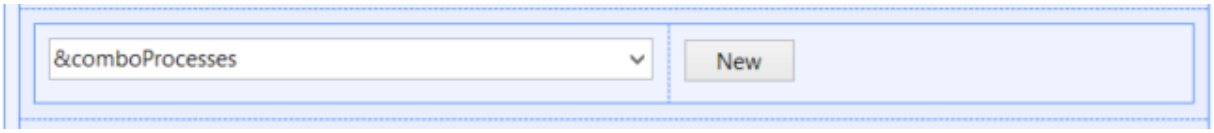
Custom Client Part 2

En este video vamos a ver la parte 2 del Custom Client. En esta parte veremos la bandeja de entrada y algunas funciones como delegar un workitem, completar un workitem y el History.

WORKFLOW INBOX – NEW ACTION

Esta función ocurre cuando el usuario selecciona un proceso del combo box y luego hace clic en el botón New. Esto ejecutará una subrutina llamada “New”.




```
104 Sub 'New'
105     If &comboProcesses > 0
106         &processDefinition.Load(&comboProcesses)
107         &processInstance = &processDefinition.CreateInstance()
108         &error = &processDefinition.Error
109         If &error.Code > 0
110             Do 'Error'
111         Else
112             &processInstance.Owner = &user
113             &processInstance.Start()
114             &error = &processInstance.Error
115             If &error.Code > 0
116                 Do 'Error'
117             Endif
118         Endif
119         Commit
120     Endif
121 EndSub

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

En esta subrutina vamos a tener una variable del tipo de dato Process Definition y ejecutaremos el método Load. Esta función carga la definición de proceso en una variable, cuyas definiciones de proceso la recibe del parámetro seleccionado en el combo box. Después de haber seleccionado el tipo de datos de definición de proceso, usaremos el método CreateInstance de la Definición del Proceso. Este método creará una nueva instancia de proceso y se asignará a la variable de tipo de datos ProcessInstance del workitem. Es muy importante verificar si hay un error cada vez que ejecutamos un método para controlar los errores y tomar acciones en caso de que lo haya. Si no hay ningún error, continuaremos y asignaremos el usuario que inició sesión en la propiedad Owner de la instancia del proceso. Después de eso, vamos a iniciar la instancia del proceso y luego comenzaremos a crear todos los workitems necesarios. Posteriormente vamos a comprobar si hay algún error al inicio de este proceso y si existe vamos a mostrar un mensaje al usuario.

Al final de esta subrutina está un commit. Es importante tener el commit, porque todas las API que admiten el Workflow no hacen commit por sí mismas, por lo que tenemos que administrar el UTL de la transacción directamente en el commit

WORKFLOW INBOX – EXECUTE ACTION

Ahora vamos a ver la acción Execute. Esta acción tiene lugar cuando el usuario selecciona un workitem del grid y luego presiona el botón Ejecutar. En el evento del mismo nombre se asignara el valor “Execute” del dominio WorkflowAction a la variable &action del tipo de dato WorkflowAction . Luego ejecutará la subrutina Button Pressed, donde, la variable workitem, que es el tipo de dato workflow workitem, usara el método Load para poder cargar el workitem seleccionado en la variable Id, tomado del grid.

```
38 Event 'Execute'
39       &action = WorkflowAction.EXECUTE
40       Do 'Button Pressed'
41 EndEvent
42
```

```
78 Sub 'Button Pressed'
79       &workitem.Load(&id)
80       Do Case
81            Case &action = WorkflowAction.EXECUTE
82                 Do 'Execute'
83            Case &action = WorkflowAction.COMPLETE
84                 Do 'Complete'
85            Case &action = WorkflowAction.PREVIEW
86                 Do 'Preview'
87            Case &action = WorkflowAction.DELEGATE
88                 Do 'Delegate'
89            Case &action = WorkflowAction.COLLABORATE
90                 Do 'Collaborate'
91            Case &action = WorkflowAction.VIEW_HISTORY
92                 Do 'View History'
93            Case &action = WorkflowAction.ADD_COMMENTS
94                 Do 'Add Comments'
95            Case &action = WorkflowAction.VIEW_DOCUMENTS
96                 Do 'Documents'
97       EndCase
98    Commit
99    If &error.Code > 0
100       Do 'Error'
101   Endif
102 EndSub
```

Después de que el workitem se haya cargado en la variable, se ejecutará la subrutina llamada Execute. En la subrutina Execute, lo primero que debe hacer es establecer el In- proceses state del workitem, por lo que ejecutará la subrutina “Set In-process state”, la cual invoca a la subrutina Take. En esta subrutina se verificará el estado del workitem. Si el workitem está OPEN_ACTIVE_READY, significa que no ha sido asignado a ningún usuario, lo primero que debe hacerse es asignar el workitem al usuario que ejecuto la bandeja de entrada. Entonces usamos el método “Assign” de la variable workitem, pasando la variable &user del tipo Workflowuser que se cargó en el evento Start. Si no hay ningún error, continuará.

En caso de que el workitem esté OPEN_ACTIVE_ASSIGNED, validará que el usuario que inició sesión es el usuario que ha estado ejecutando este workitem, pero en caso de que el usuario que ejecuta no sea el usuario que ha ejecutado previamente el workitem, reasignará el workitem al usuario que está ejecutando la bandeja de entrada, por lo que aquí usaremos el método Reasing del workitem y reasignaremos al participante al usuario que está ejecutando la bandeja de entrada como se muestra en el código usando la variable workitem con el método participante y la variable user del tipo WorkflowUser..

```
155 Sub 'Execute'
156     Do 'Set In-Process State'
157     If &error.Code = 0
158         &app = &workitem.Activity.Application
159         If Not &app.IsEmpty()
160             Do 'Open App'
161         Else
162             Do 'Documents'
163         Endif
164     Endif
165 EndSub
```

```
Sub 'Set In-Process State'

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

EndSub
```

```
145 Sub 'Take'
146     Do Case
147         Case &workitem.State = WorkflowWorkitemState.OPEN_ACTIVE_READY
148             &workitem.Assign(&user)
149             &error = &workitem.Error
150         Case &workitem.State = WorkflowWorkitemState.OPEN_ACTIVE_ASSIGNED
151             If &workitem.Participant.Id = &user.Id
152                 //Do nothing
153             Else
154                 If &workitem.Participant.Id = !'N/A' //Assigned to a role or a list of users
155                     &workitem.Reassign(&workitem.Participant, &user)
156                     &error = &workitem.Error
157                 Else
158                     &error.Code = 203 //The task is already assigned to another user
159                 Endif
160             Endif
161
162         Otherwise
163             &error.Code = 200 //Invalid transition
164     EndCase
165 EndSub
```

Después del método “Take”, cambiará el estado del workitem usando el método “ChangeState” y cambiará el estado a OPEN_ACTIVE_INPROCESES.

Una vez que el workitem tenga el estado correcto, se abrirá la aplicación definida en el diagrama de la definición del proceso. En dicho diagrama el usuario indica en la propiedad application la actividad que se ejecutará en el workitem. Entonces vamos a usar nuevamente la variable workitem, activity y la propiedad application, para obtener la aplicación que se va ejecutar.

Si la aplicación no está vacía, ejecutará la subrutina Open App. Esta subrutina tiene una variable del tipo de dato Window para crear una ventana emergente y abrir la aplicación.

```
277 Sub  'Open App'
278     If Not &app.IsEmpty()
279         &app = WorkflowBuildApplicationUrl(&app, &workitem)
280         &window.Url = &app
281         &window.Autoresize = False
282         &window.Width = WorkflowWindowSize.APP_WIDTH
283         &window.Height = WorkflowWindowSize.APP_HEIGHT
284         &window.Open()
285     Endif
286 EndSub
```

En caso de que la variable application esté vacía, ejecutará la subrutina “Documents”. En este caso cuando no haya una aplicación asociada a una actividad de proceso, los documentos van a aparecer en un Trabajar con documentos, en caso de que se haya definido para ello. Por lo tanto, validará que la actividad del workitem tiene la propiedad canWorkWithtDocuments en True, si no es cierto, validará si el estado del workitem es correcto. En caso de que sea correcto ejecutará el Web panel Work With Document para mostrar el trabajo con documentos. Si no puede trabajar con los documentos configurados en la actividad, entonces no hará nada porque no tiene una aplicación y no puede trabajar con los documentos.

```
258 Sub 'Documents'
259     If &workitem.Activity.canWorkWithDocuments = True
260         If &workitem.State <> WorkflowWorkitemState.OPEN_ACTIVE_INPROCESS
261             Do 'Set In-Process State'
262         Endif
263         If &error.Code = 0
264             &window.Object = WorkflowWorkWithDocuments.Create(&workitem.Id)
265             &window.Open()
266         Endif
267     Else
268         msg('Operation not allowed')
269     Endif
270 EndSub
```

A continuación veremos la acción complete.

WORKFLOW INBOX COMPLETE ACTION

Cuando un usuario toma un workitem del grid y hace clic en el botón Complete, ejecutará el evento del mismo nombre. Aquí usaremos el dominio WorkflowAction con el valor complete y asignado a la variable action. Después de eso, ejecutará la subrutina “Button Pressed”, así como en la acción Execute se cargara el Id del Workitem seleccionado en la variable del tipo de dato Workflow Workitem.

```
43  Event 'Complete'
44      &action = WorkflowAction.COMPLETE
45      Do 'Button Pressed'
46  EndEvent
47
```

```
78  Sub 'Button Pressed'
79          &workitem.Load(&id)
80          Do Case
81              Case &action = WorkflowAction.EXECUTE
82                  Do 'Execute'
83              Case &action = WorkflowAction.COMPLETE
84                  Do 'Complete'
85              Case &action = WorkflowAction.PREVIEW
86                  Do 'Preview'
87              Case &action = WorkflowAction.DELEGATE
88                  Do 'Delegate'
89              Case &action = WorkflowAction.COLLABORATE
90                  Do 'Collaborate'
91              Case &action = WorkflowAction.VIEW_HISTORY
92                  Do 'View History'
93              Case &action = WorkflowAction.ADD_COMMENTS
94                  Do 'Add Comments'
95              Case &action = WorkflowAction.VIEW_DOCUMENTS
96                  Do 'Documents'
97          EndCase
98      Commit
99      If &error.Code > 0
100         Do 'Error'
101     Endif
102 EndSub
```

Después de eso, ejecutará la subrutina Complete y en esta subrutina verificará que el workitem tenga el estado correcto antes de continuar. Todos los workitems a completar deben tener el estado OPEN_ACTIVE_INPROCESES que quiere decir que se ha ejecutado previamente.

Si tiene el estado correcto, completará el workitem, lo que significa que este workitem finalizará y se crearán los siguientes workitems del proceso. Si hay un error, por ejemplo, porque el usuario debe seleccionar una ruta opcional, se le solicitará seleccionar una actividad y elegir la ruta que debe de tomar. En caso de que el error se deba a que es un proceso Ad-hoc y debe seleccionarse también la siguiente actividad a ejecutar, pero si el caso es que el error se deba a que se requieren algunos comentarios en esta actividad, se abrirá un prompt del Workflow, para que el usuario ingrese los comentarios necesarios para completar esta tarea. Si hay un error, se le mostrará al usuario, en caso de que no haya, la tarea ha finalizado.

```
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
```

WORKFLOW INBOX – HISTORY ACTION

En la bandeja de entrada del Custom Client con el workitem preseleccionado en el grid y presionando el botón History, presentará el History Panel. Este panel presenta una lista de workitems que se han ejecutado desde la instancia del proceso. Para ello, se ejecutará el siguiente código del evento History, y este evento utilizará, al igual que las demás acciones de este panel, asignar en la variable action el dominio WorkflowAction con el valor VIEW_HISTORY y luego ejecutar la subrutina Button Pressed. En esta subrutina se cargará la variable workitem, con el Id que se ha seleccionado en el grid. Este es el Id de workitem, y luego tenemos el workitem asignado a la variable.

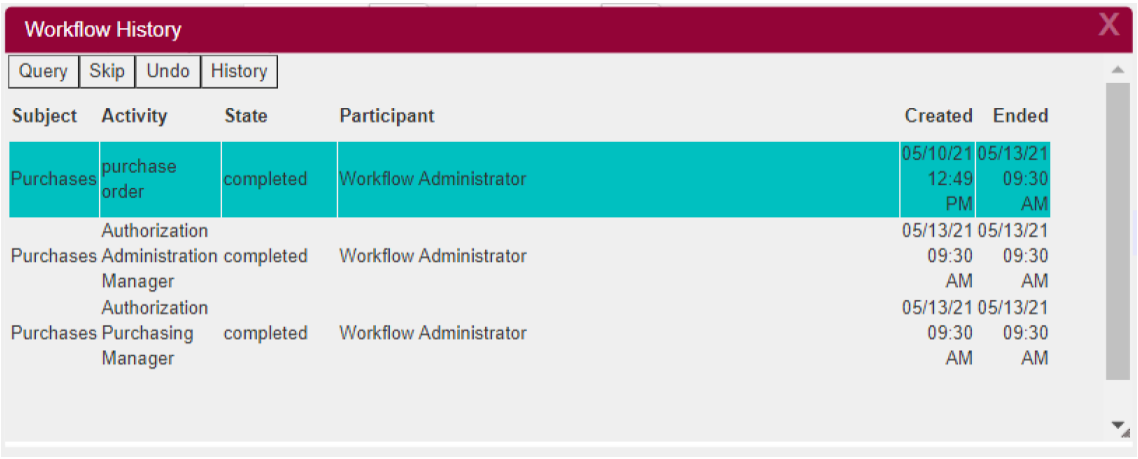
```
48  Event 'History'
49      &action = WorkflowAction.VIEW_HISTORY
50      Do 'Button Pressed'
51  EndEvent
```

```
78  Sub 'Button Pressed'
79          &workitem.Load(&id)
80          Do Case
81              Case &action = WorkflowAction.EXECUTE
82                  Do 'Execute'
83              Case &action = WorkflowAction.COMPLETE
84                  Do 'Complete'
85              Case &action = WorkflowAction.PREVIEW
86                  Do 'Preview'
87              Case &action = WorkflowAction.DELEGATE
88                  Do 'Delegate'
89              Case &action = WorkflowAction.COLLABORATE
90                  Do 'Collaborate'
91              Case &action = WorkflowAction.VIEW_HISTORY
92                  Do 'View History'
93              Case &action = WorkflowAction.ADD_COMMENTS
94                  Do 'Add Comments'
95              Case &action = WorkflowAction.VIEW_DOCUMENTS
96                  Do 'Documents'
97          EndCase
98      Commit
99      If &error.Code > 0
100         Do 'Error'
101     Endif
102 EndSub
```

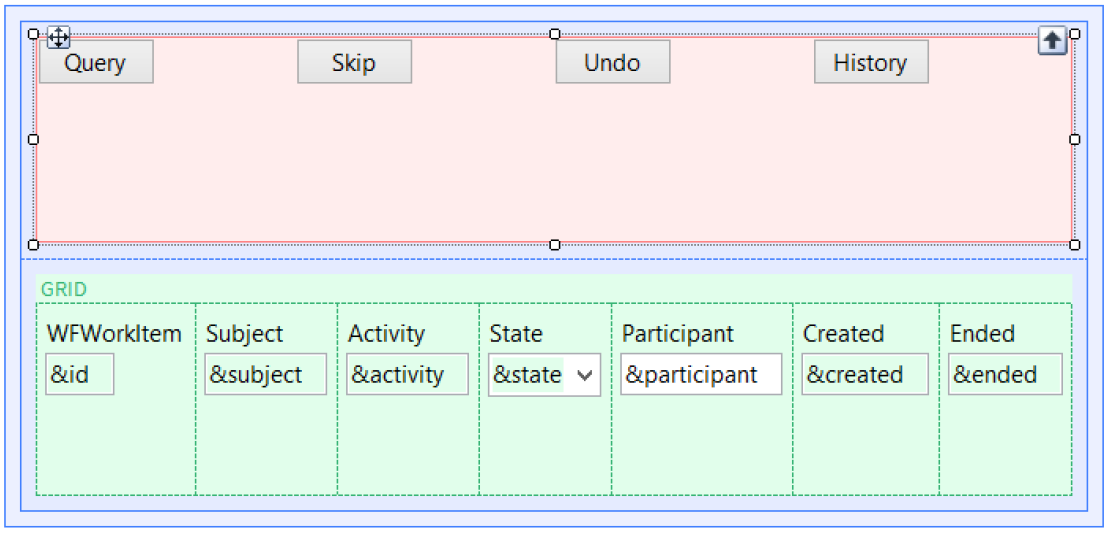
Después de eso, ejecutaremos la subrutina View History. En esta subrutina se usará la variable Window para abrir un Web component que se llama Workflow History. Este panel recibe por parámetro el Id de la instancia del proceso usando la variable workitem que se le ha asignado en la subrutina Button pressed. Esta variable tiene una propiedad que se llama ProcessInstanceId, que toma la instancia del proceso de este workitem y luego abrirá la página con la propiedad Open de la variable Window.

```
240 Sub 'View History'
241     &window.Object = WorkflowHistory.Create(&workitem.ProcessInstanceId)
242     &window.Open()
243 EndSub
```

Si observamos la ejecución de este panel, veremos que hay una lista todos los workitems que se han ejecutado de esta instancia de proceso donde se muestra el asunto, la actividad, el estado, el participante, cuando fue creado, cuando termino y una lista de acciones que pueden ser ejecutadas para estos workitems.



Si vemos el diseño del Web panel tiene un grid con todos los elementos que hemos mencionado, y una tabla con todas las acciones.



Si observamos el código, veremos en la sección reglas que recibe el parámetro &ProcessInstanceId.

En el Evento Start hará lo mismo que todos los paneles en el Custom Client, comprobará que hay una sesión válida y luego asignará el usuario conectado, que obtuvimos por la propiedad de ConnectedUser de la variable server del tipo WorkflowServer. Después de eso, se cargará en la variable e &ProcessInstance, que es del tipo de datos WorkflowProcessInstance, el Id de instancia del proceso que habrá recibido por parámetro. Posteriormente en el evento Refresh, obtendrá todos los workitems que se han creado para esta instancia, utilizando la propiedad workitems.

Esta propiedad devuelve una lista de workitems, son del tipo de dato WorkflowWorkitems y se asignan a la variable &workitem. Esta lista se iterará en el evento de carga y para cada workitem de esta lista, asignará sus valores al grid. Por ejemplo, asignará el Id del workitem, el nombre de la actividad, el estado, el sujeto de la instancia del proceso, el nombre del participante que ejecutó esta tarea, la fecha en que se creó y la fecha en que finalizó.

```
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
```

Cuando ejecutemos esta acción vamos a delegar un workitem que tendremos en nuestra bandeja de entrada a otra persona. Para ello, en el evento Delegate, así como en otras acciones, asignaremos WorkflowAction, pero con el valor DELEGATE en la variable &action y luego se ejecutara la subrutina "Button Pressed". En esta subrutina, cargará el Id de los workitems seleccionados en el grid a la variable workitem y ejecutará la subrutina Delegate. En esta subrutina vamos a comprobar que la actividad del workitem tiene habilitada la delegación de funciones. Puede habilitar la delegación seleccionando en el diagrama la actividad de la tarea y en las propiedades poniendo true en la propiedad "Allow delegation", esto habilitará la acción de delegación. Si la tarea puede delegar, presentará el flujo de trabajo del panel asignado al usuario. Este panel presentará todos los usuarios que pueden recibir esta tarea.

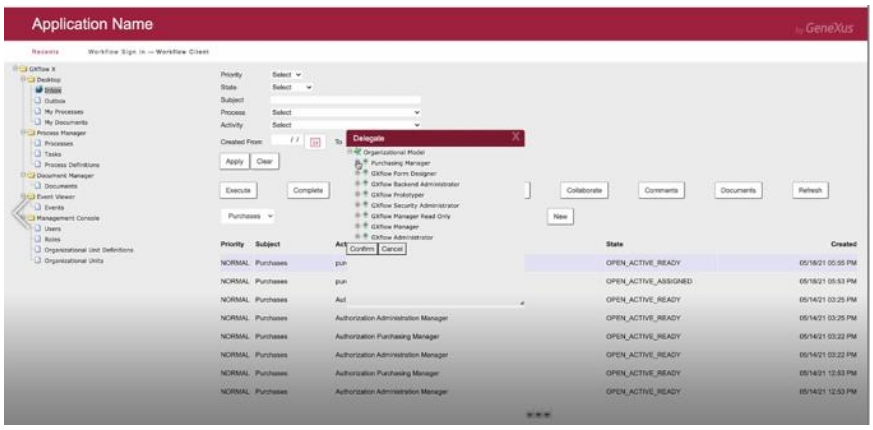
```
58  Event 'Delegate'
59        &action = WorkflowAction.DELEGATE
60        Do 'Button Pressed'
61  EndEvent
62
```

```
78  Sub 'Button Pressed'
79        &workitem.Load(&id)
80        Do Case
81              Case &action = WorkflowAction.EXECUTE
82                    Do 'Execute'
83              Case &action = WorkflowAction.COMPLETE
84                    Do 'Complete'
85              Case &action = WorkflowAction.PREVIEW
86                    Do 'Preview'
87              Case &action = WorkflowAction.DELEGATE
88                    Do 'Delegate'
89              Case &action = WorkflowAction.COLLABORATE
90                    Do 'Collaborate'
91              Case &action = WorkflowAction.VIEW_HISTORY
92                    Do 'View History'
93              Case &action = WorkflowAction.ADD_COMMENTS
94                    Do 'Add Comments'
95              Case &action = WorkflowAction.VIEW_DOCUMENTS
96                    Do 'Documents'
97        EndCase
98     Commit
99     If &error.Code > 0
100          Do 'Error'
101    Endif
102 EndSub
```

```
241 Sub 'Delegate'
242       If &workitem.Activity.canDelegate = True
243             &window.Object = WorkflowAssign.Create(&workitem.Id, WorkflowAction.DELEGATE)
244             &window.Open()
245       Else
246             msg('Operation not allowed')
247       Endif
248 EndSub
```

Si observamos la función de delegar en ejecución, podemos ver que si seleccionamos nuestro workitem del grid de la bandeja de entrada de nuestro flujo de trabajo y luego presionamos el botón delegar, aparecerá una página web con todo el modelo organizacional y todos los roles, que puede tener la Knowledge base, se puede abrir un rol, ver los usuarios y seleccionar uno confirmar y esta tarea se delegará a ese usuario.

Este Web panel se crea con un objeto Window y recibe dos parámetros, el Id workitem, que se ha cargado en la subrutina de Button Pressed y la acción, en este caso Workflow Action delegate.



En el layout vamos a ver que es diferente al resto de paneles, tiene tres controles y dos botones, confirmar y cancelar.

Si vemos el código, veremos que recibe dos parámetros, Id del workitem y action. En el evento Start comprobará la sesión de flujo de trabajo con el procedimiento WorkflowCheckServerSession al igual que otros paneles. Luego obtendrá del WorkflowServer que se ha cargado el Modelo Organizacional, que es parte del servidor. Con el método Organizational Model esto devolverá una variable del tipo de datos WorkflowOrganizationalModel, que en este caso será la variable OrgModel, luego cargará el parámetro de recepción de workitems a la variable workitem

.

Luego se hace Refresh y se invoca a la subrutina “Populate Tree”. Esta subrutina lista todos los roles usando el método “list roles”, este recibe un parámetro de filtro, en este caso está vacío, y recibirá aquí todos los roles del Modelo Organizacional. Cargará la ruta de todos los roles del árbol y para cada rol de la lista también cargará la información asociada.

```
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

Cuando se está asignando un elemento porque seleccionamos un usuario o rol y luego presionamos confirmar validará que el Nodo del Árbol no está vacío y la acción no es colaborar, obtendrá el rol seleccionado en el árbol con el ID seleccionado y la variable, luego obtendrá el rol, en el Modelo Organizacional, obtendrá su ID. Luego, obtendremos el rol de flujo de trabajo en la variable Role, si no hay ningún error, lo asignará. En caso de que estemos asignando a un usuario, debemos obtenerlo mediante el Modelo Organizacional usando GetUserById, y luego, si no hay error, validaremos la acción. En el caso de que la acción sea delegada, usaremos la variable Workitem.DELEGATE, este método delegará este workitem a dicho usuario. Si no hay error se hace commit para guardar todos los cambios y luego regresar a la página web.

```
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
```

En este video hemos visto el uso de Custom Client, cómo podemos importarlo en una Knowledge base, el por qué lo usamos y algunas de sus acciones.