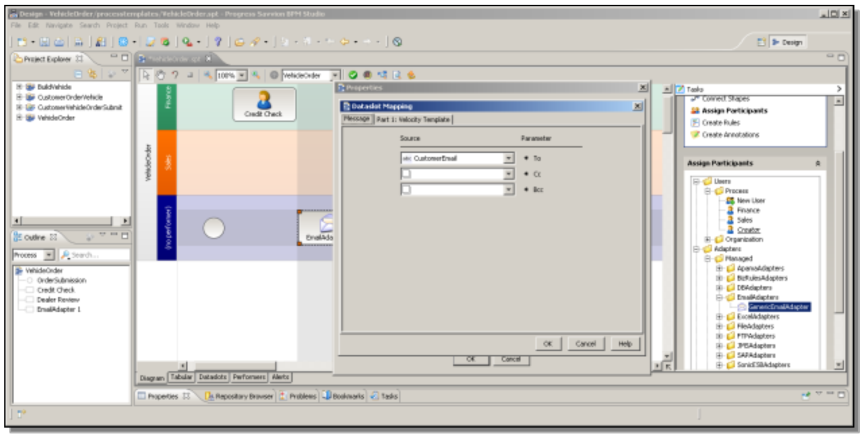
Sorry, but your order number $OrderNum for a new $VehicleBrand has been rejected because your credit limit of $CreditLimit is too low.

Regards,

$DealerName

CustomerName,OrderNum,VehicleBrand,CreditLimit,DealerName

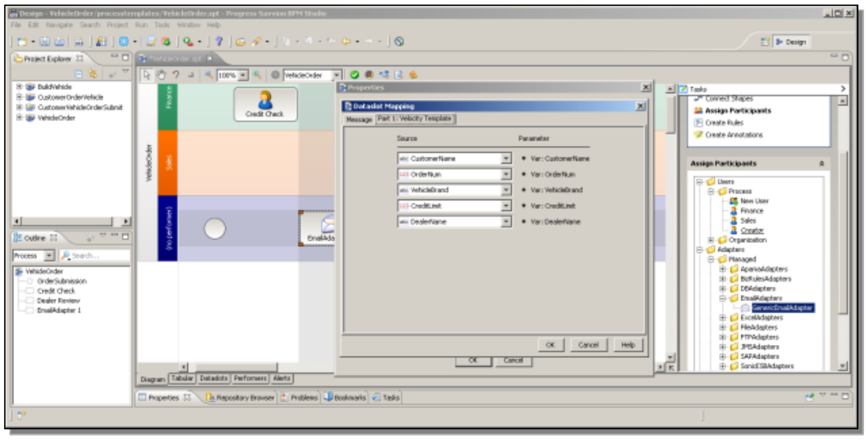
When I OK this, the adapter wizard first wants me to identify dataslots that will supply values for the To address, as well as optionally the CC and BCC addresses.



Source	Parameter
CustomerEmail	To
	Cc
	Bcc

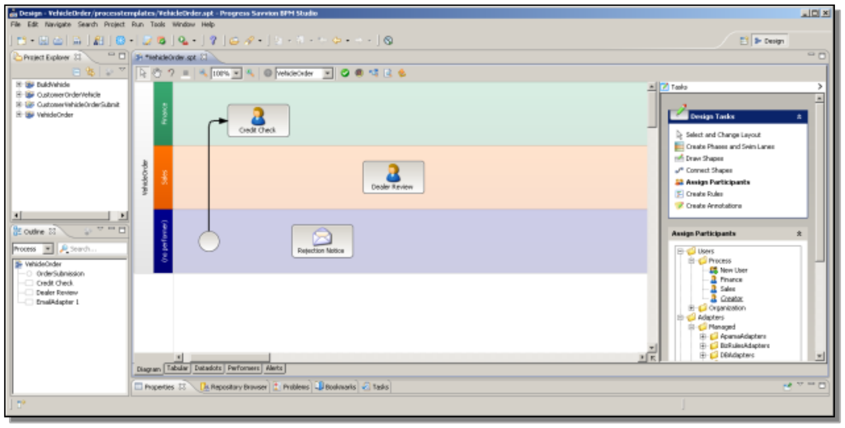
In this case, because I defined exactly one dataslot as an email address – remember that I specified that as a validation type – it has suggested that as the value to plug in, which is correct. That's the only one I need, so I select the **Velocity Template** tab. Velocity is a standard for embedding values into a boilerplate email template.

Here I see all the variables I defined with dollar signs in the message. Because I gave them all the same names as the Dataslots I want to fill them in with, the adapter wizard has filled them in for me; otherwise I could choose the right Dataslots here to map to my variables:

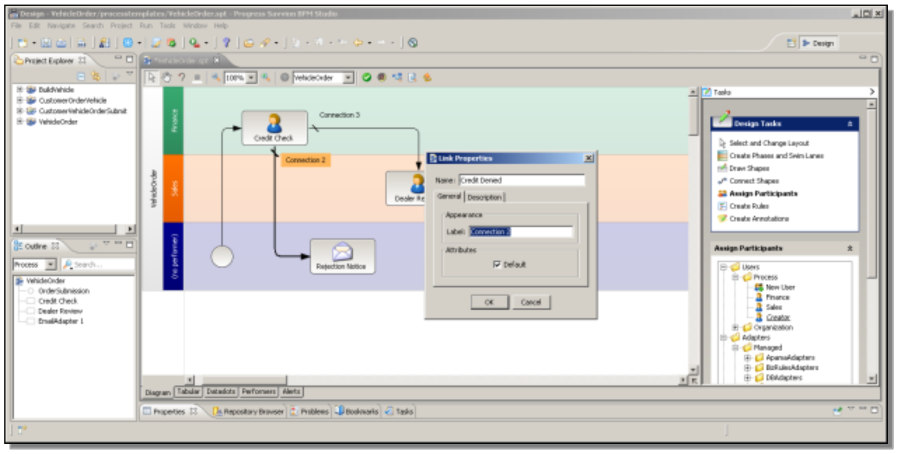


Source	Parameter
CustomerName	Var : CustomerName
OrderNum	Var : OrderNum
VehicleBrand	Var : VehicleBrand
CreditLimit	Var : CreditLimit
DealerName	Var : DealerName

That's all there is to defining a basic email adapter task. Let me scroll up here to get everything adjusted in the design window. Now I'm going to start to connect the different worksteps together. I begin with the Start step. I could click **Connect Shapes** over in the Design Tasks, but I use a shortcut here, which is to press the Ctrl key, then right-click and drag the mouse from one shape to the next. Here's my first link:

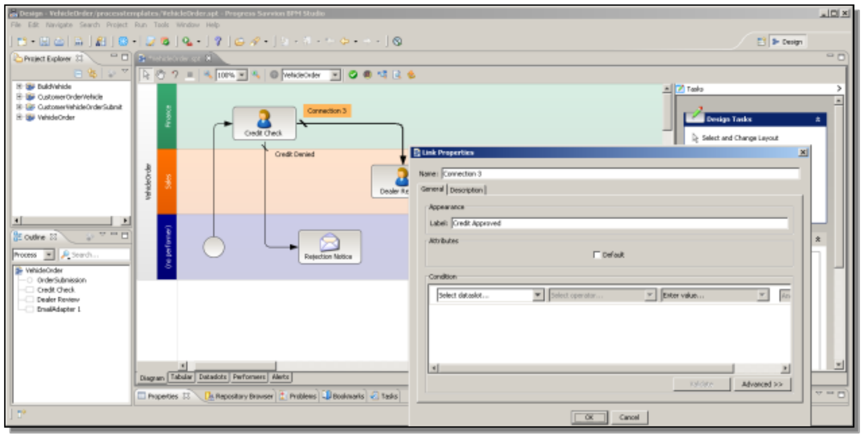


This tells the process that after the Start step is complete, it should continue with the next step, the Credit Check task. In this case the link is unconditional; the process always proceeds from the Start step to Credit Check. But from CreditCheck I'm going to define a couple of links, so there are different paths the process can take out of the Credit Check step. The first goes to the Rejection Notice email adapter. The other goes from Credit Check to the Dealer Review step, in the event that credit is approved. So I need to assign names and conditions to these two alternatives. The link to the rejection notice means credit was denied, so I enter that as the name and label.

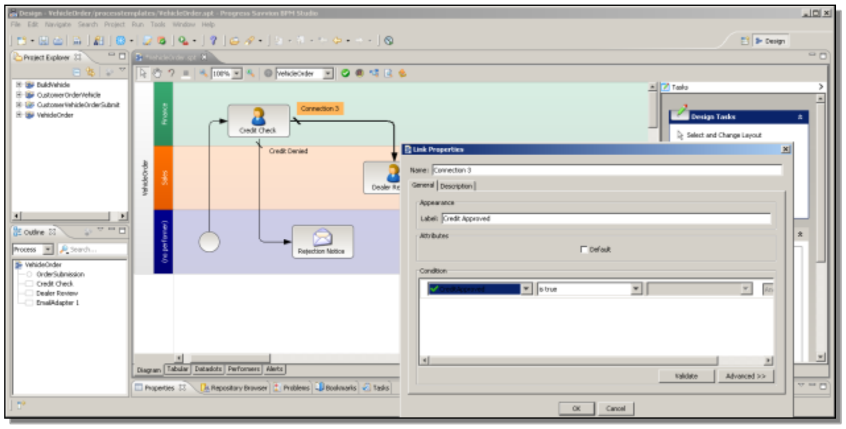


Note that there’s a checkbox here labeled **Default**. I leave that on, so that this will be the default path if some other condition isn’t true.

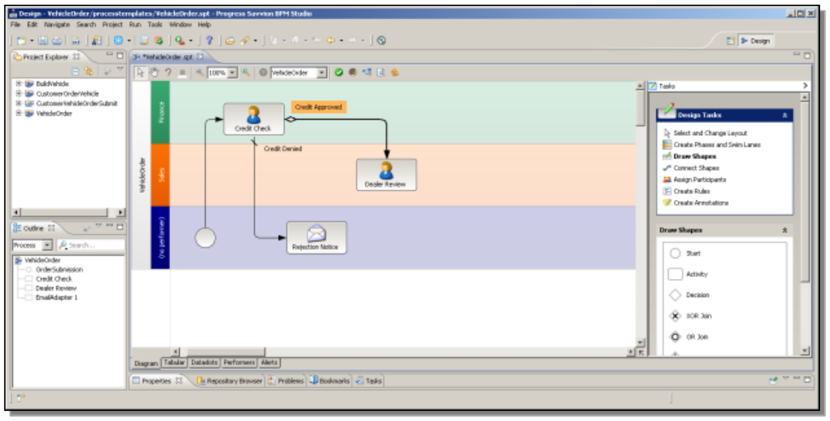
Notice also that there’s a slash through the link in the diagram. That indicates that this is the default path. I don’t want the other path to be the default, so I have to define a condition for it. In its properties, the label is really all I need to set. This is the path to take if credit is approved. If I uncheck Default, then this panel appears to get me to define a condition based on one or more Dataslot values:



I defined a Boolean dataslot called CreditApproved for just this purpose, and added it to the CreditCheck form so that the Finance person can set it. The condition of **Is True** is what I want, so I accept this:

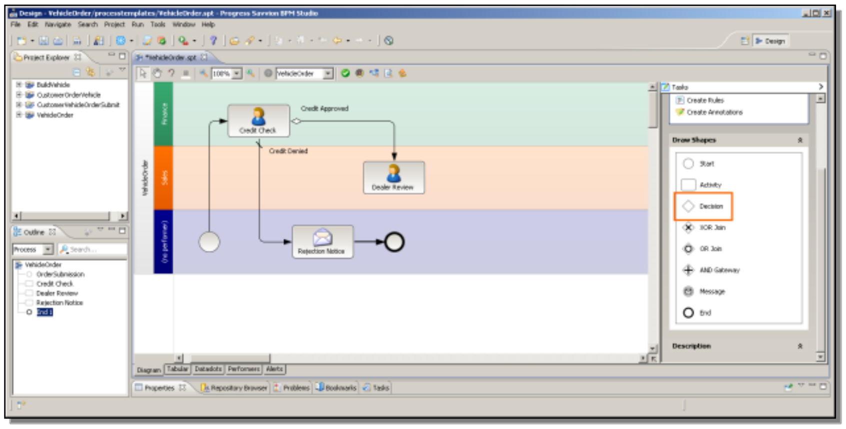


Below you can see that the Credit Approved link has changed. The slash is gone, so I now have only one default path. Instead, the start of the Credit Approved link is marked with a diamond shape, indicating that this is a path that will be taken only if a particular condition is true:

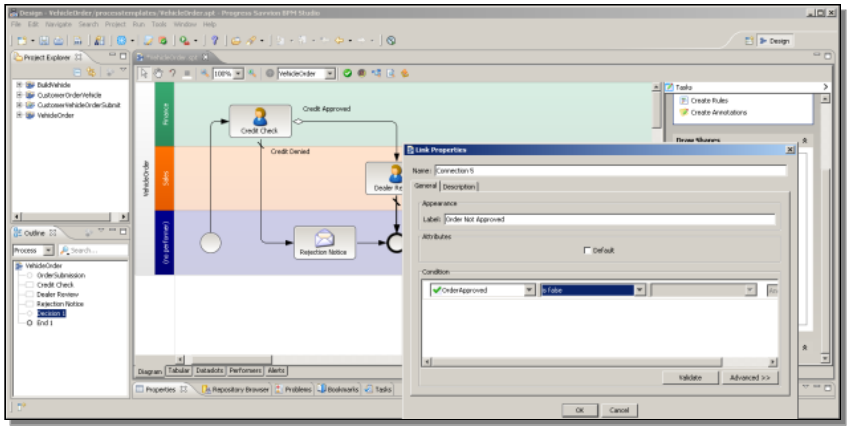


Now I need to select a shape I haven't used before, the **End** step. This marks an endpoint in the process. There can be more than one endpoint depending on the different paths through the process model. I drag that onto the diagram, drop it, and now I can indicate that the Rejection Notice email is the end of the line for a customer whose credit isn't approved, by connecting that to the end step.

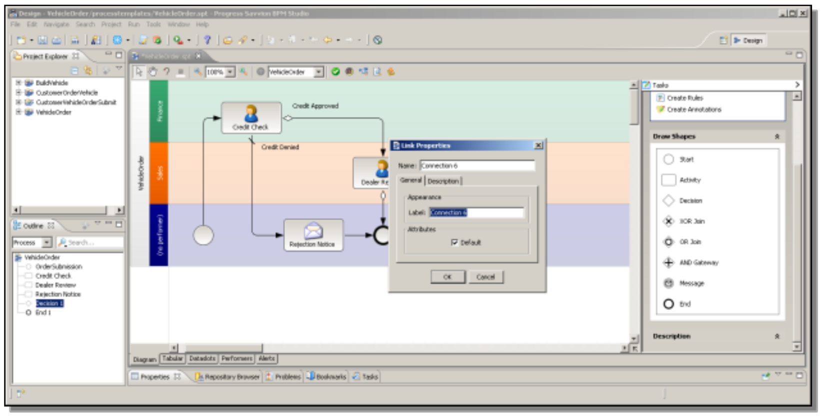
Now I've got some connections and a condition that is established as part of a workstep. But it can also be useful to define a condition that is really in between other worksteps, and that's what I'll do next. That's what this diamond-shaped BPMN shape is for, called a **Decision** gateway:



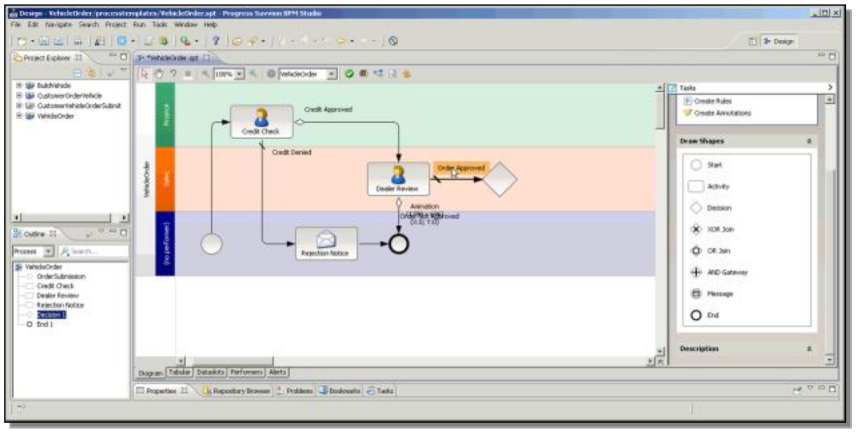
I drag one of those onto the diagram, and create some more connections. First a connector from the Dealer Review step to the End step. This is going to be the path taken if the salesperson at the Dealer does not approve the order for some reason. On the other hand, if the order is approved, I want to direct the model flow to this new Decision step, where I'm going to test another condition. The first of these new connectors I label **Order Not Approved**, and I'll set a condition for that, which is if the **OrderApproved** Boolean that is part of the DealerReview form is false:


Link Properties
Name: Connection 5
Label: Order Not Approved
Default [ ]
Condition: OrderApproved | is false

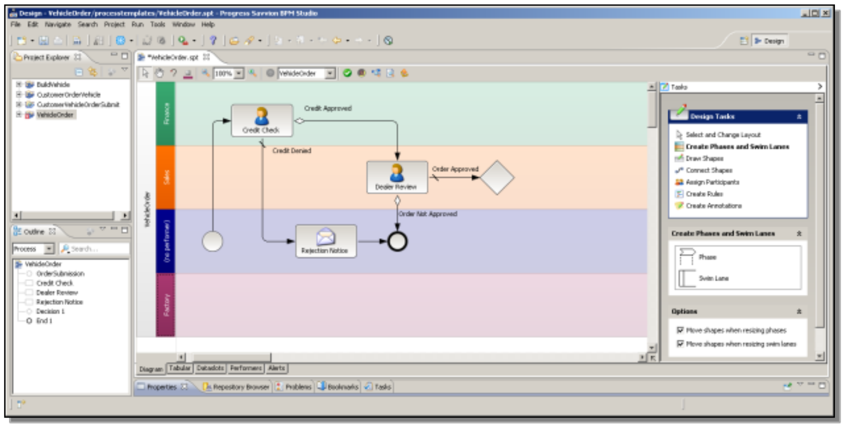
So in this case I'm making the positive outcome, which is that the Order is approved, the default:


Link Properties
Name: Connection 6
Label: Connection 6
Default [✓]

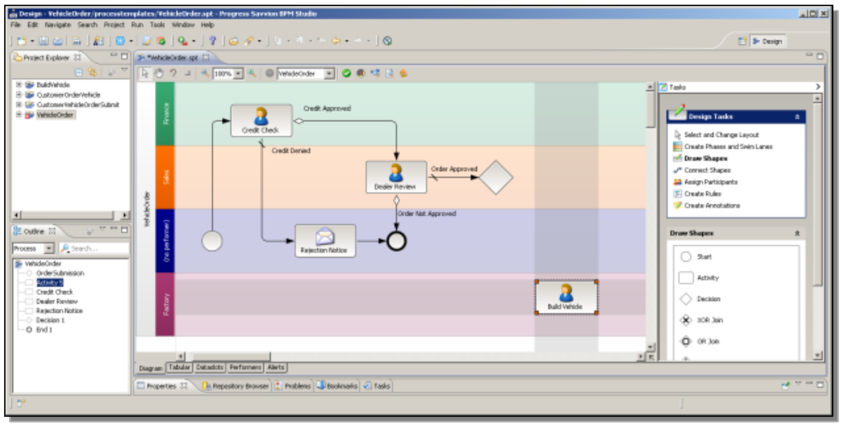
Of course, I can move these labels around, or any of the other elements in the model, to make it look clearer:



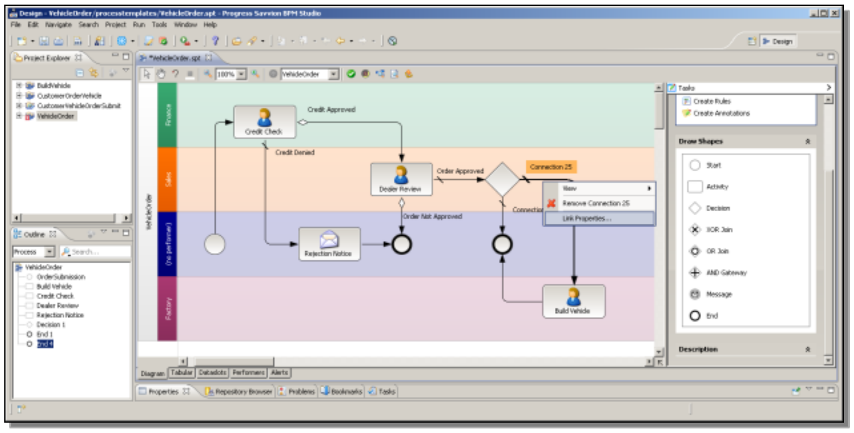
Now I'm ready to add the final workstep to my model. This step will represent the whole process of getting the factory to build the vehicle to order and deliver it to the dealer. In the full Factory application, this is done by a separate sub-process, but I'll just represent that with an Activity step. So here I've just added a Performer called **Factory** and a Factory swimlane to represent that process:



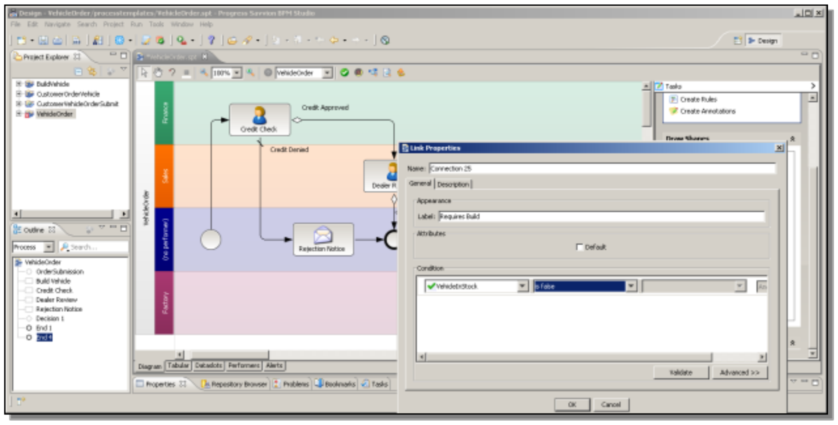
I drag another Activity onto the diagram, right-click to open its Properties, and name it **BuildVehicle**:



Now I can complete the last connections in my model. My Decision gateway is reached if the salesperson at the Dealer has approved the order. It is going to route the process onwards depending on whether the vehicle is in stock at the dealer or not. Once again pressing Ctrl and right-clicking the mouse starts a link. A link goes from the Decision gateway to the BuildVehicle step if the Vehicle isn't in stock. Remember that that's another dataslot that I defined for the DealerReview form. I then grab another End step, and bring that onto the form. It's perfectly valid to have more than one End step, just depending on the diagram layout. I connect the Decision step to the Endpoint, and I make a final connection from the BuildVehicle step to the endpoint. Now it's time for me to fill in the final conditions:

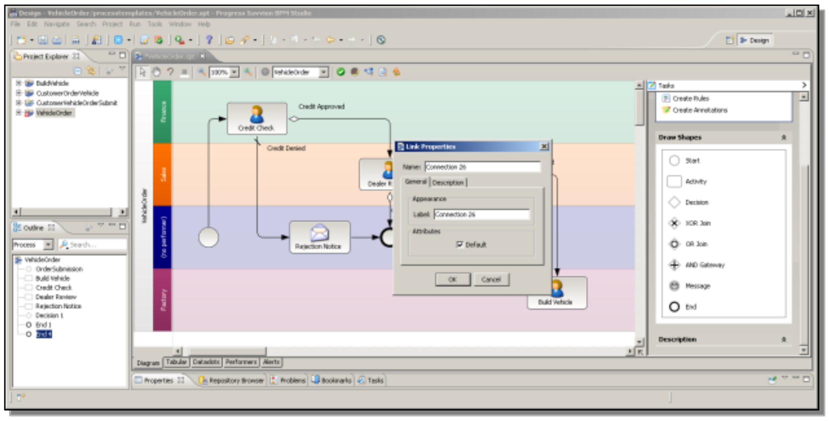


One link represents the path when VehicleInStock is false, so a custom Build is required:



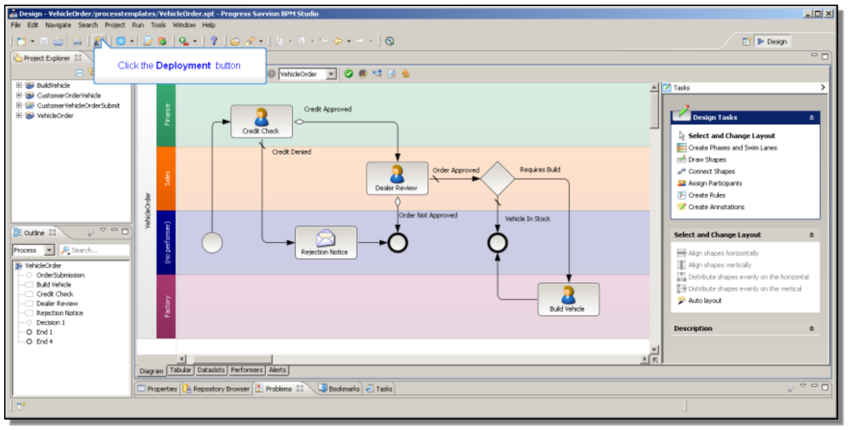
Name: Connection 25
Label: Requires Build
Default
Condition: VehicleInStock is False

So I set the condition where VehicleInStock is false. I go straight to this new End step if the Vehicle is in stock, so this becomes the default path:



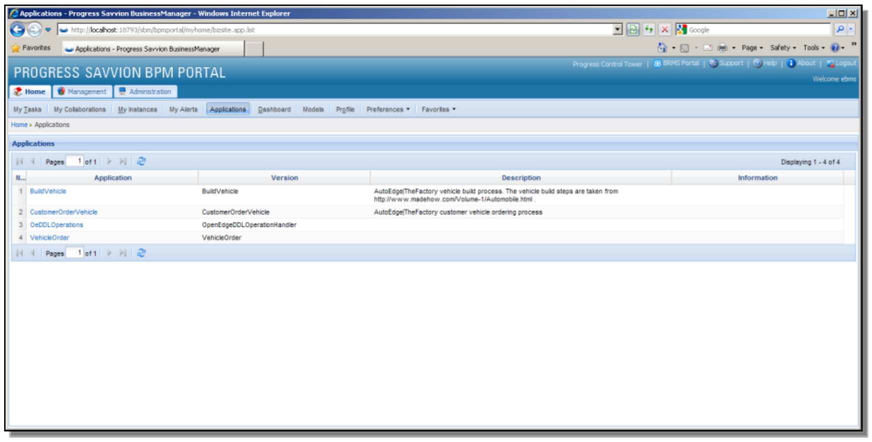
I clean up the position of my labels a little, and I'm done. I've created really a very substantial model here, with human and non-human participants, an email message, and various conditions to check that will send the process through different paths. Now it's time to deploy the model and test it out.

First, I just save the process model by pressing Ctrl-S or selecting File->Save from the main menu. A model is saved as a **.spt** file, for **Savvion Process Template**. I could save the model to a **Repository**, which is a database containing models and other data Savvion uses, but in this case I just save it as a File in my workspace. In order to run and test the model, I have to deploy it to the Savvion server that will host it for execution. That's a process I've started separately.



In order to connect to the server, I need a username and password. I use **ebms/ebms**, which is a standard Savvion admin username. Once I'm connected to the server, the deployment completes; my model has been deployed and is ready to test.

Now it's time to leave BPM Studio and show you a bit of what goes on in the **BPM Portal**, which supports all the runtime activities. I use the same **ebms** administration username and password that was used to deploy the process, which lets me control what goes on in the runtime environment.

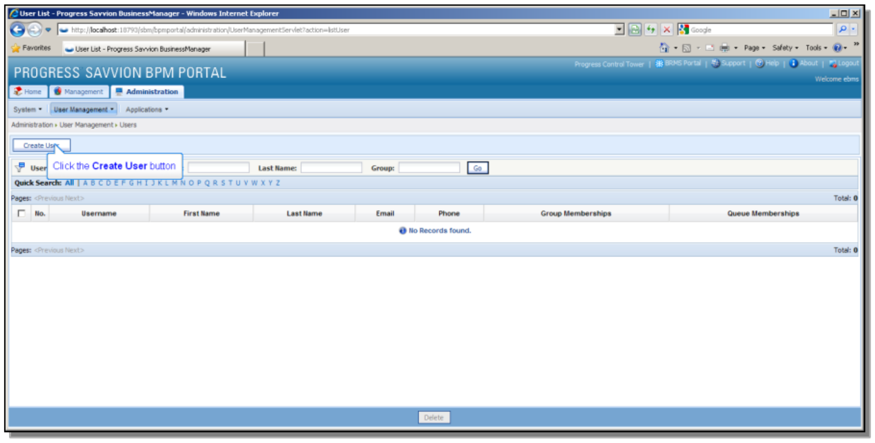


Applications

Displaying 1 - 4 of 4

N...	Application	Version	Description	Information
1	BuildVehicle	BuildVehicle	AutoEdge|TheFactory vehicle build process. The vehicle build steps are taken from http://www.madehow.com/Volume-1/Automobile.html .	
2	CustomerOrderVehicle	CustomerOrderVehicle	AutoEdge|TheFactory customer vehicle ordering process	
3	OeDDLOperations	OpenEdgeDDLOperationHandler		
4	VehicleOrder	VehicleOrder		

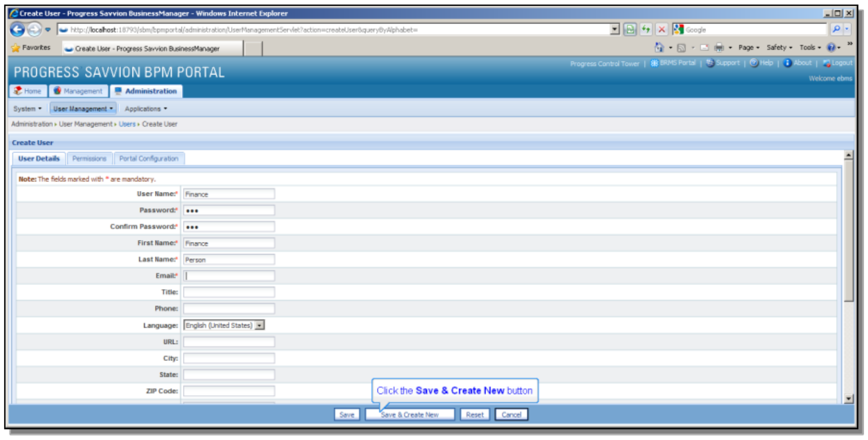
The first thing I need to do is make sure that the runtime environment supports the usernames that I used in defining the model. The ones that actually execute worksteps are **Finance** and **Sales**, so I define those. The ebms user has admin privileges, so I can select the **Administration** tab. The admin user can control all kinds of things here, including installing and uninstalling applications and so forth, but I select **User Management**. Then I select **Users** to create new users, and then the **Create User** button:


User List - Progress Savvion BusinessManager - Windows Internet Explorer
PROGRESS SAVVION BPM PORTAL
Home
Management
Administration
System
User Management
Applications
Administration › User Management › Users
Create User
Click the Create User button
Last Name:
Group:
Go
Quick Search: All | A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Delete

No.	Username	First Name	Last Name	Email	Phone	Group Memberships	Queue Memberships
No Records found.							

Total: 0

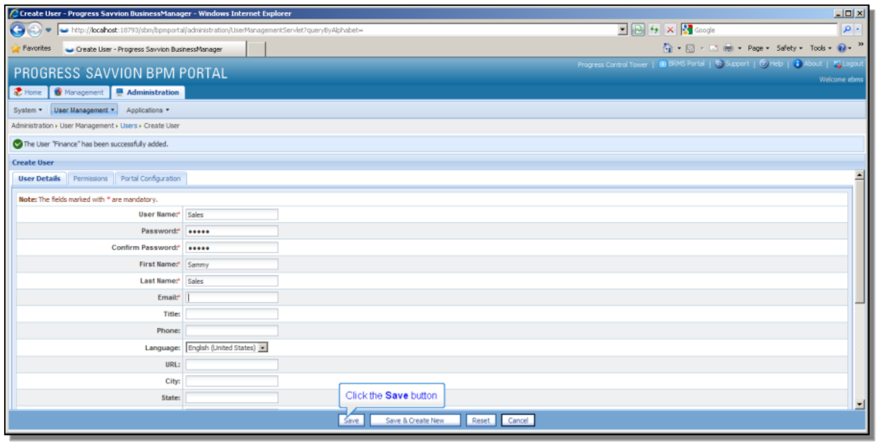
I need to fill in the first user name,.. **Finance**, plus a password, and identify the user by first and last names. Then I just make up an email address for now.


Create User - Progress Savvion BusinessManager - Windows Internet Explorer
PROGRESS SAVVION BPM PORTAL
Home
Management
Administration
System
User Management
Applications
Administration › User Management › Users › Create User
Create User
User Details
Permissions
Portal Configuration
Click the **Save & Create New** button
Save
Save & Create New
Reset
Cancel

Note: The fields marked with * are mandatory.

User Name:*	Finance
Password:*	•••
Confirm Password:*	•••
First Name:*	Finance
Last Name:*	Person
Email:*	
Title:	
Phone:	
Language:	English (United States)
URL:	
City:	
State:	
ZIP Code:	

Then I create a second new user profile for the **Sales** person:

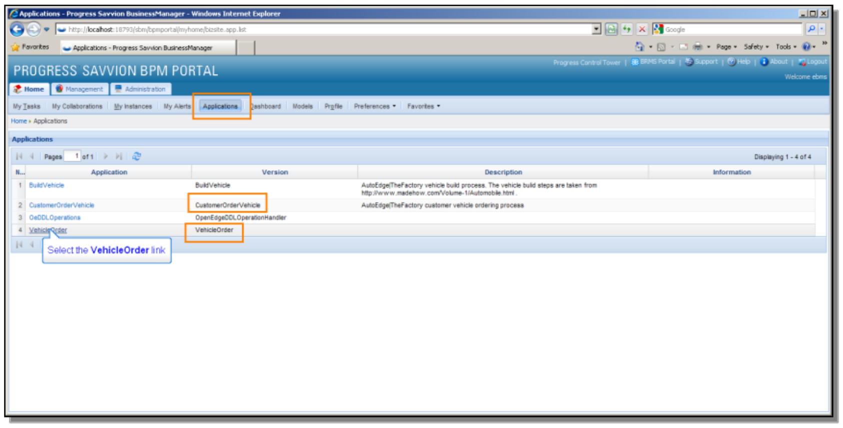

Create User - Progress Savvion BusinessManager - Windows Internet Explorer
PROGRESS SAVVION BPM PORTAL
Home
Management
Administration
System
User Management
Applications
Administration › User Management › Users › Create User
Create User
User Details
Permissions
Portal Configuration
Click the **Save** button
Save
Save & Create New
Reset
Cancel

The User "Finance" has been successfully added.

Note: The fields marked with * are mandatory.

User Name:*	Sales
Password:*	•••••
Confirm Password:*	•••••
First Name:*	Sammy
Last Name:*	Sales
Email:*	
Title:	
Phone:	
Language:	English (United States)
URL:	
City:	
State:	

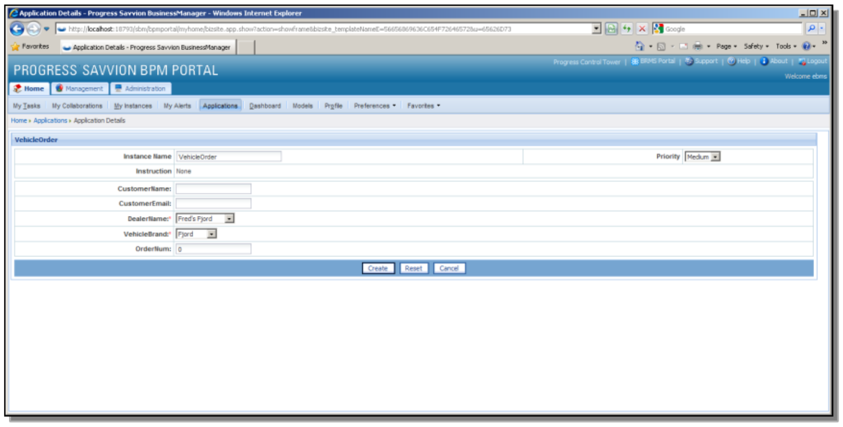
When I'm done, I select **Home** to get on with testing the new application. I then select **Applications** to see all the installed business process applications. Take a look at what's here:



N...	Application	Version	Description	Information
1	BuildVehicle	BuildVehicle	AutoEdge|TheFactory vehicle build process. The vehicle build steps are taken from http://www.madehow.com/Volume-1/Automobile.html .	
2	CustomerOrderVehicle	CustomerOrderVehicle	AutoEdge|TheFactory customer vehicle ordering process	
3	OeDDLOperations	OpenEdgeDDLOperationHandler		
4	VehicleOrder	VehicleOrder		

I'll show you just a few of the many capabilities that are exposed through this interface as I go along, but many of them will have to wait for other presentations. Here I am in the **Applications** tab. My VehicleOrder process is one of the applications listed, because I just deployed it. You can also see CustomerOrderVehicle, which is the complete Auto Edge – The Factory application that my example is a subset of. If I select the **VehicleOrder** link, then I'm starting an instance of my application.

In real life, the customer might initiate the application from a website, or by some other means; here, the administrator is starting an instance. We see the form come up that's associated with the Start step. Remember that this was an auto-generated form that I defined just as an ordered list of Dataslot Field values to be filled in:

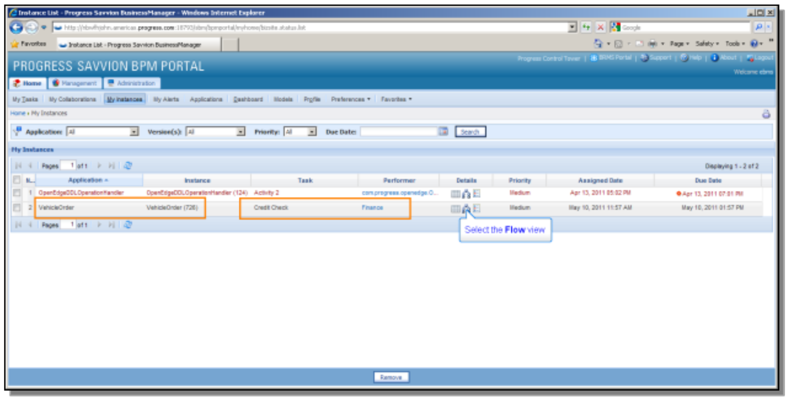

Application Details - Progress Savvion BusinessManager - Windows Internet Explorer
PROGRESS SAVVION BPM PORTAL
Home
Management
Administration
My Tasks
My Collaborations
My Instances
My Alerts
Applications
Dashboard
Models
Profile
Preferences
Favorites
Home › Applications › Application Details
Create
Reset
Cancel

VehicleOrder

Instance Name	VehicleOrder	Priority	Medium
Instruction	None		
CustomerName:			
CustomerEmail:			
DealerName:*	Fred's Fjord		
VehicleBrand:*	Fjord		
OrderNum:	0		

Let's suppose that the customer is running this, so they enter their name. I want the email message to really work if credit is denied, so I enter my own email address so I get the message. In a real application, of course, the customer would not assign an Order number, but I'll just do it that way to keep the model simpler. And when the customer's done, he clicks **Create**, one of the standard Start step buttons in the form footer. Now the process instance is really created, and off and running. An OK message is displayed when the Start step completes as a default confirmation, as you pass from step to step in the portal.

Back in the Portal as the administrator, I want to see what's going on, so I can select the **My Instances** tab, which will show all the running process instances that I own. And sure enough, here's the instance of VehicleOrder that I just created:



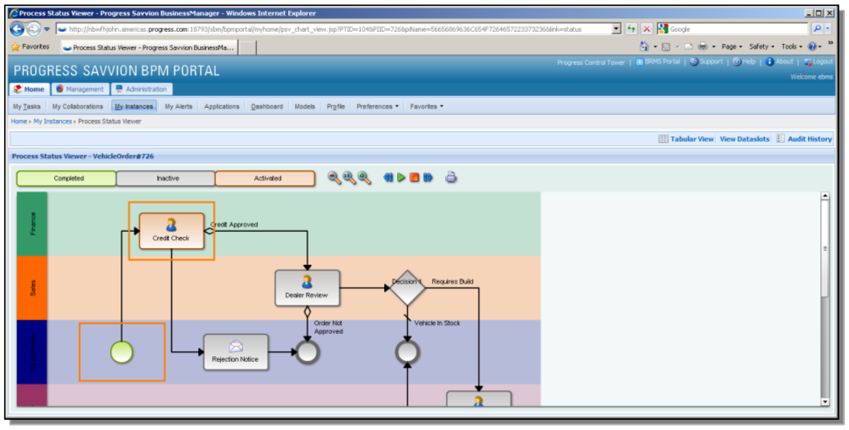
Application: All Version(s): All Priority: All Due Date:

My Instances

Displaying 1 - 2 of 2

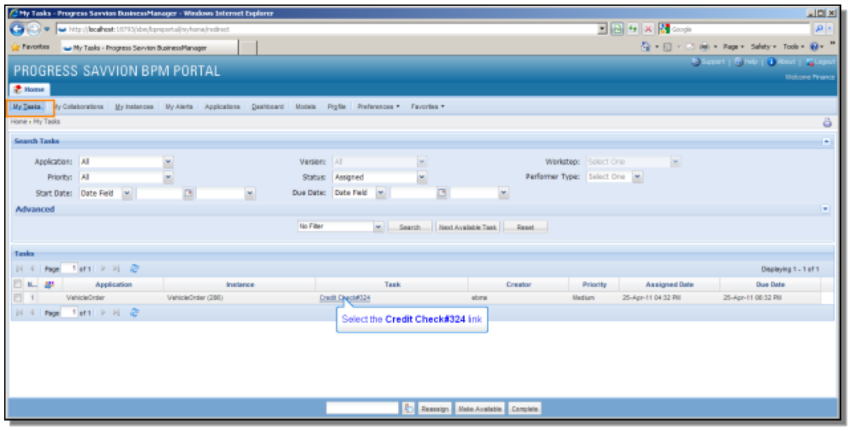
N..	Application	Instance	Task	Performer	Details	Priority	Assigned Date	Due Date
1	OpenEdgeDDLOperationHandler	OpenEdgeDDLOperationHandler (124)	Activity 2	com.progress.openedge.O...		Medium	Apr 13, 2011 05:02 PM	Apr 13, 2011 07:01 PM
2	VehicleOrder	VehicleOrder (726)	Credit Check	Finance		Medium	May 10, 2011 11:57 AM	May 10, 2011 01:57 PM

It's got its own instance number – there could be many instances of a process running concurrently, of course – and you can see that the next task to be completed is Credit Check. That's the task that's linked to directly from the Start step. You can also see that the Performer for this task is Finance. Now take a quick look at one of the process management tools available in the Portal, the **Flow View**:



This shows you the process diagram with color coding to tell you what's going on. In this case, the pink color of the Credit Check step confirms that this is the step in the application that has just been activated. The green color of the Start step tells me that it has been completed.

I go back to the instances list, and I now want to show what this will look like from the Finance employee’s perspective. The Finance people would be running their own BPM Portal, but so that I can reuse this one, I log out as the administrator, and log back in as Finance. This is a user that has already been defined as part of the Administration role in the portal. Remember that in a completed application Finance would presumably be a group of users, and each one could be assigned Finance tasks based on rules that we could define as part of the model. But here I am as the one Finance user:



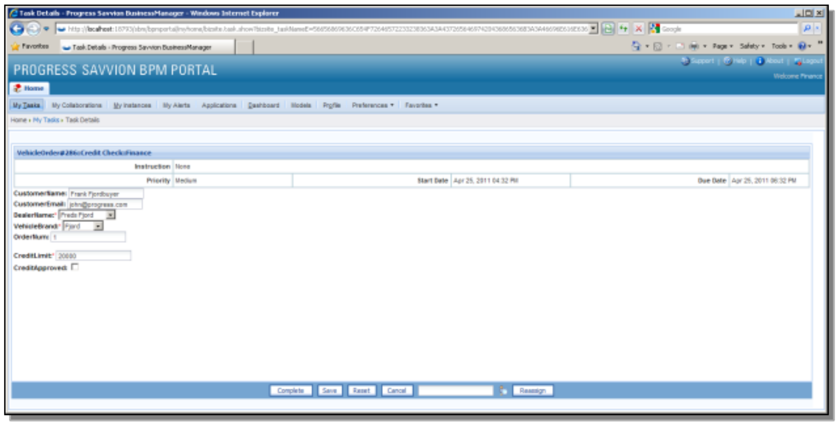
Application: All
Priority: All
Start Date: Date Field
Version: All
Status: Assigned
Due Date: Date Field
Workstep: Select One
Performer Type: Select One
No Filter

Page 1 of 1 Displaying 1 - 1 of 1

N...	Application	Instance	Task	Creator	Priority	Assigned Date	Due Date
1	VehicleOrder	VehicleOrder (286)	Credit Check#324	ebms	Medium	25-Apr-11 04:32 PM	25-Apr-11 08:32 PM

Select the **Credit Check#324** link

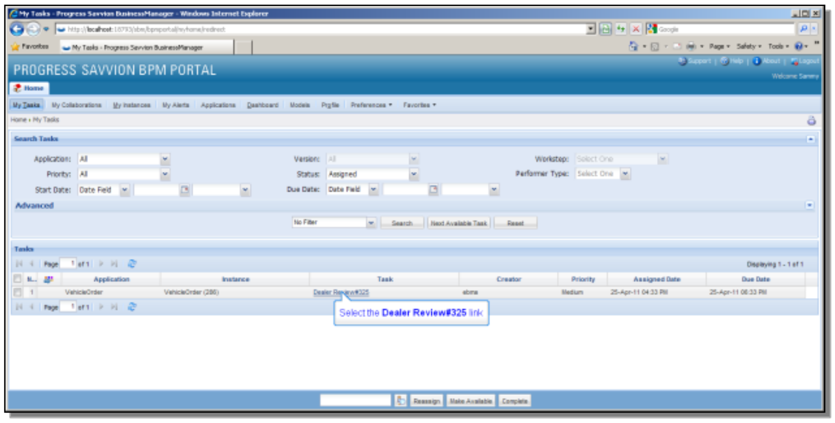
I’m in the **My Tasks** tab of the Portal. This shows all the tasks that have been assigned to me, and in my workday I would keep an eye on this tab and select all my tasks as they came in. So I select the Credit Check link to do the credit check for Fred Fjordbuyer. When I execute that step, the form I defined for the step comes up:


VehicleOrder#286::Credit Check::Finance

Instruction	None
Priority	Medium
Start Date	Apr 25, 2011 04:32 PM
Due Date	Apr 25, 2011 06:32 PM

CustomerName: Frank Fjordbuyer
CustomerEmail: john@progress.com
DealerName:* Freds Fjord
VehicleBrand:* Fjord
OrderNum: 1
CreditLimit:* 20000
CreditApproved: ☐

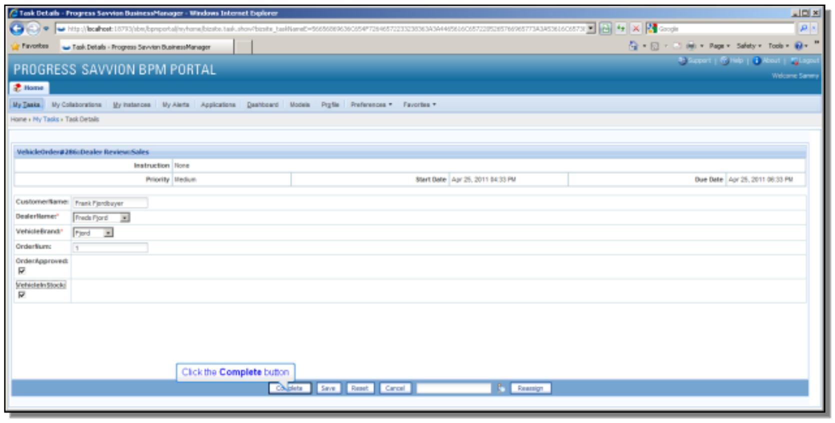
Somehow I’ve determined that this customer has good credit – the application isn’t complete, after all -- so I increase Fred’s credit limit to 25000, and I check the checkbox on to indicate that Fred’s credit is approved for this purchase. That’s all I have to do, so I click **Complete**. Now the Finance person is back to waiting for the next Finance task to come in. Meanwhile, there are salespeople at the dealerships waiting for their own tasks to be assigned. Let’s go see this process from their perspective. I log back in to the Portal as Sales, and in the My Tasks tab in the Sales view of the Portal, I see tasks assigned to Sales. Because the Finance guy approved the customer’s credit, the process followed the **Credit Approved** link to the **Dealer Review** step. I can select that task:


Search Tasks

Application: All
Priority: All
Start Date: Date Field
Version: All
Status: Assigned
Due Date: Date Field
Workstep: Select One
Performer Type: Select One

N...	Application	Instance	Task	Creator	Priority	Assigned Date	Due Date
1	VehicleOrder	VehicleOrder (286)	Dealer Review#325	ebms	Medium	25-Apr-11 04:33 PM	25-Apr-11 06:33 PM

Now the Dealer Review form comes up. The only thing to do here is to decide, on whatever basis, whether to approve the order or not, and then to check to see whether the vehicle the customer wants is in stock or not. I’ll click the checkbox to indicate that it is, and this task is complete:



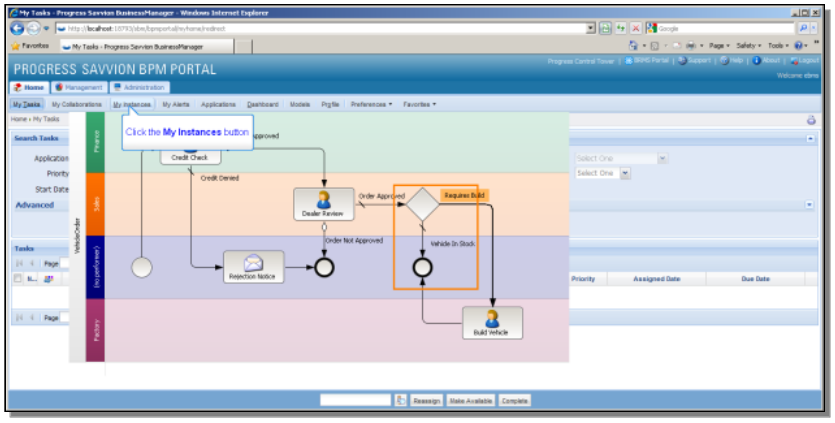
PROGRESS SAVVION BPM PORTAL

VehicleOrder#286:Dealer Review:Sales

Instruction	None				
Priority	Medium	Start Date	Apr 25, 2011 04:33 PM	Due Date	Apr 25, 2011 06:33 PM

CustomerName:	Frank Fjordbuyer
DealerName:*	Freds Fjord
VehicleBrand:*	Fjord
OrderNum:	1
OrderApproved: ✓	
VehicleInStock: ✓	

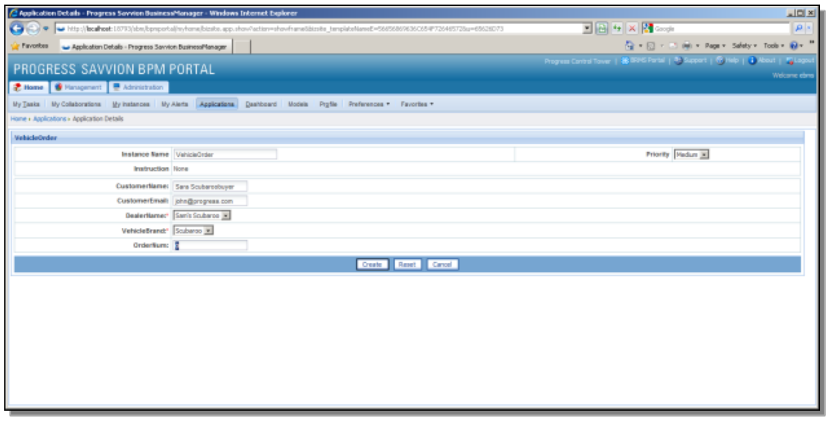
To go back to the administrator’s view of what’s going on, I log out and back in again as the administrator. Take another look at the process model from BPM Studio:



Since this is a simplified example, nothing happens after Dealer Review if the vehicle is in stock. The process just goes to an End step. If that’s true, then the instance should have completed and will be gone from the instance list.

And indeed, when I select **My Instances**, the process instance is not there any more. For our purposes, that represents success. The model works, or at least that path through the model does.

Now I go back to the Applications tab and start up another instance of VehicleOrder. Here’s the Start step form:

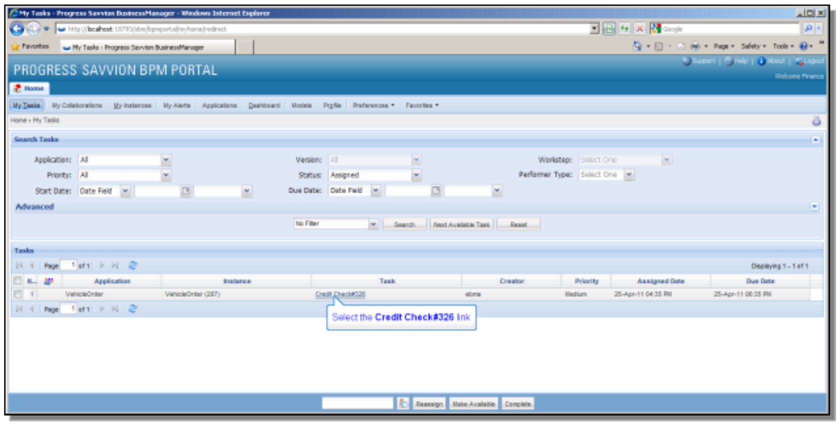


VehicleOrder

Instance Name	VehicleOrder
Priority	Medium
Instruction	None
CustomerName:	Sara Scubaroobuyer
CustomerEmail:	john@progress.com
DealerName:*	Sam's Scubaroo
VehicleBrand:*	Scubaroo
OrderNum:	2

This time Sara tries to order a Scubaroo. I have her enter my own email address, which this time will be significant. From the list of dealers, she selects Sam's Scubaroo dealer. And from the list of Vehicles, of course she selects Scubaroo. She enters another order number to her liking, and she's done.

Let's go back to the Finance portal to see what happens there. Here's the new instance of the Credit Check task:

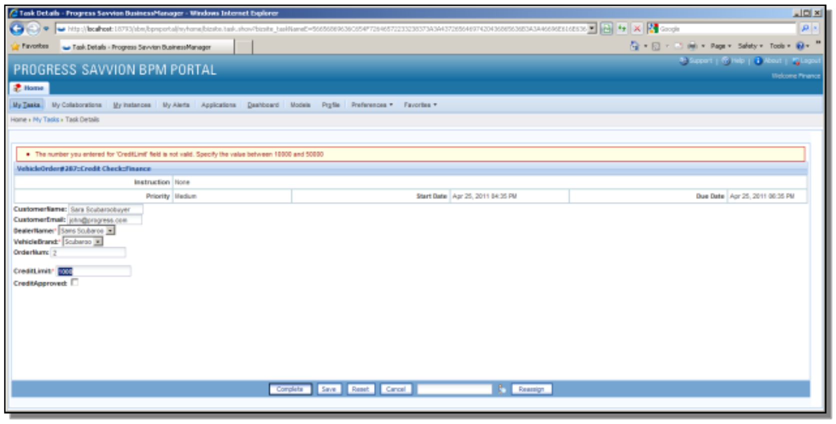


Application: All
Version: All
Workstep: Select One
Priority: All
Status: Assigned
Performer Type: Select One
Start Date: Date Field
Due Date: Date Field

Tasks — Displaying 1 - 1 of 1

N...	Application	Instance	Task	Creator	Priority	Assigned Date	Due Date
1	VehicleOrder	VehicleOrder (287)	Credit Check#326	ebms	Medium	25-Apr-11 04:35 PM	25-Apr-11 08:35 PM

This time, the Finance person determines that Sara's credit is lacking. So he tries to set it all the way down to 1000 dollars. But remember that when I defined the CreditLimit dataslot, I set the valid range as 10000 to 50000. So the value of 1000 is rejected. This is part of Savvion's support for its own forms:



PROGRESS SAVVION BPM PORTAL

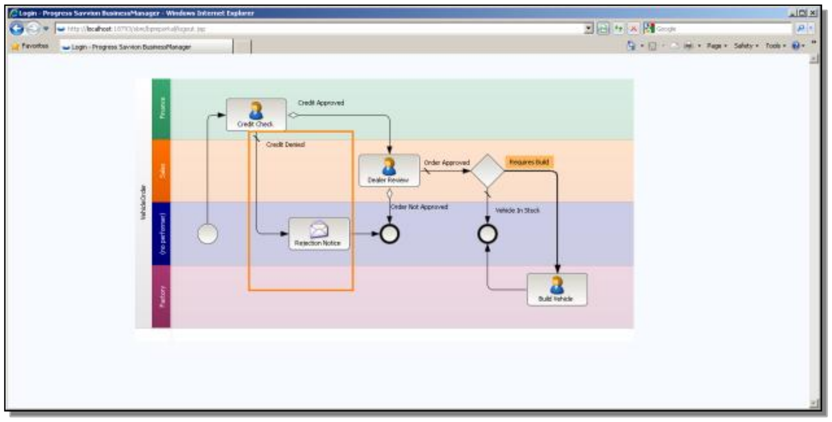
- The number you entered for 'CreditLimit' field is not valid. Specify the value between 10000 and 50000

VehicleOrder#287::Credit Check::Finance

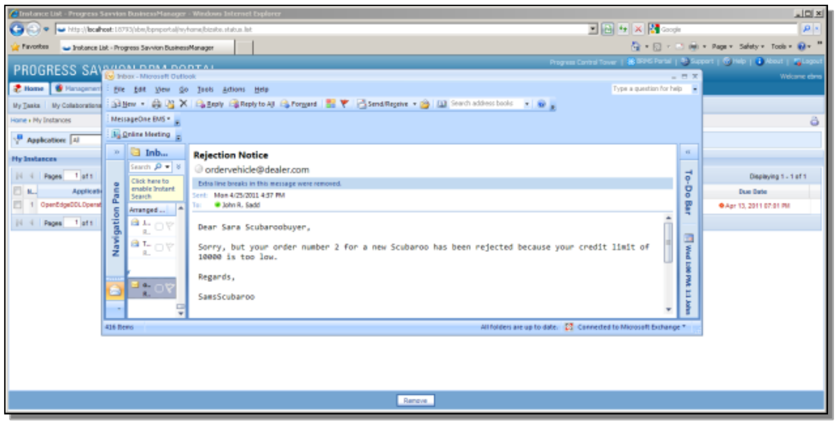
Instruction	None				
Priority	Medium	Start Date	Apr 25, 2011 04:35 PM	Due Date	Apr 25, 2011 06:35 PM

CustomerName: Sara Scubaroobuyer
CustomerEmail: john@progress.com
DealerName:* Sams Scubaroo
VehicleBrand:* Scubaroo
OrderNum: 2
CreditLimit:* 1000
CreditApproved: ☐

I set the CreditLimit to 10000 instead, but I still don't check the CreditApproved box. After I click Complete, is there any point to logging in as sales and looking for a Dealer Review task? Take another look at the process diagram, and you'll see that if the Finance person doesn't check on CreditApproved, then the process goes to the Rejection Notice email step, and then ends. So I hope that's what happened in this case:



Let me log back in as the administrator. I go back to My Instances, and I see that the instance is gone. It has completed, which I hope indicates that it took the right path through the process model. If I look in my inbox for an email message, I see one:



Rejection Notice
ordervehicle@dealer.com
Extra line breaks in this message were removed.
Sent: Mon 4/25/2011 4:37 PM
To: John R. Sadd

```
Dear Sara Scubaroobuyer,

Sorry, but your order number 2 for a new Scubaroo has been rejected because your credit limit of 10000 is too low.

Regards,

SamsScubaroo
```

All the variables have been replaced with the Dataslot values from this application instance.

So my process works, and that brings me to the end of this paper on building your first process model in BPM Studio. I've covered a lot of territory here, and really shown you at least a bit about many of the key features of BPM Studio as a design tool, and at least an introduction to the BPM Portal as a runtime tool. Additional presentations will go into much more detail about many aspects of both the design, as well as the execution and testing process, focusing of course on what you need to know to use Savvion in conjunction with your OpenEdge applications.