

Laboratorio 3

Contenidos

Objetivos.....	2
Creación Process Model	2
Configuración del Process Model	2
Añadir input task y configurar	4
Añadir Script Task.....	7
Añadir writeDataStore y configurar.....	9

Objetivos

En este laboratorio crearemos nuestro modelo de proceso (BPM), que servirá para determinar qué hacer con los vehículos que se deseé incorporar a nuestro sistema.

Según ciertos criterios se tomarán unos caminos u otros.

Nota: BPM es una forma estructurada, coherente y consistente de comprender, documentar, modelar, analizar, ejecutar, monitorizar y optimizar los procesos de negocios, así como los recursos asociados que conducen a la mejora de los negocios.

Creación Process Model

Descripción:

Crearemos nuestro Process Model, con el que conseguiremos guardar en base de datos el vehículo.

Pasos a realizar:

- Dentro de nuestra aplicación XXX_Vehicle Fleet Management, en la carpeta XXX_Process creada con anterioridad, pulsamos New -> Process Model.
- **Name:** XXX_AddVehicle
- **Description:** Process Model para para tratar el alta de vehículos
- **Save in:** XXX_Process
- Pulsamos Create.

Configuración del Process Model

Descripción:

Una vez creado nuestro proceso XXX_Add_Vehicle, pulsamos sobre él para que se nos abra el entorno Appian Process Modeler para implementarlo.

En primer lugar, procederemos a configurar su información propia.

Pasos a realizar:

- En el menú superior, pulsamos sobre el ícono de Properties. Se nos abrirá una ventana emergente, con varias pestañas.
- Pestaña Variables: Son las variables que necesitaremos manejar dentro de nuestro proceso. Pulsando sobre la opción “Add Variable” añadiremos las siguientes variables:
 - supervisorComment, donde guardaremos los comentarios del supervisor.

- approveDecision, donde recogeremos la decisión de si se acepta o no el vehículo.
- counter, variable que usaremos como contador.
- vehicle, donde guardaremos el vehículo.

Ninguna de estas variables las usaremos como parámetro de entrada de nuestro proceso.

Al finalizar el alta, se nos deben quedar nuestras variables como se muestra en la siguiente imagen:

Process Variables							
+ Add Variable							
Name	Type	Value	Parameter?	Required?	Multiple?	Hidden?	
SupervisorComment	Text	(No Value)	No	N/A	No	No	X
approvalDecision	Text	(No Value)	No	N/A	No	No	X
counter	Number (Integer)	0	No	N/A	No	No	X
vehicle	Open_vehicle	(No Value)	No	N/A	No	No	X

Para hacer uso de estas variables en nuestro proceso, únicamente tendremos que escribir `pv!` (process variables). Por ejemplo, para hacer referencia a la marca del vehículo, usaremos `pv!vehicle.make`

Nota: en nuestro caso, el tipo de la variable `vehicle` será de nuestra CDT creada en el laboratorio 1, apartado 5.CDT. Por tanto, su nombre a buscar será `XXX_Vehicle`. Si nos pudimos completar dicho apartado del laboratorio 1, podemos usar la CDT `Open_Vehicle`.

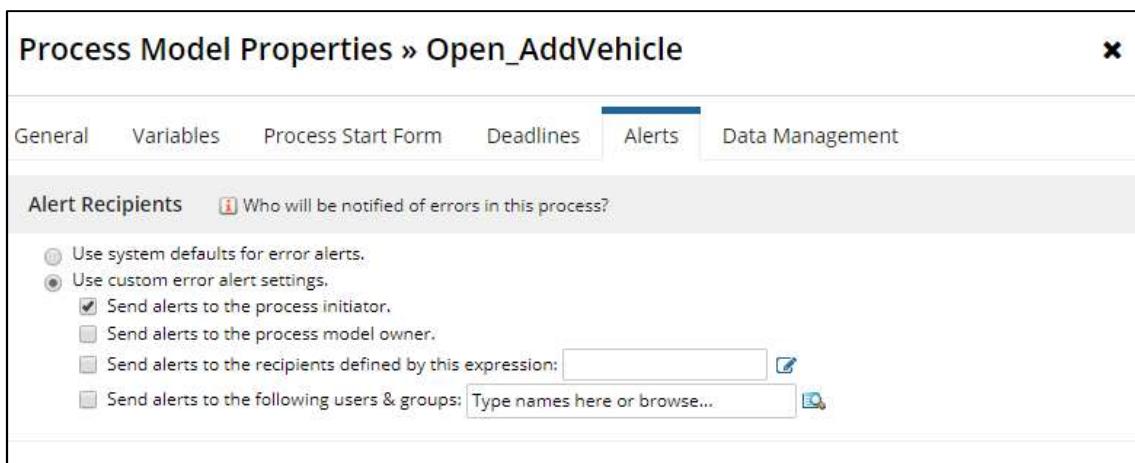
- Pestaña General. En esta pestaña se recogen datos generales del proceso. Únicamente modificaremos un par de campos de esta pestaña:
 - **Process Model Name:** XXX_AddVehicle
 - **Process Display Name:** ="Add Vehicle License Plate " & `pv!vehicle.licensePlate`

Nota: el signo igual también debe ser incluido en el campo Process Display Name.

Este campo `Process Display Name` es el que usaremos posteriormente en las pruebas para localizar nuestro proceso en la monitorización de las pruebas. Por esto, mostramos la matrícula del vehículo introducido, para así tener una fácil identificación de cada proceso lanzado.

- Pestaña Alertas

Debemos configurarla para que el resultado sea el de la imagen siguiente:



Con esto, conseguimos que, en el caso de producirse un error cuando se lanza algún proceso, se envíe correo de alerta al usuario que lanzó la ejecución

- Pulsamos OK para guardar los cambios

Añadir input task y configurar

Descripción:

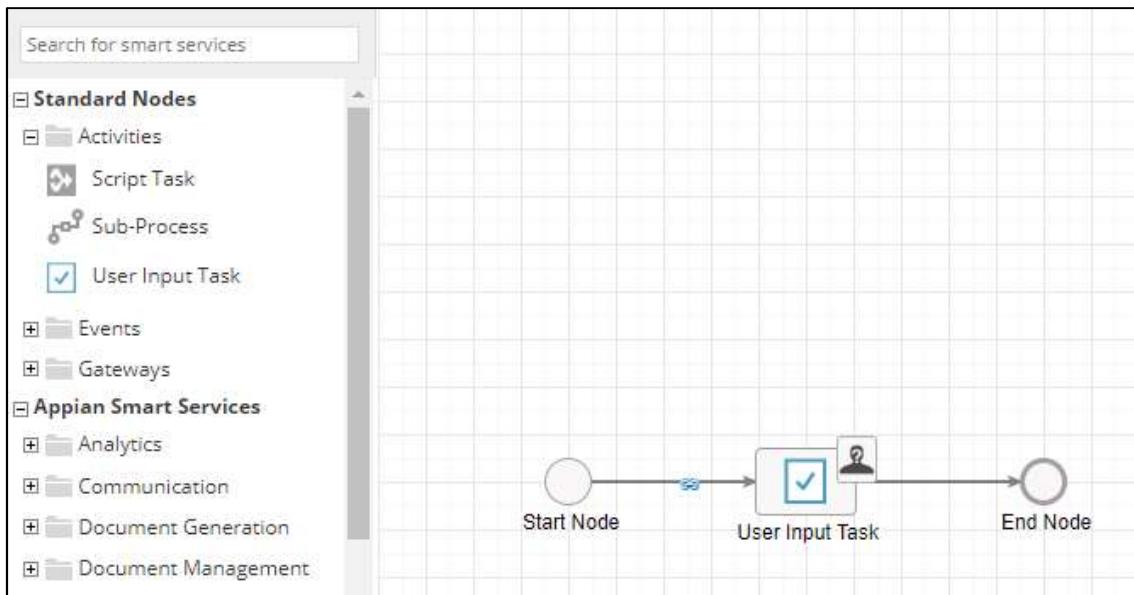
Añadiremos en nuestro flujo, como primer paso, el interfaz que hemos definido en el laboratorio anterior (XXX_AddVehicleForm).

Pasos a realizar:

- En el menú de la izquierda (Palette), localizar el nodo denominado “User Input Task” (se encuentra dentro en Standard Nodes – Activities. También existe sobre el árbol un campo a modo de buscador) y arrastrarlo al panel central (canvas), justo sobre la línea que une los eventos Start y End.
- Hacerlo de la forma siguiente:



- Pulsar sobre la línea que une Start Node con User Input Task, y marcar la propiedad de “Enable activity chaining” a YES. Esta propiedad permite que el usuario pueda continuar en la misma ventana entre la inicialización del proceso y la apertura del interfaz. El resultado debe ser el siguiente:



- Si pulsamos con el botón derecho sobre el ícono recién añadido, Properties, se nos muestra una ventana emergente con varias pestañas, con todo lo que podemos gestionar sobre dicho componente.
- Pestaña General. Contiene la información básica del componente: name, prioridad, etc.
 - **Name:** Add Vehicle
 - **Task Display Name:** =if(isnull(pv!vehicle), "Add Vehicle", "More Info LP " & pv!vehicle.licensePlate)
 - Este campo Task Display Name es el que usaremos posteriormente para localizar nuestro proceso entre las tareas que tengamos asignadas.
 - Con la regla incluida, la propiedad Task Display Name, tomará valor “Add Vehicle” cuando estemos en el proceso de alta (nuestra variable vehicle aún es nula), o tendrá valor “More Info LP” + licensePlate cuando estamos revisando un vehículo
- Pestaña Forms. En esta pestaña indicaremos qué interfaz será el que se mostrará en este punto. En nuestro caso, XXX_AddVehicleForm. Debemos indicar que sí queremos que nos genere automáticamente los inputs de entrada necesarios.

General Data Forms Scheduling Assignment Escalations Exceptions Other

Edit Form

English (US)

Enable this language

Select an interface Write an expression

"Open_AddVehicleForm" Clear

EDIT INTERFACE

Specify the data to pass to the interface Refresh

Rule Input	Type	Multiple?	Value
vehicle	Open_vehicle	No	vehicle

Allow users to save a draft of in-progress tasks

- Pestaña Data. Al acceder a la pestaña, observaremos que se ha creado 1 inputs de entrada. Debemos modificarle los campos necesarios para que sus propiedades queden como sigue:

General Data Forms Scheduling Assignment Escalations Exceptions Other

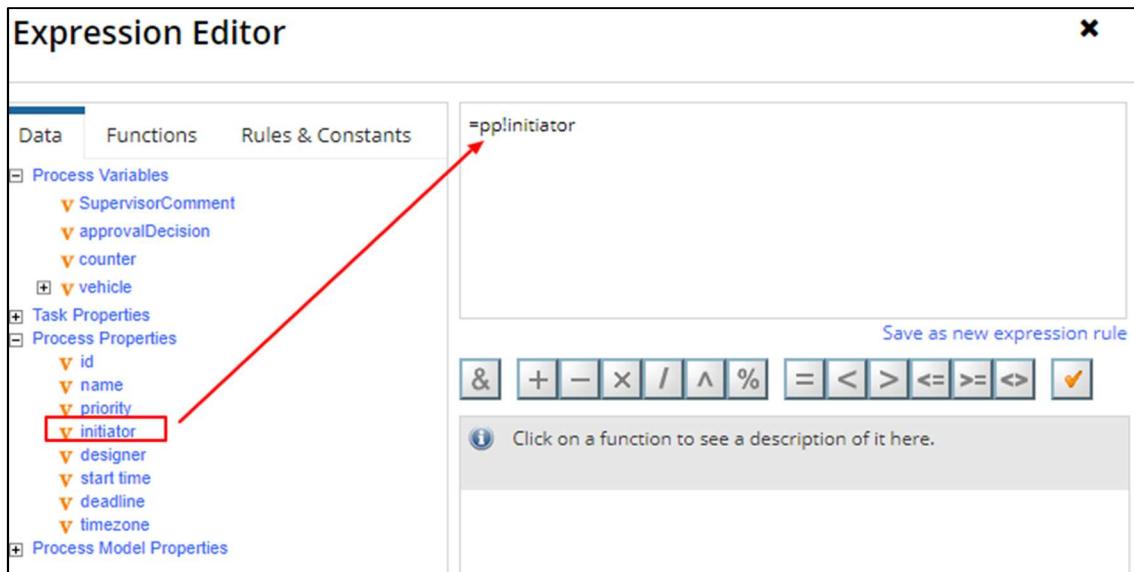
Inputs Outputs

Node Inputs ! Map the value(s) for the inputs of the node

+ New Input ✖ Delete Input	Field Properties
- vehicle (Open_vehicle) <ul style="list-style-type: none"> - id (Number (Integer)) - make (Text) - model (Text) - licensePlate (Text) - category (Text) - vehicleCondition (Text) - lastUpdated (Date and Time) - nextServiceDate (Date) - mileage (Number (Integer)) - pictureId (Number (Integer)) 	Name: vehicle Type: Open_vehicle Multiple: <input type="checkbox"/> Value: =pv!vehicle Required: <input type="checkbox"/> Save into: vehicle

Nota: el campo Type debe ser nuestro CDT creado, esto es, XXX_Vehicle. El valor que tomará será el de la variable del proceso definida anteriormente (pv!vehicle).

- Pestaña Assignment. Se indica a quién se asignará la tarea. En nuestro caso, se la asignaremos a la misma persona que inició el proceso. Para ello, dentro de la pestaña Assignment, en el campo Assign to the following, pulsando en el icono "From Expression" que se muestra a la derecha del campo, incluir lo siguiente:



- Pulsamos OK, que aparece en la parte inferior.

Hasta el momento, hemos creado nuestro proceso BPM, al que le hemos definido, entre otras, una variable global de tipo XXX_Vehicle, que estará visible durante todo el proceso.

Al proceso BPM le hemos incluido un componente de tarea humana (formulario), para que el usuario pueda introducir los datos del vehículo.

Dicho interfaz, tiene definido como parámetro de entrada (Rule Input) un objeto de tipo XXX_Vehicle. Su valor lo relacionamos con la variable global del BPM del mismo tipo.

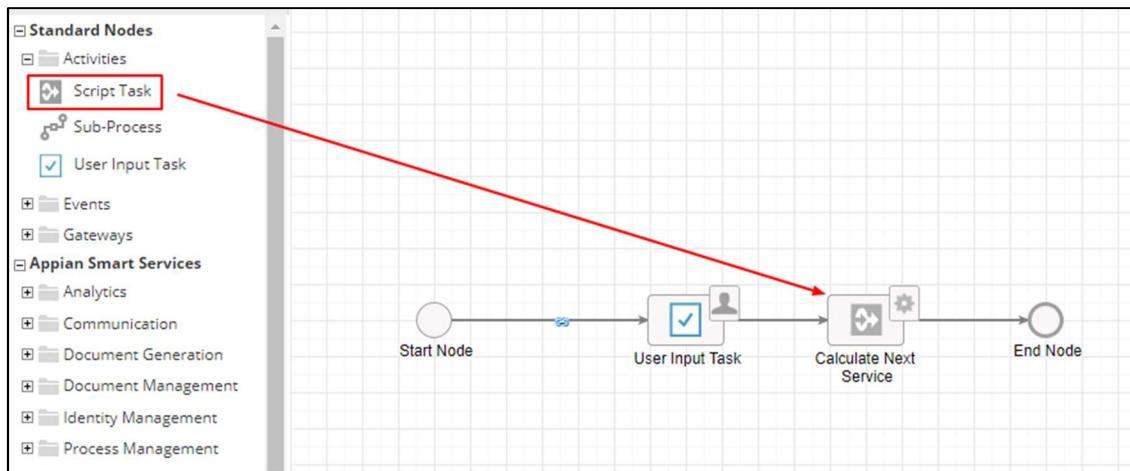
Añadir Script Task

Descripción:

Incluimos un componente Script Tasks para calcular cuando el vehículo tendrá que pasar la próxima revisión.

Pasos a realizar:

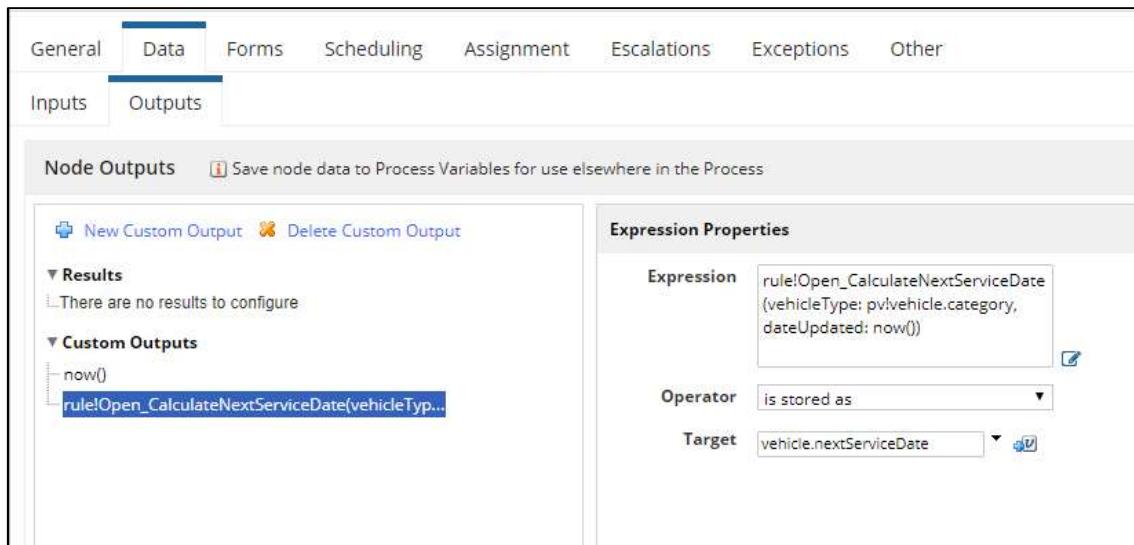
- En el menú de la izquierda, localizar el componente "Script Task" (se encuentra dentro en Standard Nodes – Activities) y lo arrastramos sobre la línea que une el componente Input Task y el evento End. El resultado será el siguiente:



- Al pulsar sobre el componente con el botón derecho – properties se nos mostrará una ventana emergente con varias pestañas. Completaremos la siguiente información:
- Pestaña General
 - **Name:** Calculate Next Service
 - **Display Name:** Calculate Next Service
- Pestaña Data - Node Outputs. Añadiremos también un par de outputs de salida, pulsando el botón New Custom Output:
- Actualizaremos el campo lastUpdated del vehículo (última actualización) al momento actual.

This screenshot shows the 'Data' tab of the node properties for the 'Calculate Next Service' node. The 'Outputs' sub-tab is selected. The 'Node Outputs' section contains a 'New Custom Output' button and a 'Delete Custom Output' button. Below these are sections for 'Results' (empty) and 'Custom Outputs'. Under 'Custom Outputs', there is a list with one item: 'now()' under 'rule!Open_CalculateNextServiceDate(vehicleTyp...'. To the right, the 'Expression Properties' panel is visible, showing an 'Expression' field with 'now()', an 'Operator' field set to 'is stored as', and a 'Target' field set to 'vehicle.lastUpdated'.

- Calcularemos cuando será la próxima vez que necesite pasar revisión el vehículo. Para ello, usaremos una regla ya definida que, en función de la categoría del vehículo (Small, Economy, Large o Luxury), establece si la siguiente revisión será dentro de 3 meses o 6 meses.



- La regla Open_CalculateNextServiceDate tiene 2 parámetros de entrada: vehicleType y dateUpdated.
- Al parámetro vehicleType le pasamos el valor de la categoría de nuestro vehículo (pv!vehicle.category).
- Al parámetro dateUpdated le pasamos una función que nos devuelve el momento actual.

En este paso, acabamos de añadir un script que se encarga de actualizar cuando el vehículo tendrá que volver a pasar la revisión.

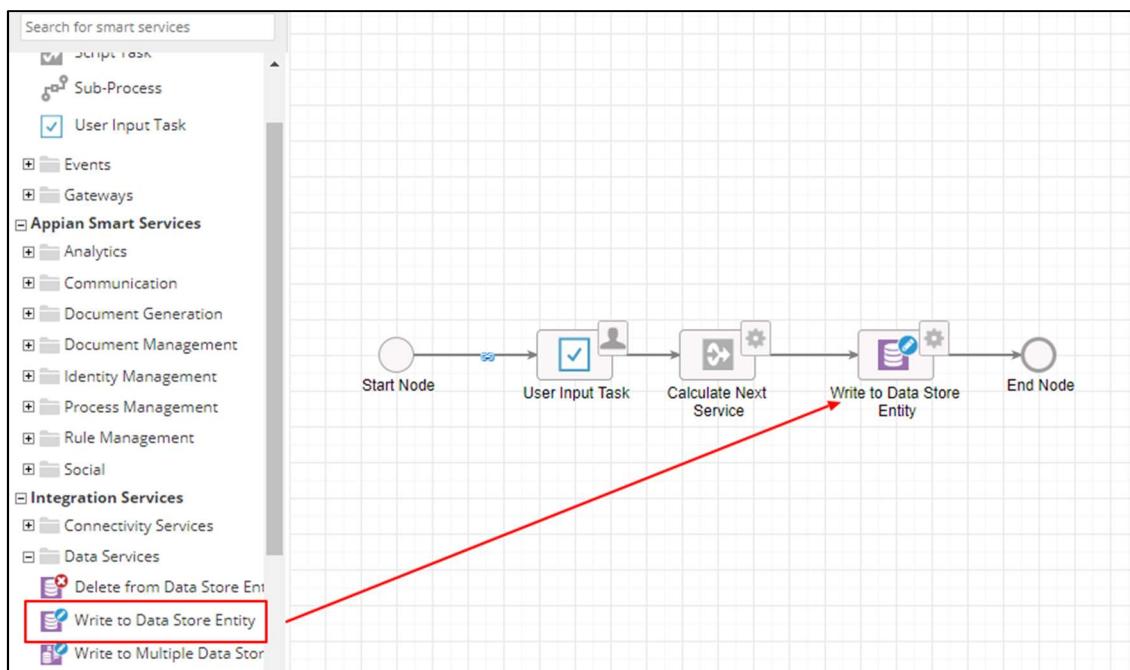
Añadir writeDataStore y configurar

Descripción:

Vamos a incluir en nuestro proceso un componente de integración con base de datos, que nos permitirá almacenar en nuestra tabla xxx_vehicle los vehículos.

Pasos a realizar:

- En el menú de la izquierda, localizar el componente Write to Data Store Entity (se encuentra en Integration Services - Data Services) y arrastrarlo al panel central, sobre la línea que conecta el nodo Calculate Next Service y el evento End. El resultado debe ser:



- Pulsando con el botón derecho sobre el componente añadido de Write to Data Store – Properties, se nos abre una ventana emergente, con las propiedades de dicho componente. Debemos completar las siguientes:
- Pestaña Data
 - Data Store Entity
 - **Value:** pulsando sobre la lupa, buscamos nuestro Data Store concreto (XXX_DS) y seleccionamos el entity (xxx_Vehicle).

General	Data	Forms	Scheduling	Assignment	Escalations	Exceptions	Other												
Inputs	Outputs																		
Node Inputs <small>Map the value(s) for the inputs of the node</small>																			
<input type="button" value="New Input"/> Data Store Entity * (Data Store Entity)				Field Properties <table border="1"> <tr> <td>Name</td> <td>Data Store Entity</td> </tr> <tr> <td>Type</td> <td>Data Store Entity</td> </tr> <tr> <td>Multiple</td> <td><input type="checkbox"/></td> </tr> <tr> <td>Value</td> <td>"Open_vehicle"</td> </tr> <tr> <td>Required</td> <td><input type="checkbox"/></td> </tr> <tr> <td>Save into</td> <td></td> </tr> </table>				Name	Data Store Entity	Type	Data Store Entity	Multiple	<input type="checkbox"/>	Value	"Open_vehicle"	Required	<input type="checkbox"/>	Save into	
Name	Data Store Entity																		
Type	Data Store Entity																		
Multiple	<input type="checkbox"/>																		
Value	"Open_vehicle"																		
Required	<input type="checkbox"/>																		
Save into																			

- New Node Input. Añadimos otro Node Input

Inputs Outputs

Node Inputs Map the value(s) for the inputs of the node

New Input **Delete Input**

Data Store Entity (Data Store Entity)
vehicle (Open_vehicle)

Field Properties	
Name	vehicle
Type	Open_vehicle
Multiple	<input type="checkbox"/>
Value	=pv!vehicle
Required	<input type="checkbox"/>
Save into	

Nota: El campo type, en nuestro caso será XXX_Vehicle.

- Pulsamos OK
- Nos vamos al menú superior – File – Save and Publish.

Una vez terminada la publicación, nos debe aparecer en la parte inferior del panel central un mensaje informativo indicándonos que el process model se ha publicado con éxito.

Cada vez que realicemos alguna modificación sobre un proceso, debemos volver a publicarlo para que sus cambios sean visibles en las próximas pruebas.

Appian Process Modeler

File Edit View Tools Lanes

New Process Model
Open...
Close
Close All
Save (Ctrl+S)
Save As... (Ctrl+Shift+S)
Save & Publish (Ctrl+Alt+S) 1
Versions...
Security
Start Process for Debugging (Ctrl+D)
Print Preview
Print... (Ctrl+P)
Import (Ctrl+I)
Properties...
Exit
Send Message
Start Event
Timer
Gateways
Appian Smart Services
Analytics

FAR_Review_Vehicle v2.0

Start Node → Review Vehicle → End Node

2

Process Model The process model has been published.

Copyright © 2019 Accenture
All rights reserved.

Accenture, its Signature, and
High Performance Delivered
are trademarks of Accenture.