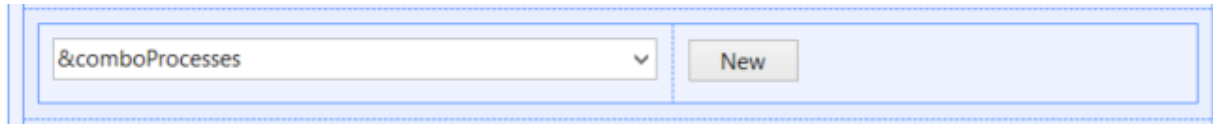


Custom Client Part 2

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.



```
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
```

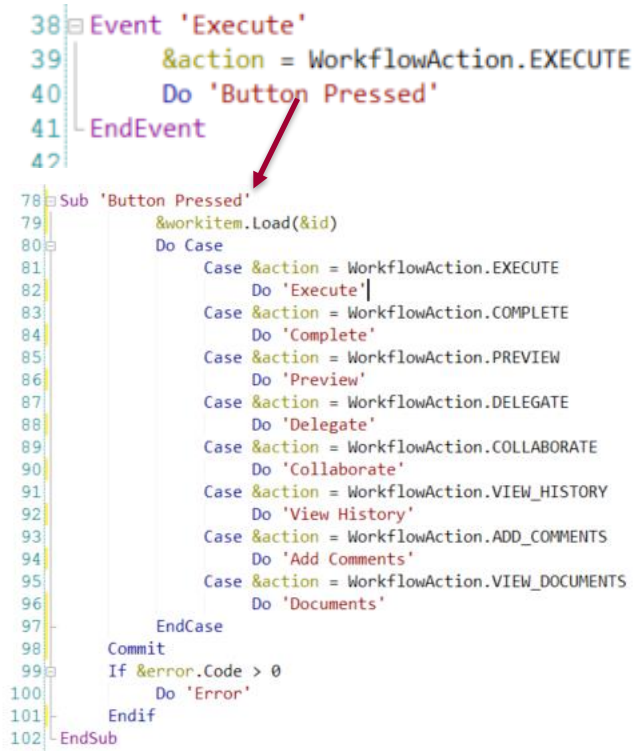
This will execute a subroutine called new. In this subroutine we are going to have a variable of the process definition data type and will execute the load method of this process definition data type. These functions loads a process definition into this variable, which process definitions The one passed 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.

```
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
```



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 of 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.

```

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     Otherwise
162       &error.Code = 200 //Invalid transition
163     EndCase
164   EndSub
165
166

```

After the take method, it will change the workitem's state using the method ChangeState and it will change to OPEN_ACTIVE_INPROCESSES, 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.

```
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
```

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

```
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
```

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.

```

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

```

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.

```

77 Sub 'Complete'
78     If &workitem.State = WorkflowWorkitemState.OPEN_ACTIVE_INPROCESS
79         &workitem.Complete()
80         &error = &workitem.Error
81         If &error.Code > 0
82             Do Case
83                 Case &error.Code = WorkflowError.OPTIONALS_SELECTION_REQUIRED
84                     &window.Object = WorkflowSelectActivity.Create(&workitem.Id, WorkflowSelectionMode.OPTIONALS, WorkflowAction.COMPLETE)
85                     &window.Open()
86                     &error = new()
87                 Case &error.Code = WorkflowError.ADHOC_SELECTION_REQUIRED
88                     &window.Object = WorkflowSelectActivity.Create(&workitem.Id, WorkflowSelectionMode.ADHOC, WorkflowAction.COMPLETE)
89                     &window.Open()
90                     &error = new()
91                 Case &error.Code = WorkflowError.COMMENTS_REQUIRED
92                     &window.Object = WorkflowComments.Create(&workitem.Id, WorkflowObjectType.WORKITEM, False)
93                     &window.Open()
94                     &error = new()
95             EndCase
96             Do 'Error'
97         Endif
98     Else
99         &error.Code = 204 // The task has not been processed yet
100     Endif
101 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.

```

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

```

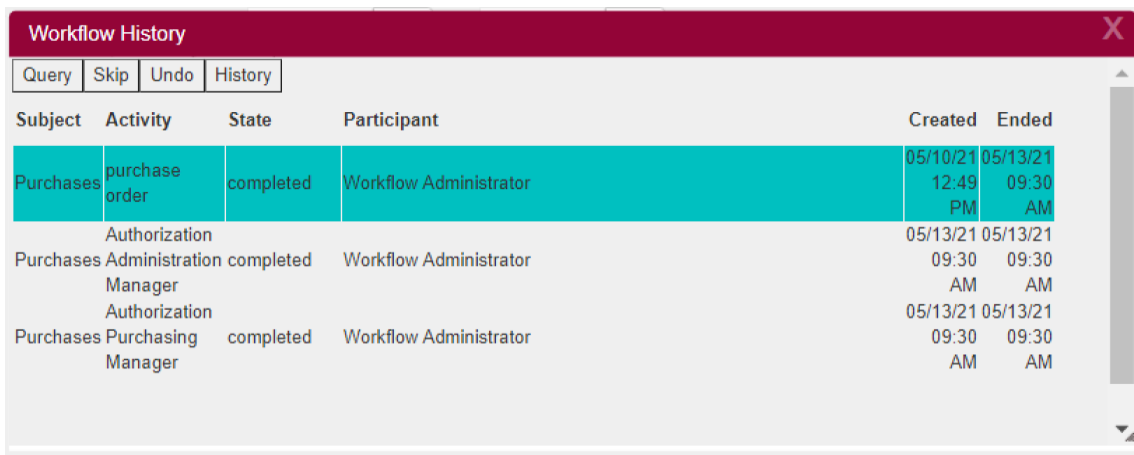
After that we will execute the event subroutine View History. In this subroutine it will use the window variable to open a web component, which is called Workflow History. This panel have received by parameter the process instance Id, and here we are going to pass this process instance Id using the workitem 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.

```

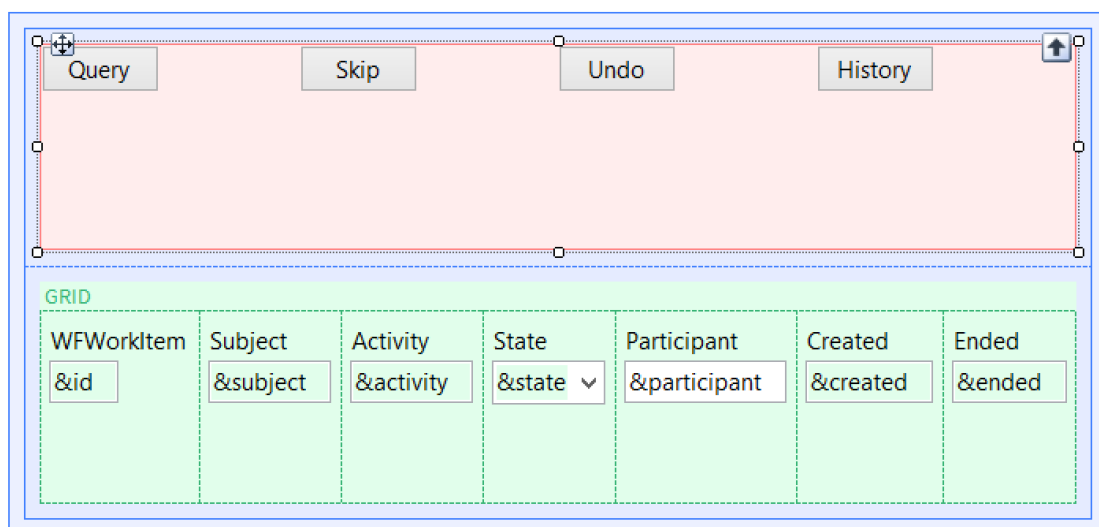
240 Sub 'View History'
241     &window.Object = WorkflowHistory.Create(&workitem.ProcessInstanceId)
242     &window.Open()
243 EndSub

```

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.



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.



WORKFLOW 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

```
1  Event Start
2      &server = WorkflowCheckServerSession()
3      &user = &server.ConnectedUser
4      &processInstance.Load(&processInstanceId)
5  EndEvent
6
7  Event Grid.Refresh
8      &workitems = &processInstance.Workitems
9  EndEvent
10
11 Event Grid.Load
12     For &workitem in &workitems
13         &id = &workitem.Id
14         &activity = &workitem.Activity.Name
15         &state = WorkflowWorkitemState.Convert(&workitem.State)
16         &subject = &workitem.ProcessInstance.Subject
17         &participant = &workitem.Participant.Name
18         &created = &workitem.Created
19         &ended = &workitem.Ended
20
21         Grid.Load()
22     Endfor
23 EndEvent
24 |
```


WORKFLOW INBOX – DELEGATE ACTION

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.

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

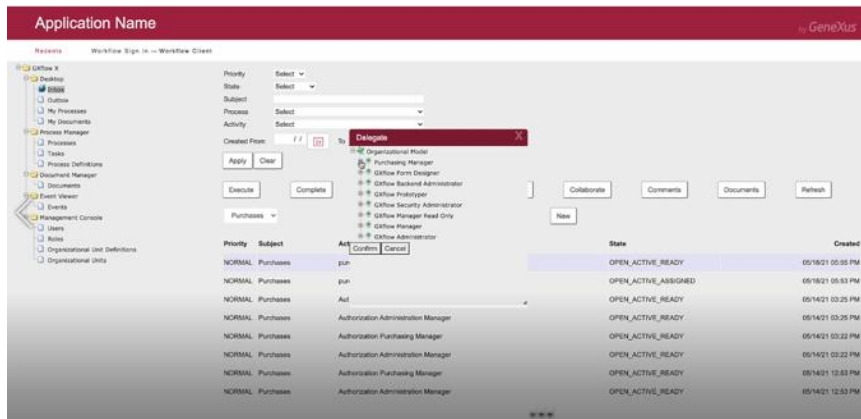


```
78 | Sub 'Button Pressed'  
79 |     &workitem.Load(&id)  
80 |     Do Case  
81 |         Case &action = WorkflowAction.EXECUTE  
82 |             Do 'Execute'  
83 |         Case &action = WorkflowAction.COMPLETE  
84 |             Do 'Complete'  
85 |         Case &action = WorkflowAction.PREVIEW  
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  
249 |
```

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.



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

```

Web Form | Rules | Events | Conditions | Variables | Help | Documentation |
1 param(in: &workitemId, in: &action);
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Workflow Data Type Class Hierarchy

- Server
 - Process Definition
 - Activity
 - Process Instance
 - Workitem
 - Application Data
 - Organizational Model
 - Role
 - User
 - Organizational Unit

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.

```
127 Sub 'Populate Tree'
128
129     &roles = &orgModel.ListRoles(&filter)
130
131     &root.Id = !"OM"
132     &root.Name = "Organizational Model"
133     &root.Icon = WorkflowOrganizationalModel.Link()
134     &root.IconWhenSelected = WorkflowOrganizationalModel.Link()
135     &root.Expanded = True
136     &treeNodeCollectionData.Add(&root)
137
138     For &role in &roles
139
140         If &role.hasParent = False
141
142             &treeNode = new()
143             &treeNode.Id = Trim(Str(&role.Id))
144             &treeNode.Name = &role.Name
145             &treeNode.Icon = WorkflowRole.Link()
146             &treeNode.IconWhenSelected = WorkflowRole.Link()
147             &treeNode.DynamicLoad = True
148             &root.Nodes.Add(&treeNode)
149
150         Endif
151
152     Endfor
153
154 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.

```
62 Event Enter
63   If Not &selectedTreeNode.Id.IsEmpty()
64     If &selectedTreeNode.Id.ToNumeric() > 0 //Role
65       If &action <> WorkflowAction.COLLABORATE
66         &role = &orgModel.GetRoleById(&selectedTreeNode.Id.ToNumeric())
67         &error = &orgModel.Error
68         If &error.Code = 0
69           If Block
70             &error = &workitem.Error
71           Endif
72         Endif
73       Else // User
74         &user = &orgModel.GetUserById(&selectedTreeNode.Id)
75         &error = &orgModel.Error
76         If &error.Code = 0
77           Do Case
78             Case &action = WorkflowAction.COLLABORATE
79               &workitem.Collaborate(&user)
80             Case &action = WorkflowAction.DELEGATE
81               &workitem.Delegate(&user)
82             Case &action = WorkflowAction.REASSIGN
83               &workitem.Reassign(&workitem.Participant, &user)
84           EndCase
85           &error = &workitem.Error
86         Endif
87       Endif
88     Commit
89     If &error.Code = 0
90       Return
91     Else
92       Do 'Error'
93     Endif
94   Endif
95 EndEvent
96
```

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.