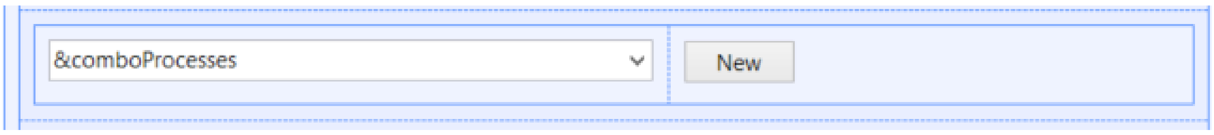
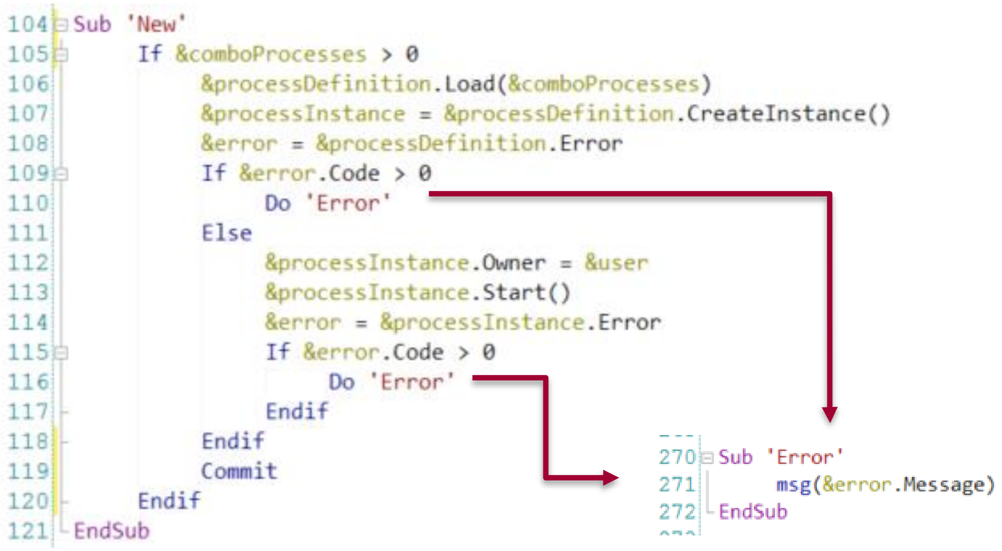
Custom Client Part 2

WORKFLOW INBOX – NEW ACTION

This function occurs when the user selects a process from the combo box and then clicks the new button. This will execute a subroutine called new.





```
Sub 'New'
    If &comboProcesses > 0
        &processDefinition.Load(&comboProcesses)
        &processInstance = &processDefinition.CreateInstance()
        &error = &processDefinition.Error
        If &error.Code > 0
            Do 'Error'
        Else
            &processInstance.Owner = &user
            &processInstance.Start()
            &error = &processInstance.Error
            If &error.Code > 0
                Do 'Error'
            Endif
        Endif
        Commit
    Endif
EndSub
```

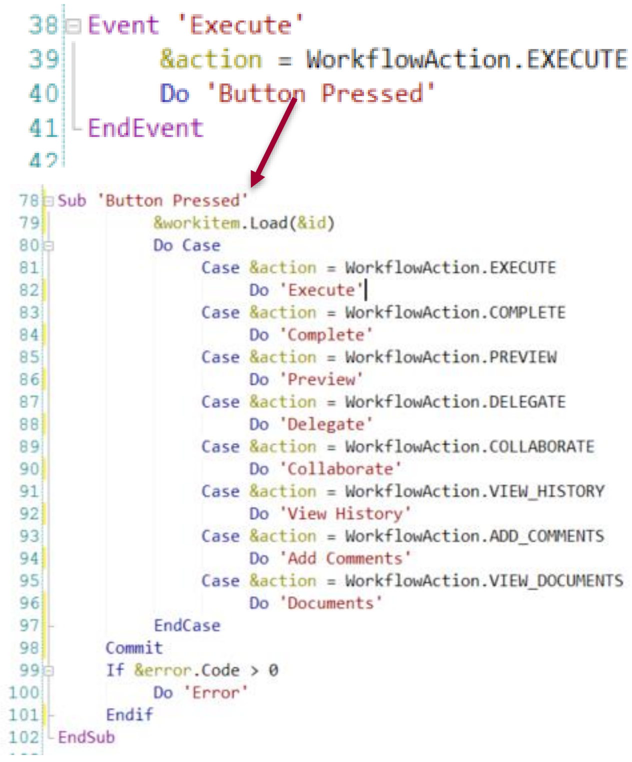
```
Sub 'Error'
    msg(&error.Message)
EndSub
```

This will execute a subroutine called new. In this subroutine we are going to have a variable of the process definition data type and will execute the load method of this process definition data type. These functions loads a process definition into this variable, which process definitions The one passed by, in parameter this case the one selected in the combo box. After we have selected the process definition data type we are going to use the method create instance of this process definition. This method will create a new process instance, and that will be assigned to the workflow process instance data type variable. It's very important to check if there's an error every time we execute a method to control the errors and take actions in case there are. If there is no error we are going to continue and assign the user logged in to the process instance owner property. After that we are going to start this process instance, we are going to start the execution of the process and then we will start creating all the work items needed. After that we are going to check if there is any error at the start of this process and if there is an error we are going to show a message to the user

At the end of this subroutine there is the commit statement. It's important to have the commit statement, because all the APIs support of commit workflow don't commit by themselves, so we have to manage the UTL of the transaction right in the commit statement.

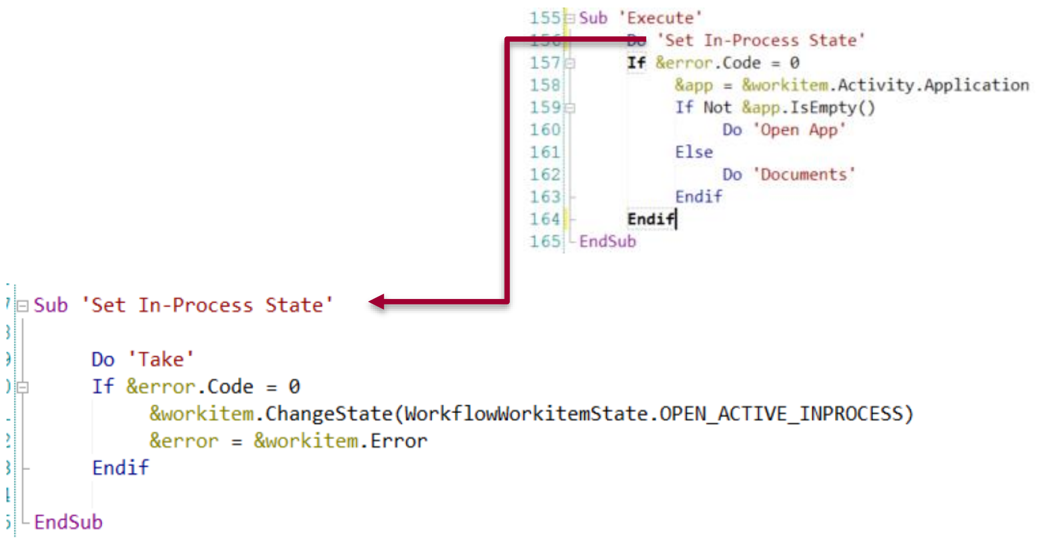
WORKFLOW INBOX – EXECUTE ACTION

Now we are going to see the execute action. This action takes place when the user selects a workitem in the grId and then press the execute bar. The first thing to do is execute the execute event, here you see the workflow action domain with the value execute, it will be assign this value to the variable action. Then it will execute the button pressed subroutine, in the subroutine, in the workitem variable, which is the type workflow workitem data type, we will use the method load in order to load the workitem selected in a variable, using the Id taken from the grId.



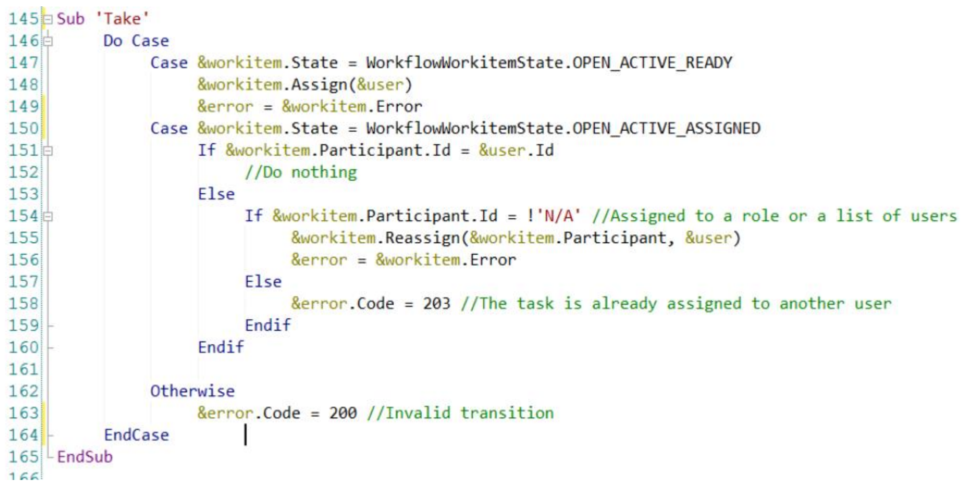
```
Event 'Execute'
      &action = WorkflowAction.EXECUTE
      Do 'Button Pressed'
EndEvent

Sub 'Button Pressed'
        &workitem.Load(&id)
        Do Case
            Case &action = WorkflowAction.EXECUTE
                Do 'Execute'
            Case &action = WorkflowAction.COMPLETE
                Do 'Complete'
            Case &action = WorkflowAction.PREVIEW
                Do 'Preview'
            Case &action = WorkflowAction.DELEGATE
                Do 'Delegate'
            Case &action = WorkflowAction.COLLABORATE
                Do 'Collaborate'
            Case &action = WorkflowAction.VIEW_HISTORY
                Do 'View History'
            Case &action = WorkflowAction.ADD_COMMENTS
                Do 'Add Comments'
            Case &action = WorkflowAction.VIEW_DOCUMENTS
                Do 'Documents'
        EndCase
    Commit
    If &error.Code > 0
        Do 'Error'
    Endif
EndSub
```

After the workitem has been loaded to the workitem variable it's going to execute the subroutine called execute. In this subroutine the first thing to do is to set in processes state of the work items, so it's going to execute the do subroutine. This subroutine is going to verify the state of the workitem, so it will verify the state with the workflow workitem state. If it´s open active ready, which means that it has been not assigned to any user, the first thing to do is to assign the workitem to the user executing the workflow inbox. So we use the assign method or the workitem variable and it passed the variable user which has been loaded at the start event. If there is no error it will continue. In case the workitem is open active assigned, it will validate that the user logged in is the user which that has been executing this workitem. But in case that the user executing is not the user that has previously executed the workitem, it will reassign the workitem to the user who is running the inbox, so here we will use the reassign method of the workitem and will reassign the participant to the user who is running the workflow inbox.



```
Sub 'Execute'
    Do 'Set In-Process State'
    If &error.Code = 0
        &app = &workitem.Activity.Application
        If Not &app.IsEmpty()
            Do 'Open App'
        Else
            Do 'Documents'
        Endif
    Endif
EndSub
```

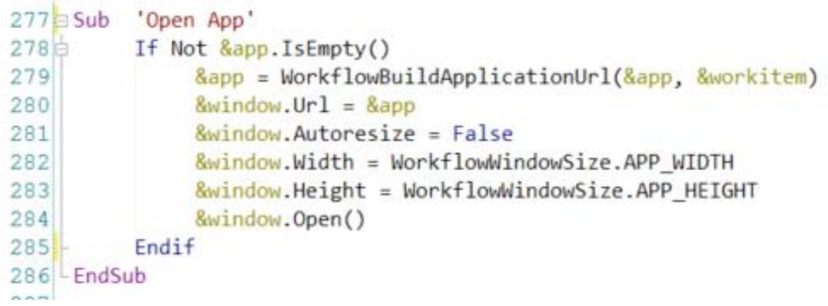
```
Sub 'Set In-Process State'

    Do 'Take'
    If &error.Code = 0
        &workitem.ChangeState(WorkflowWorkitemState.OPEN_ACTIVE_INPROCESS)
        &error = &workitem.Error
    Endif

EndSub
```



```
Sub 'Take'
    Do Case
        Case &workitem.State = WorkflowWorkitemState.OPEN_ACTIVE_READY
            &workitem.Assign(&user)
            &error = &workitem.Error
        Case &workitem.State = WorkflowWorkitemState.OPEN_ACTIVE_ASSIGNED
            If &workitem.Participant.Id = &user.Id
                //Do nothing
            Else
                If &workitem.Participant.Id = !'N/A' //Assigned to a role or a list of users
                    &workitem.Reassign(&workitem.Participant, &user)
                    &error = &workitem.Error
                Else
                    &error.Code = 203 //The task is already assigned to another user
                Endif
            Endif

        Otherwise
            &error.Code = 200 //Invalid transition
    EndCase
EndSub
```

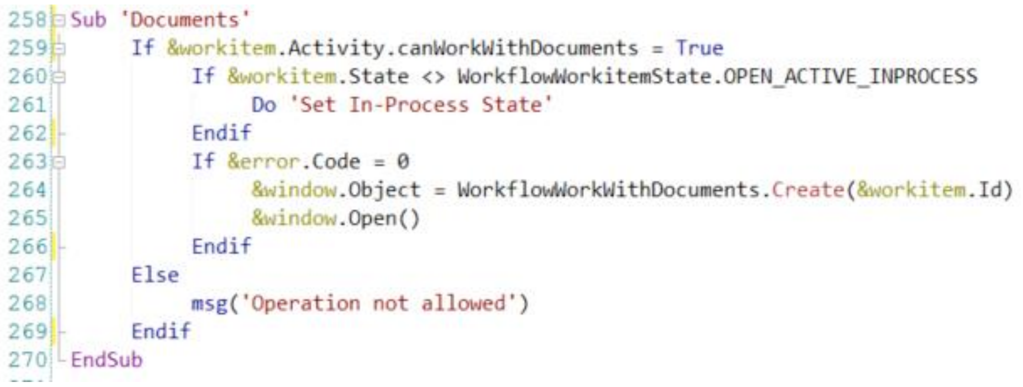
After the take method, it will change the workitems state using the method ChangeState and it will change to OPEN_ACTIVE_INPROCESES, which is that the workitem is running.

After the workitem has the correct state it will open the application defined at the process definition diagram. So we are going to use the work item variable again, the activity and the application property, to obtain the application needed to run. If the application is not empty then it will execute the open up subroutine. In this subroutine it has the window variable data type to create a pop up and open the application.



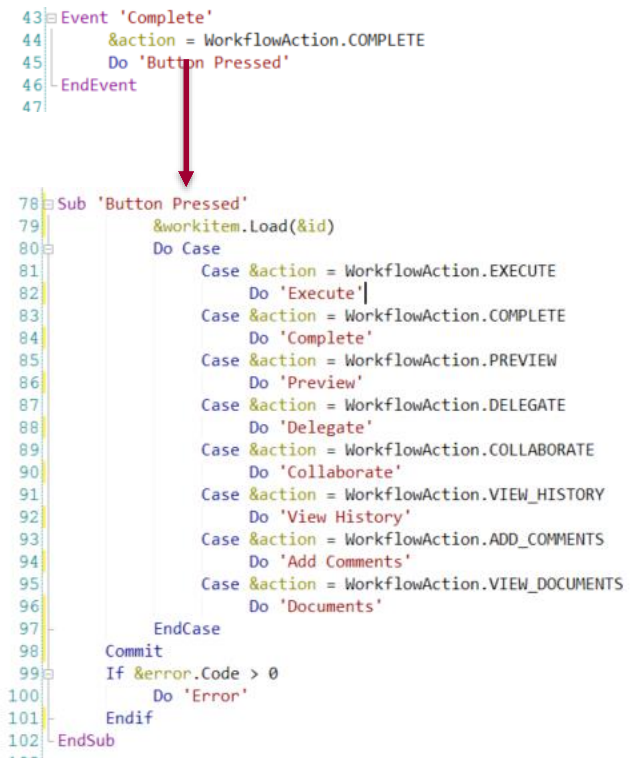
```
Sub  'Open App'
     If Not &app.IsEmpty()
          &app = WorkflowBuildApplicationUrl(&app, &workitem)
          &window.Url = &app
          &window.Autoresize = False
          &window.Width = WorkflowWindowSize.APP_WIDTH
          &window.Height = WorkflowWindowSize.APP_HEIGHT
          &window.Open()
     Endif
EndSub
```

In case the application variable is empty it will execute the subroutine documents. In this case when no application is associated to a process activity, the documents is going to appear in case that it has been defined to do so. So it will validate that the workitem activity has the property can work with work items true, if it hasn't true the It will validate if the state of the work item is correct. In case it's correct it will execute the work with document web panel to show the work with documents. If there is no can work with documents set up on the activity, then it will do nothing because it has no application and cannot work with documents



```
Sub 'Documents'
     If &workitem.Activity.canWorkWithDocuments = True
          If &workitem.State <> WorkflowWorkitemState.OPEN_ACTIVE_INPROCESS
               Do 'Set In-Process State'
          Endif
          If &error.Code = 0
               &window.Object = WorkflowWorkWithDocuments.Create(&workitem.Id)
               &window.Open()
          Endif
     Else
          msg('Operation not allowed')
     Endif
EndSub
```

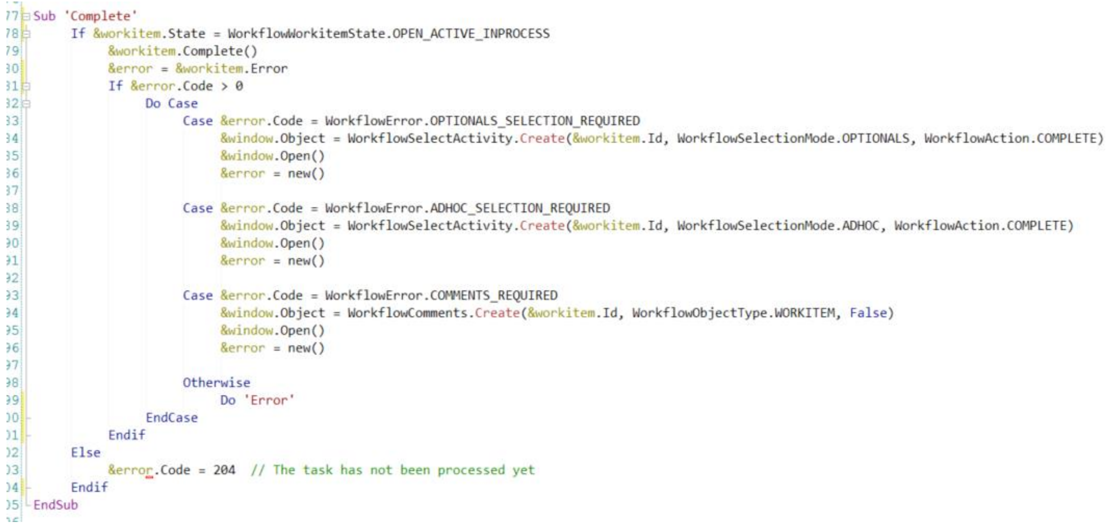
When a user selects a workitem from the grid and clicks the complete button, it will execute the event complete. Here we will use the workflow action domain with the value complete and assigned to the variable action. After that it will execute the subroutine Button Pressed in the subroutine, as well as in the execute action, which loads the selected workitem Id into the workitem variable, which is of the workflow workitem datatype.



```
Event 'Complete'
	&action = WorkflowAction.COMPLETE
	Do 'Button Pressed'
EndEvent
```

```
Sub 'Button Pressed'
		&workitem.Load(&id)
		Do Case
			Case &action = WorkflowAction.EXECUTE
				Do 'Execute'
			Case &action = WorkflowAction.COMPLETE
				Do 'Complete'
			Case &action = WorkflowAction.PREVIEW
				Do 'Preview'
			Case &action = WorkflowAction.DELEGATE
				Do 'Delegate'
			Case &action = WorkflowAction.COLLABORATE
				Do 'Collaborate'
			Case &action = WorkflowAction.VIEW_HISTORY
				Do 'View History'
			Case &action = WorkflowAction.ADD_COMMENTS
				Do 'Add Comments'
			Case &action = WorkflowAction.VIEW_DOCUMENTS
				Do 'Documents'
		EndCase
	Commit
	If &error.Code > 0
		Do 'Error'
	Endif
EndSub
```

After that it will execute the subroutine Complete and in this subroutine it will check that the workitem has the correct state before it continues. All the workitems to be completed have to have the workitem state OPEN_ACTIVE_INPROCESES that means that it has been executed previously.

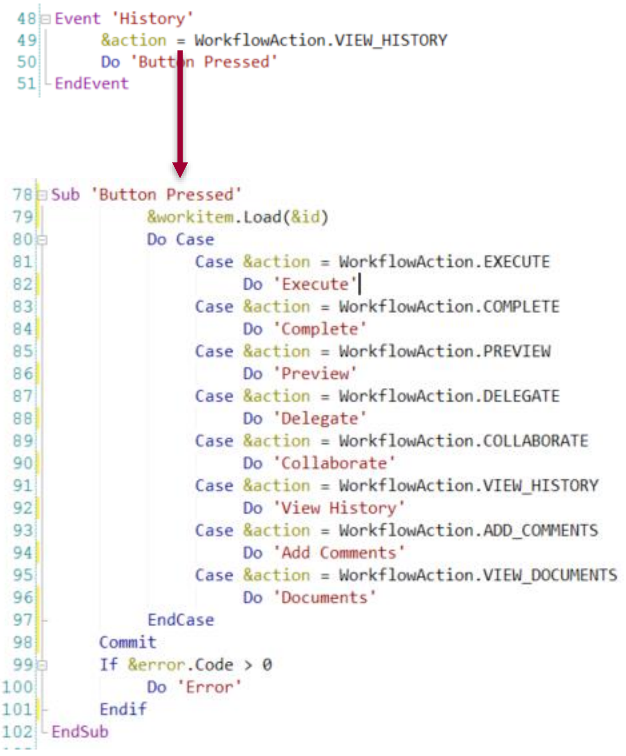


```
Sub 'Complete'
	If &workitem.State = WorkflowWorkitemState.OPEN_ACTIVE_INPROCESS
		&workitem.Complete()
		&error = &workitem.Error
		If &error.Code > 0
			Do Case
				Case &error.Code = WorkflowError.OPTIONALS_SELECTION_REQUIRED
					&window.Object = WorkflowSelectActivity.Create(&workitem.Id, WorkflowSelectionMode.OPTIONALS, WorkflowAction.COMPLETE)
					&window.Open()
					&error = new()

				Case &error.Code = WorkflowError.ADHOC_SELECTION_REQUIRED
					&window.Object = WorkflowSelectActivity.Create(&workitem.Id, WorkflowSelectionMode.ADHOC, WorkflowAction.COMPLETE)
					&window.Open()
					&error = new()

				Case &error.Code = WorkflowError.COMMENTS_REQUIRED
					&window.Object = WorkflowComments.Create(&workitem.Id, WorkflowObjectType.WORKITEM, False)
					&window.Open()
					&error = new()

				Otherwise
					Do 'Error'
			EndCase
		Endif
	Else
		&error.Code = 204  // The task has not been processed yet
	Endif
EndSub
```

If it has the correct state it will complete the workitem that means that this workitem is going to end and the next workitems in the process will be created. If there is an error, for example because optional path has to be selected from the user, it will be prompted a panel workflow select activity to select which path it has to be taken. In case the error is because it's Ad-hoc process and it has to be selected also the next activity to execute, will be prompted to say the workflow selected activity, in order the user to select which activity will continue. In case the error is because some comments are required in this activity, then it's going to be prompted the workflow comment web panel, in order to the user to entry the comments needed to complete this task. If there is an error, the error will be shown to the user, in case there is no error, the task has finished.

WORKFLOW INBOX – HISTORY ACTION

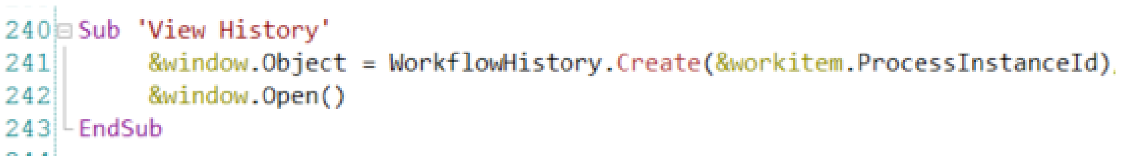
In the workflow inbox of the Custom Client with the preselected work item from the grid and pressed the History Button, it will present the history panel. This panel presents a list of folder work items that had been executed from the process instance. To do so, it executes the following code, when the Button Jistory is pressed it will execute the event History event, and in this event it will use, as same as the other actions of this panel, assign the workflow action domain, but with the value VIEW_HISTORY to the action variable, and then execute the button pressed subroutine. In this subroutine it will load to the workitem variable, which is of the workitem data type, the Id that has been selected in the grid. This is the Id of the workitem, and then we have the workitem assigned in this variable.



```
Event 'History'
	&action = WorkflowAction.VIEW_HISTORY
	Do 'Button Pressed'
EndEvent
```

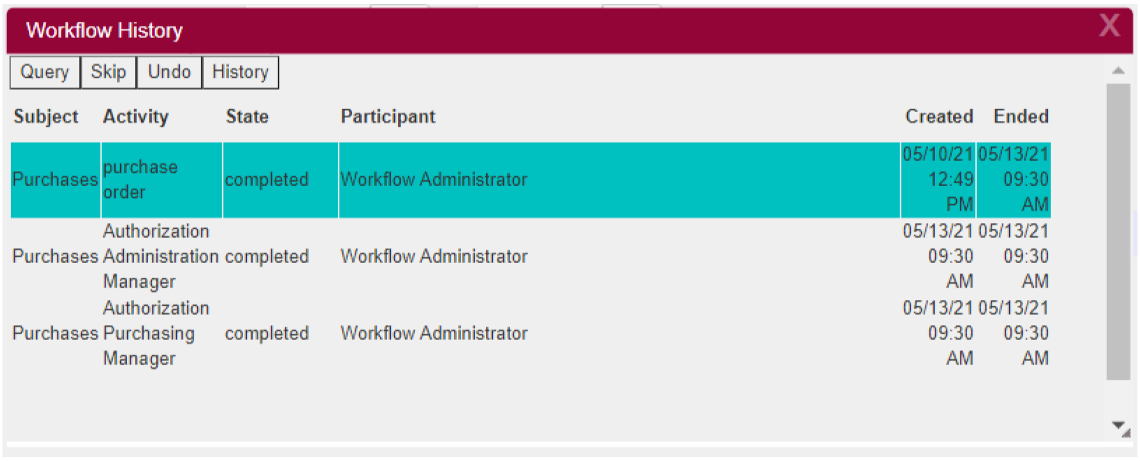
```
Sub 'Button Pressed'
		&workitem.Load(&id)
		Do Case
			Case &action = WorkflowAction.EXECUTE
				Do 'Execute'
			Case &action = WorkflowAction.COMPLETE
				Do 'Complete'
			Case &action = WorkflowAction.PREVIEW
				Do 'Preview'
			Case &action = WorkflowAction.DELEGATE
				Do 'Delegate'
			Case &action = WorkflowAction.COLLABORATE
				Do 'Collaborate'
			Case &action = WorkflowAction.VIEW_HISTORY
				Do 'View History'
			Case &action = WorkflowAction.ADD_COMMENTS
				Do 'Add Comments'
			Case &action = WorkflowAction.VIEW_DOCUMENTS
				Do 'Documents'
		EndCase
	Commit
	If &error.Code > 0
		Do 'Error'
	Endif
EndSub
```

After that we will execute the event subroutine View History. In this subroutine it will use the window variable to open a web component, which is called Workflow History. This panel have received by parameter the process instance Id, and here we are going to pass this process instance Id using the workitem variable that has been assigned in the sub button pressed. This variable has a property which is called process instance Id that returns the Id of the process instance for this workitem and then will open the page.



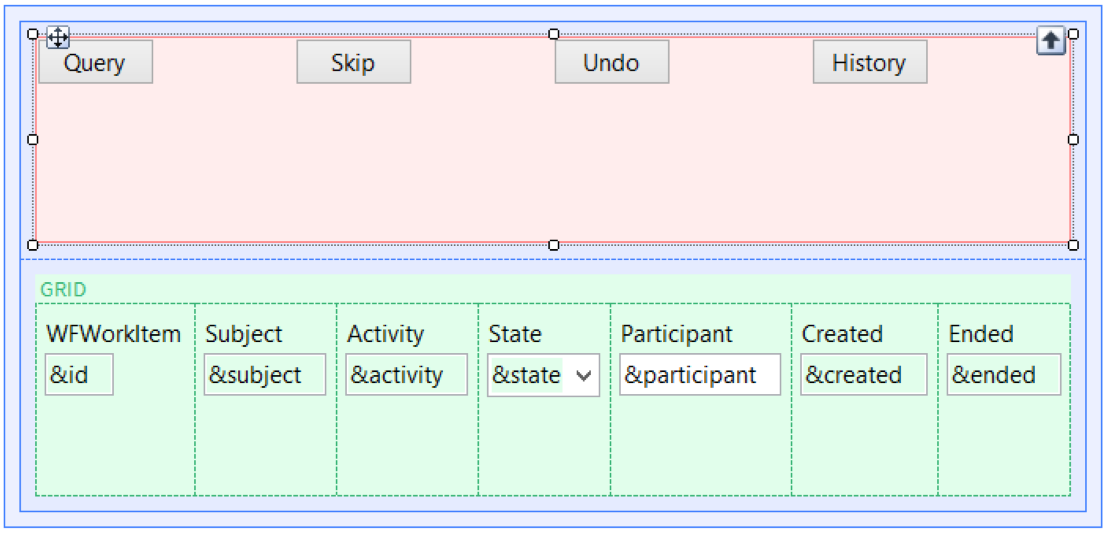
```
Sub 'View History'
	&window.Object = WorkflowHistory.Create(&workitem.ProcessInstanceId),
	&window.Open()
EndSub
```

If we see this panel execution, we are going to see that there is a list of workitems, all the work item that has been executed of this process instance with the subject, the activity, the state, the participant, when it was created, when it was ended; and a list of actions that can be executed for this workitems.



Subject	Activity	State	Participant	Created	Ended
Purchases	purchase order	completed	Workflow Administrator	05/10/21 12:49 PM	05/13/21 09:30 AM
Purchases	Authorization Administration Manager	completed	Workflow Administrator	05/13/21 09:30 AM	05/13/21 09:30 AM
Purchases	Authorization Purchasing Manager	completed	Workflow Administrator	05/13/21 09:30 AM	05/13/21 09:30 AM

If we see the Web panel layout it has a grid with all the elements we have seen, and a table with all the actions.



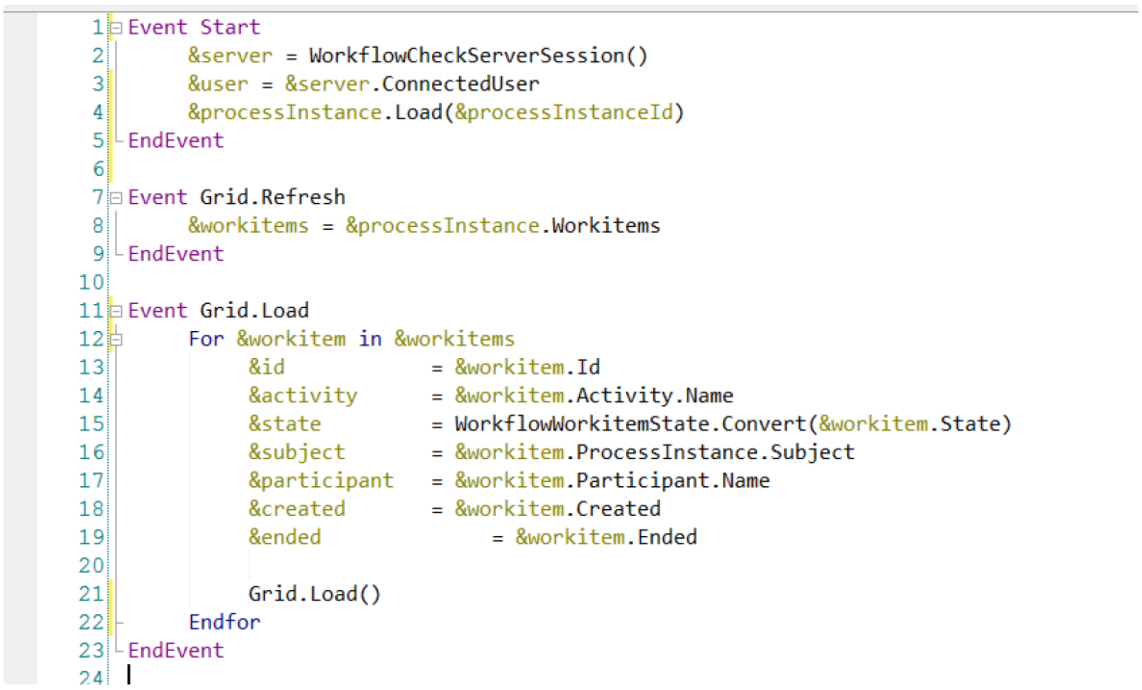
WFWorkItem	Subject	Activity	State	Participant	Created	Ended
&id	&subject	&activity	&state	&participant	&created	&ended

WORFKOW HISTORY – CODE

If we look at the code we are going to see at the rules that it receives a parameter, &ProcessInstanceId.

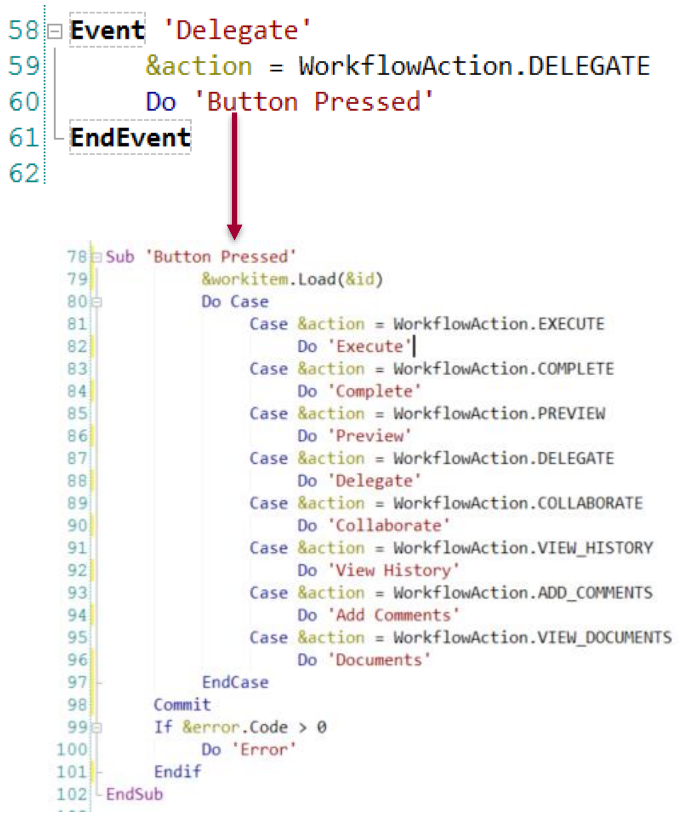
In the Event Start it will do the same as all the panels in the Custom Client, it will check that there is a valId session and then it will assign the user logged in, that we obtained by the connected user property of the workflow server variable, to the variable user. After that, it will load into the process instance variable, which is of workflow process instance data type, the process instance Id that will have received by parameter. This will load the Process Instance into this variable. And now that we have the process instance at the Refresh event, it will obtain all the workitems that had been created for this Process Instance, using the workitems property.

This property returns a list of workitems, these workitems are the workflow workitems data type and it's assigned to the workitems variable. This list it's going to be iterated at the load event and for each workitem of this list, is going to assign its values to the grid. For example it will assign the Id of the workitem, the name of the activity of the workitem, the state, the subject of the process instance, the name of the participant who executed this task, the date it was created and the date it was ended, after it loads a role to the grid



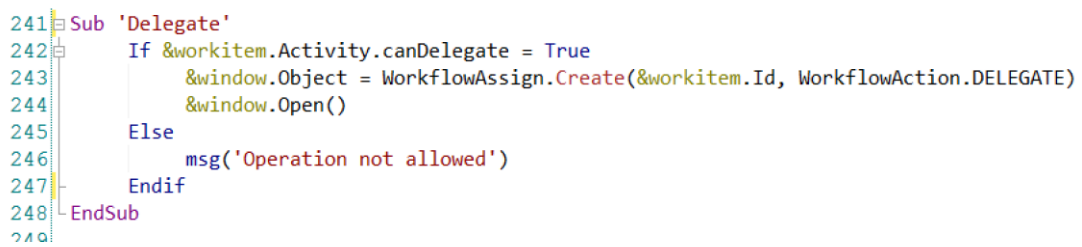
```
Event Start
	&server = WorkflowCheckServerSession()
	&user = &server.ConnectedUser
	&processInstance.Load(&processInstanceId)
EndEvent

Event Grid.Refresh
	&workitems = &processInstance.Workitems
EndEvent

Event Grid.Load
	For &workitem in &workitems
		&id		= &workitem.Id
		&activity	= &workitem.Activity.Name
		&state		= WorkflowWorkitemState.Convert(&workitem.State)
		&subject	= &workitem.ProcessInstance.Subject
		&participant	= &workitem.Participant.Name
		&created	= &workitem.Created
		&ended			= &workitem.Ended

		Grid.Load()
	Endfor
EndEvent
```

When we execute this action we are going to delegate a workitem that we will have in our inbox to another person. To do so, in the event delegate, as well as other actions, I will assign the workflow action, but with the value delegate into the action variable, and then execute the subroutine button pressed. In this subroutine it will load the Id of the work items selected in the grid to the workitem variable, and execute the delegate's subroutine. In this subroutine we are going to check that the activity of the workitem has enabled to delegate functions. You can enable the delegation selecting in the diagram the task activity and in the properties putting true into the allow delegation property, this will enable the delegation action. If the task can delegate then it will present the panel workflow assign to the user. This panel will present all the users that can receive this task.



```
Event 'Delegate'
	&action = WorkflowAction.DELEGATE
	Do 'Button Pressed'
EndEvent
```

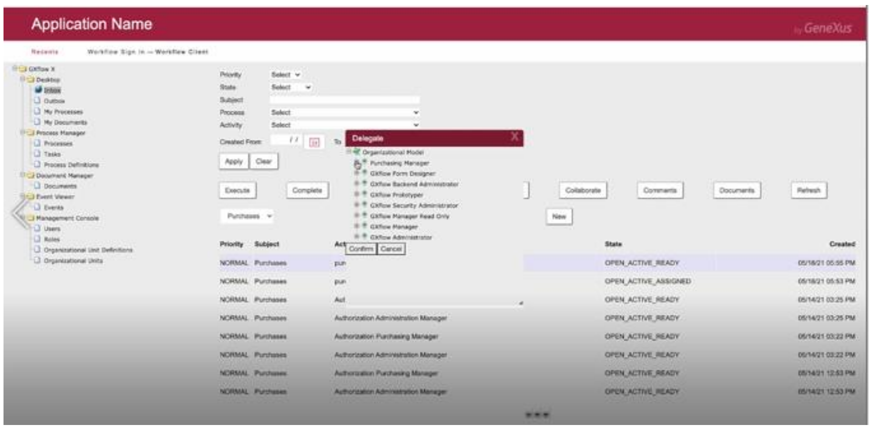
```
Sub 'Button Pressed'
		&workitem.Load(&id)
		Do Case
			Case &action = WorkflowAction.EXECUTE
				Do 'Execute'
			Case &action = WorkflowAction.COMPLETE
				Do 'Complete'
			Case &action = WorkflowAction.PREVIEW
				Do 'Preview'
			Case &action = WorkflowAction.DELEGATE
				Do 'Delegate'
			Case &action = WorkflowAction.COLLABORATE
				Do 'Collaborate'
			Case &action = WorkflowAction.VIEW_HISTORY
				Do 'View History'
			Case &action = WorkflowAction.ADD_COMMENTS
				Do 'Add Comments'
			Case &action = WorkflowAction.VIEW_DOCUMENTS
				Do 'Documents'
		EndCase
	Commit
	If &error.Code > 0
		Do 'Error'
	Endif
EndSub
```



```
Sub 'Delegate'
	If &workitem.Activity.canDelegate = True
		&window.Object = WorkflowAssign.Create(&workitem.Id, WorkflowAction.DELEGATE)
		&window.Open()
	Else
		msg('Operation not allowed')
	Endif
EndSub
```

If we see the delegate function in execution, we can see that if we select our workitem from our workflow inbox grid and then press the delegate button, there will be prompted a web page with all the organization and Model and all the roles they have on my knowledge page, it says select open a role, I can see the users. If I select a user I can confirm and this task is going to be delegated to that user.

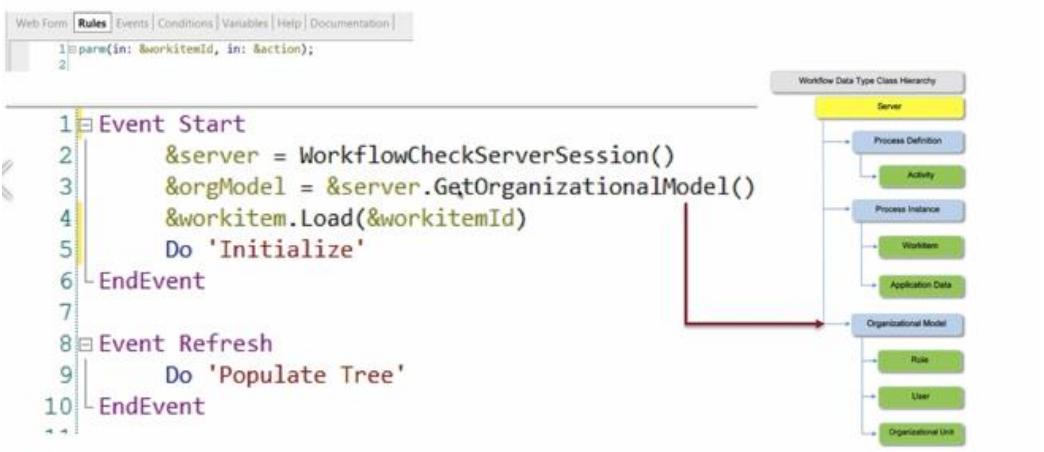
This Web panel is created with a window object and receives two parameters, the workitem Id, that has been loaded in the button pressed subroutine, and the action, in this case workflow action delegate.



Priority	Subject	Act	State	Created
NORMAL	Purchases	pun	OPEN_ACTIVE_READY	05/18/21 05:55 PM
NORMAL	Purchases	pun	OPEN_ACTIVE_ASSIGNED	05/18/21 05:53 PM
NORMAL	Purchases	Aut	OPEN_ACTIVE_READY	05/14/21 03:25 PM
NORMAL	Purchases	Authorization Administration Manager	OPEN_ACTIVE_READY	05/14/21 03:25 PM
NORMAL	Purchases	Authorization Purchasing Manager	OPEN_ACTIVE_READY	05/14/21 03:22 PM
NORMAL	Purchases	Authorization Administration Manager	OPEN_ACTIVE_READY	05/14/21 03:22 PM
NORMAL	Purchases	Authorization Purchasing Manager	OPEN_ACTIVE_READY	05/14/21 12:53 PM
NORMAL	Purchases	Authorization Administration Manager	OPEN_ACTIVE_READY	05/14/21 12:53 PM

In the layout we are going to see that this is different from the others panels, it has three controls and two buttons, confirm and cancel.

If we see the code, we are going to see that receives two parameters, workitem Id and action. The Event Start it will check the workflow session with the WorkflowCheckServerSession procedure as same as other panels. Then it will get from the WorkflowServer that has been loaded the Organizational Model, which is part of the server. With the method Organizational Model this will return our workflow Organizational Model data type, which in this case is going to be OrgModel variable, then it will load the workitems receive parameter to the workitem variable

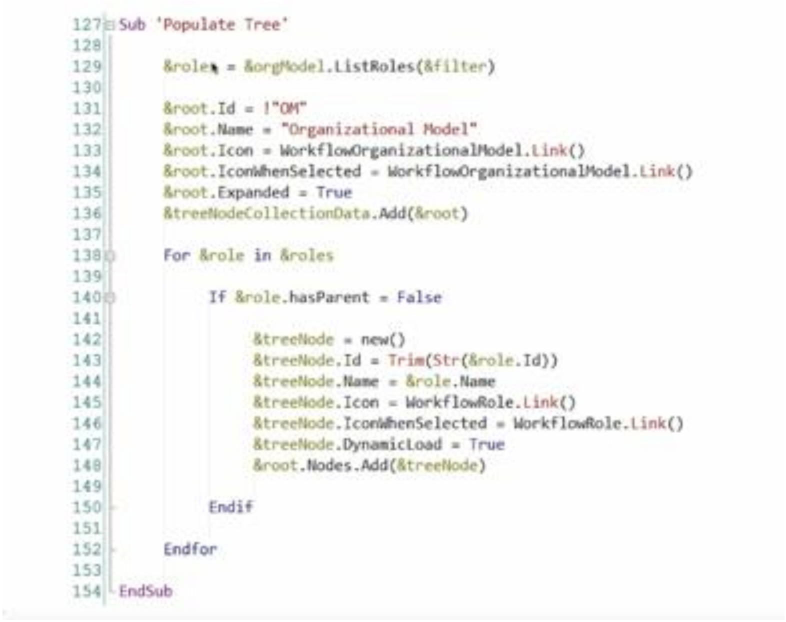


```
parm(in: &workitemId, in: &action);
```

```
Event Start
        &server = WorkflowCheckServerSession()
        &orgModel = &server.GetOrganizationalModel()
        &workitem.Load(&workitemId)
        Do 'Initialize'
EndEvent

Event Refresh
        Do 'Populate Tree'
EndEvent
```

.

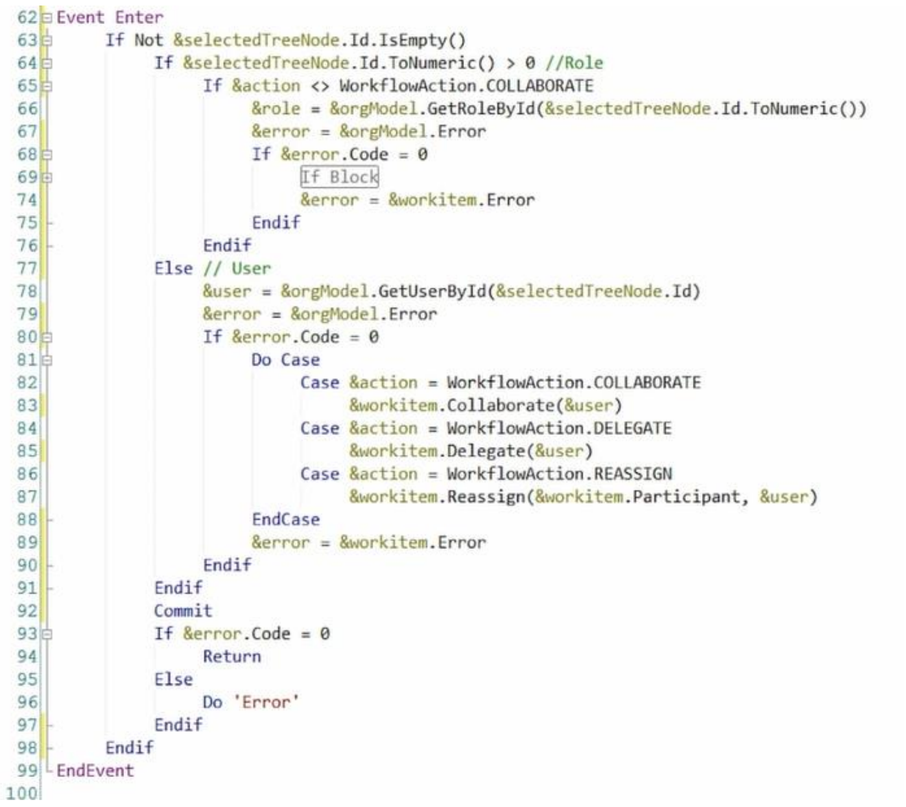
Then it will refresh and will Populate Tree control and this is going to do in this subroutine. Here list all the roles using a transitional variable method list roles, this receives a parameter, filter in this case its empty, and it will receive here in the roles all the roles of the Organizational Model. It will load the route of the tree and then for each role, in the role list, it will load all the roles into the tree.



```
Sub 'Populate Tree'

    &roles = &orgModel.ListRoles(&filter)

    &root.Id = !"OM"
    &root.Name = "Organizational Model"
    &root.Icon = WorkflowOrganizationalModel.Link()
    &root.IconWhenSelected = WorkflowOrganizationalModel.Link()
    &root.Expanded = True
    &treeNodeCollectionData.Add(&root)

    For &role in &roles

        If &role.hasParent = False

            &treeNode = new()
            &treeNode.Id = Trim(Str(&role.Id))
            &treeNode.Name = &role.Name
            &treeNode.Icon = WorkflowRole.Link()
            &treeNode.IconWhenSelected = WorkflowRole.Link()
            &treeNode.DynamicLoad = True
            &root.Nodes.Add(&treeNode)

        Endif

    Endfor

EndSub
```

WORKFLOW ASSIGN – WEB PANEL CODE

When it's assigning an element because we select a user or role and then press confirm it will validate that the Tree Node is not empty and the action is not collaborate, it will get the role selected in the tree with the ID selected and the variable, and then will get this role, in the Organizational Model, get role by ID, that will receive the ID of the role that has been loaded and this variable. Then we are going to get the workflow role here in the variable role, if it's no error it will assign. In case we are assigning to a user, we will get the user by the Organizational Model using get user by Id, and then if there is no error it will validate the action. In the case the action is delegate, we will use the workitem variable action delegate, this method will delegate this workitem to this user. The user selected in the tree. And then that delegates the workitem to this user, if there is no error we are going to commit to save all the changes and then return to the webpage.



```
Event Enter
    If Not &selectedTreeNode.Id.IsEmpty()
        If &selectedTreeNode.Id.ToNumeric() > 0 //Role
            If &action <> WorkflowAction.COLLABORATE
                &role = &orgModel.GetRoleById(&selectedTreeNode.Id.ToNumeric())
                &error = &orgModel.Error
                If &error.Code = 0
                    If Block
                    &error = &workitem.Error
                Endif
            Endif
        Else // User
            &user = &orgModel.GetUserById(&selectedTreeNode.Id)
            &error = &orgModel.Error
            If &error.Code = 0
                Do Case
                    Case &action = WorkflowAction.COLLABORATE
                        &workitem.Collaborate(&user)
                    Case &action = WorkflowAction.DELEGATE
                        &workitem.Delegate(&user)
                    Case &action = WorkflowAction.REASSIGN
                        &workitem.Reassign(&workitem.Participant, &user)
                EndCase
                &error = &workitem.Error
            Endif
        Endif
        Commit
        If &error.Code = 0
            Return
        Else
            Do 'Error'
        Endif
    Endif
EndEvent
```

In this video we have seen all the Custom Client, how can we can import a Custom Client to a Knowledge Base, why we use a Custom Client and the panel workflow inbox and all its actions.