

finPOWER Connect 3 Workflows (Version 2)

Version 3.02
9th July 2019

Table of Contents

- Disclaimer..... 5
- Version History 6
- Introduction 7
- Workflow Type Version 2 Overview 8
- Workflow Type Changes 9
 - Account Applications 9
 - Current Workflow Item 9
 - CanActionItem Script Event..... 9
 - Workflow Recall Date..... 9
 - 'Open' Workflow Items 9
 - Outcomes on 'None' Workflow Items 10
- Workflow Summary Script (Version 2)..... 11
 - Workflow Type Options 11
 - Show Completed Item Groups as a single group 11
 - Show last Item Group as a separate group 12
 - Show future Item Groups 12
 - Show Items and Item Groups flagged as 'Not Applicable' 12
 - Show skipped Items and Item Groups 12
 - Show Train Stops Diagram 12
 - Train Stops Diagram 13
- Workflow Type Script 15
 - Completing Workflow Items from Script Code 17
 - Auto Actioning Workflow Items from Script Code 18
 - CompleteItem versus AutoActionItem..... 19
 - BeforeItemAction versus AfterItemAction 20
 - Automatic Questions and Item Groups..... 21
 - Script Events 22
 - AfterInitialise 23
 - AfterItemGroupAdd 24
 - BeforeProcess 26
 - AfterItemGroupBegin 27
 - BeforeItemAction 29
 - CouldActionItem 31
 - AfterItemAction 32
 - BeforeItemReset 34
 - AfterItemReset 35
 - BeforeClose 36
 - CanActionItem 37
 - BeforeAccountDecline 38
 - UpdateDocumentsList 39
 - GetTrainStops..... 40

GetWorkflowItemUserDataSummary	42
GetWorkflowUserDataSummary	44
Script Responsibilities	45
Recording Logs	45
Recording New Workflows	45
Executing External Web Services	45
Database Transactions.....	47
Workflow Type Script Examples	48
Set Workflow Items to 'Not Applicable' When Initialising	48
Create New Item Group with Items from Script	49
Skip Items in Group and Complete Group	50
Skip Current Item Group and Add a new Item Group.....	51
Set Check List Items Based Upon Question Outcome	52
Creating Customised Documents.....	54
Creating Customised Email Documents	56
Creating Ad-Hoc Emails	58
Automatic Client Credit Enquiry - Veda (New Zealand).....	60
Automatic Client Credit Enquiry – GreenId	64
Automatic Applicant Credit Enquiry - Centrix (New Zealand).....	66
Documents	68
Workflow Type Item Wizard	69
Workflow Type Scripts	69
'Send Document' Type Items	69
Fully Scripted Solution	69
Document Scripts	69
Account SMS	71
Account Email.....	73
Client SMS	75
Client Email.....	76
Account Application SMS.....	78
Account Application Email	80
HTML Widgets	82
Workflow Type Item	82
HTML Widget.....	82
Page Sets	85
Workflow Type Item	85
Page Set.....	85
Appendix A – Helper Functions	88
Workflow Functions (finWorkflowFunctions).....	88
Adding and Inserting Workflow Items	89
Skipping Items and Item Groups.....	91
Setting Items and Item Groups to 'Not Applicable'	92
Appendix B – Frequently Asked Questions.....	93

Workflow Items	93
How do I get the Outcome of a Workflow Item?	93
Why Does Completing an Item Group from 'BeforeItemAction' Skip the Current Item?...	93
Workflow Item Groups.....	94
How do I add an Item Group from outside of the Workflow Type Script?	94
Miscellaneous	95
How Do I Refresh Workflows on another Form, e.g., the Task Manager?.....	95

Disclaimer

This document contains information that may be subject to change at any stage.

All code examples are provided "as is".

This document may reference future functionality not currently available in the release version of finPOWER Connect.

Copyright Intersoft Systems Ltd, 2015.

Version History

Date	Version	Name	Changes
30/01/2015	1.00	PH	Created.
25/02/2015	1.01	PH	Updated with more examples and appendices.
24/03/2015	1.02	PH	StatusUpdatedFromScript property and GetTrainStops event added.
31/03/2015	1.03	PH	Updated Appendix A to list methods to skip or 'Not Applicable' items and item groups.
06/05/2015	1.04	PH	Added more details to "AfterItemGroupAdd" event.
13/05/2015	1.05	PH	FAQ for refreshing Workflows in Task Manager plus AfterItemGroupBegin and GetWorkflowItemUserDataSummary events.
19/08/2015	1.06	PH	Document Script functionality changes in version 2.03.02.
22/12/2015	1.07	PH	Added note that automatic Credit Enquiry samples are unrelated to 'Credit Enquiry' type Workflow Items.
12/07/2016	3.00	PH	Updated for finPOWER Connect version 3.
13/06/2017	3.01	PH	Added "HTML Widget" type Workflow Items.
09/07/2019	3.02	PH	BeforeItemAction vs AfterItemAction information added.

Introduction

This document discusses finPOWER Connect Version 3 Workflows (Version 2) and is focused on the creation, configuration and scripting of Workflows.

Workflows require that finPOWER Connect is licensed for the Workflows Add-On.

NOTE: This document discusses Version 2 type Workflows only.

Certain functionality may not be applicable to Version 1 type Workflows or may behave differently. Differences between Version 1 and Version 2 type Workflows are noted where applicable.

Workflow Type Version 2 Overview

- Version 2 type Workflows are processed differently:
 - A `finWorkflowExecutor` object is used to process Workflows and to action Workflow items.
 - ✦ Version 1 type Workflows were processed exclusively via methods in `finWorkflowFunctions`.
- The Workflow Type Script can no longer action other Workflow Items in the same way as Version 1 type Workflows:
 - See the [Completing Workflow Items from Script Code](#) section.
 - Version 2 Workflows will need to be structured differently.
- Question type Workflow Items and Item Groups can now be automatic and the ['BeforeItemAction'](#) event used to set the outcome.

Workflow Type Changes

This section outlines the changes (from Version 1) and new options available to Version 2 Workflows.

Account Applications

- Account Applications can only use Version 2 Workflow Types.

Current Workflow Item

- `finWorkflowItem.CurrentItemIndex` property for an Item Group gives the index of the Current Workflow Item.
- In Version 1 type Workflows, the Current Item is:
 - The first Workflow Item in the Item Group with a status of either 'Not Started' or 'Open'.
- In Version 2 type Workflows, the Current Item is:
 - Determined in the following order:
 - ✦ The first Workflow Item in the Item Group with a status of either 'Not Started' that has its `AutoAction` property set to `True`.
 - ✦ The first Workflow Item in the Item Group with a status of 'Open'.
 - ✦ The first Workflow Item in the Item Group with a status of 'Not Started'.
 - Therefore, unlike Version 1 Workflows, any 'Open' item will be the Current item, regardless of whether there are any incomplete items before it.

CanActionItem Script Event

- In Version 1 type Workflows, this event is only called if the Workflow Item passes all of the built-in checks to determine whether it can be actioned.
- In Version 2 type Workflows, this event is called for all incomplete items, regardless of whether the built-in checks passed or failed.
 - This allows the Script to provide a customised error message as to why the item cannot be actioned.
 - **NOTE:** The Script cannot force an item to be actionable if it has failed built-in checks; just customise the error message.

Workflow Recall Date

- In Version 1 type Workflows, the Recall Date is cleared whenever an item is actioned, regardless of whether there is an 'Open' item such as a Wait in progress.
- Version 2 Workflow Types have a 'Clear Recall Date when completing any other item in this group even if there is an 'Open' item' property on the Item Group to determine whether to clear the Recall Date or not.

Specify Group options.

- Must the User complete all items in this Item Group in order?
- User can only flag as 'Not Applicable' in order?
- Clear Recall Date when completing any item in this Item Group even if there is an 'Open' item?

NOTE: This applies to 'Open' items, i.e., if there is an 'Open' item in the group, the Workflow's Recall Date will NOT be cleared unless this option is checked.

'Open' Workflow Items

- 'Open' Workflow Items are those in a special 'Open' state.

- These are completed automatically upon processing the Workflow (or actioning another item) if the Workflow's recall date has been passed.
- The exception to this is 'Review' type items that must be manually completed.
- In Version 1 type Workflows, only 'Wait' and 'Review' type items could be 'Open'.
- Version 2 type Workflows have a new Outcome Action of 'Wait'.
 - Therefore, these additional item types can now be 'Open':
 - ✦ Question
 - ✦ Decision Card
 - ✦ Bank Account Enquiry Review
 - ✦ Test
 - Upon first actioning these items, where applicable, status notes etc. can be entered.
 - ✦ This will set the item into an 'Open' state.
 - Action the one of these items when it is 'Open' will simply close it.
 - ✦ No further prompt is given to enter status notes etc.

Outcomes on 'None' Workflow Items

Although 'None' Workflow Items do not specify a list of outcomes, you can set the Outcome via Script code in the 'BeforeItemAction' event, e.g.:

```
Case "BeforeItemAction"
  If workflowItem.ItemIdOriginal = "SC1" Then
    With otherParameters
      .SetString("Outcome", "CustomOutcome")
      .SetString("StatusNotes", "Custom status notes")
      .SetBoolean("OutcomeSuccess", False)
      .SetString("OutcomeIcon", "Neutral")
    End With
  End If
```

This sets the `StatusOutcome` property on the `finWorkflowItem` object and also generate a single Outcome Item so assign the success status and icon to.

Workflow Summary Script (Version 2)

A new, built-in Summary Script is used for Version 2 type Workflows.

NOTE: Version 1 Workflows still use the old Summary Script by default. This behaviour may be changed in a future version when Version 2 Workflows have stabilised.

Workflow Type Options

Instead of just using Script constants to define how the Summary Page should appear, the Workflow Type has a 'Summary Script' page that defines options that can be used by the Summary Page Script. All of these are used by the built-in Standard Blocks.

Summary Page options. ⓘ

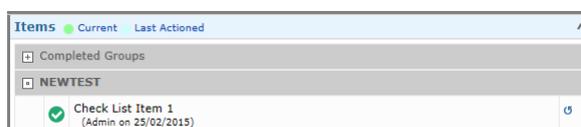
- Show completed Item Groups as a single group?
 - Show last Item Group as a separate group?
- Show future Item Groups?
- Show Items and Item Groups flagged as 'Not Applicable'?
- Show skipped Items and Item Groups?
- Show 'Train Stops' diagram?

Show Completed Item Groups as a single group

- If a Workflow has lots of groups added to it through its lifetime, this can make the 'Items' block, through which must Users action items, a little unwieldy, e.g.:

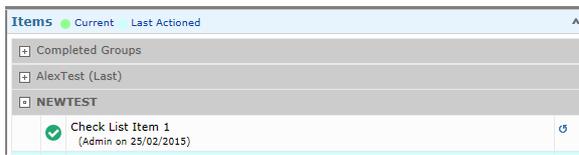


- By grouping all completed Item Groups together, the above becomes more manageable, e.g.:



Show last Item Group as a separate group

- Whilst lumping all completed Item Groups into a single group make the Summary Page more compact, often, the Workflow User may be especially interested in the last completed Item Group.
- This option separates the last completed Item Group into its own group, e.g.:



Show future Item Groups

- This allows any future Item Groups to be displayed in the Item block.
- Future groups always appear after the current Item Group and are collapsed.

Show Items and Item Groups flagged as 'Not Applicable'

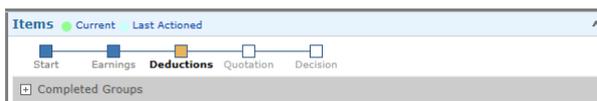
- By default, Items and Item Groups that have been flagged as 'Not Applicable' are hidden from the Items block.
- The exception to this any Item Groups that contain one or more completed items.
 - These will never be hidden, regardless of this option.

Show skipped Items and Item Groups

- By default, Items and Item Groups that have been skipped are hidden from the Items block.
- The exception to this any Item Groups that contain one or more completed items.
 - These will never be hidden, regardless of this option.

Show Train Stops Diagram

- This option forces a 'Train Stops' diagram to be displayed at the top of the Items block.
- By default, this lists all Item Groups (completed and future), e.g.:



- Clicking one of the 'Train Stops' will display a summary of that Item Group including all of the items within it.
 - This summary does not allow any items to be actioned.

Train Stops Diagram

The `finWorkflow` object has a `GetTrainStops` method that returns a `finWorkflowTrainStops` collection.

This is used to generate the Train Stops diagram.

The Workflow Type Script's '[GetTrainStops](#)' event can be used to tweak or completely override this collection.

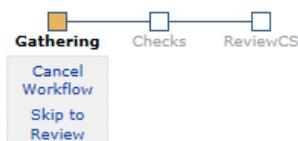
By default, the diagram will be created from all of the Workflow's Item Groups excluding the following:

- Deleted Item Groups
- Skipped Item Groups
 - Unless the Workflow Type is configured to Show Skipped Item Groups or the Skipped Item Group contains one or more completed Workflow Items.
- Item Groups flagged as 'Not Applicable'
 - Unless the Item Group contains one or more completed Workflow Items.

The diagram will look something like this:



And, if customised (via the '[GetTrainStops](#)' event) to include Actions, something like this:

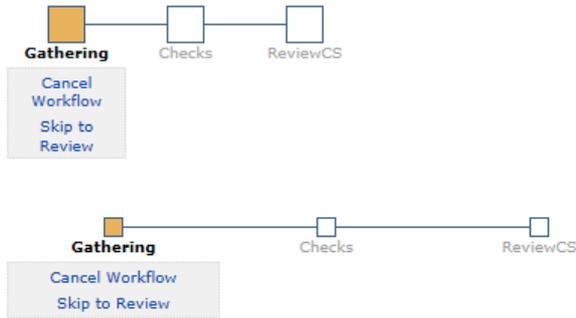


WARNING: The built-in HTML generated for the Train Stops diagram is only compatible with Internet Explorer 10 and above.

The `finWorkflowTrainStops` object has the following properties to tweak the diagram:

- `BulletShape`
 - Determines the bullet shape, either:
 - ✧ Default (Square)
 - ✧ Square
 - ✧ Circle
 - **NOTE:** Can be overridden on each Train Stop.
- `BulletSize`
 - The Bullet Size in pixels (use an odd number so the 1 pixel joiner line is correctly centred) or zero to use the default (13 pixels).
- `JoinerHalfSize`
 - The half size of the Bullet Joiner lines in pixels or zero to use the default (27 pixels).
 - The 'half size' is specified since the joiner line consists of two parts; one part that appears before the bullet and one part that appears after the bullet.

Tweaking these allows the following types of style changes which, depending on how the Workflow is configured, may make for a clearer diagram:



The `finWorkflowTrainStop` object has the following properties to tweak a particular Train Stop in the diagram:

- `BulletColour`
 - A custom HTML colour for the bullet or a blank String to use the automatically determined colour.
- `BulletShape`
 - Determines the bullet shape, either:
 - ✧ `Default` (as defined on the Train Stops collection)
 - ✧ `Square`
 - ✧ `Circle`
- `BulletStyle`
 - The underlying bullet style, either:
 - ✧ `Automatic` (determined from the Workflow Item Group)
 - ✧ `Completed`
 - ✧ `Current`
 - ✧ `Future`
 - ✧ `SkippedAll`
 - ✧ `SkippedPartial`
- `CaptionColour`
 - A custom HTML colour for the caption or a blank String to use the automatically determined colour.
- `JoinerHalfSize`
 - The half size of the Bullet Joiner line in pixels or zero to use the value specified on the Train Stops collection.

An example of tweaking these properties is shown below:



NOTE: To generate a totally customised Train Stops diagram, a Script must be defined under Global Settings, Summary Pages to override Standard Summary Page Blocks and then override the `Workflow_TrainStops` method.

Workflow Type Script

Much of the processing of Version 2 type Workflows occurs via a Workflow Executor.

The following events can access the Workflow Executor:

- AfterInitialise
- AfterItemGroupAdd
- BeforeProcess
- BeforeItemAction
- AfterItemAction
- BeforeItemReset
- AfterItemReset

The template Script code assigns a reference to the Workflow Executor, i.e.:

```
Private mWorkflowExecutor As finWorkflowExecutor

Public Function Main(workflow As finWorkflow,
                    eventId As String,
                    ByRef eventHandled As Boolean,
                    workflowItem As finWorkflowItem,
                    otherParameters As ISKeyValueList) As Boolean

    Dim Outcome As String
    Dim StatusNotes As String

    ' Assume Success
    Main = True

    ' Get Workflow Executor
    mWorkflowExecutor = DirectCast(otherParameters.GetObject("WorkflowExecutor"),
    finWorkflowExecutor)

End Function
```

NOTE: The Workflow Executor is used to perform much of the functionality that Version 1 type Workflows used in `finWorkflowFunctions` (`finBL.WorkflowFunctions`).

The Workflow Executor contains many helper functions to make it easy to complete, skip, or make items 'Not Applicable'. Many of these methods allows different ways of identifying a Workflow Item, e.g.:

- `WorkflowExecutor.ItemSetStatusSkipped`
- `WorkflowExecutor.ItemSetStatusSkippedByItemId`
- `WorkflowExecutor.ItemSetStatusSkippedByOriginalItemId`

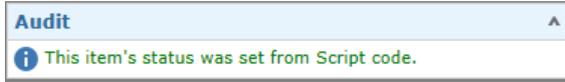
NOTE: When identifying a Workflow Item by its Item Id or Original Item Id, only the first Workflow Item with a matching Id in the current Item Group is returned or used.

When a Workflow Item's Status has been updated from one of the following Workflow Executor helper methods, its `StatusUpdatedFromScript` property will be set to `True`:

- `CompleteItem`
- `CompleteCheckListItem`

- ItemSetStatusNotApplicable
- ItemSetStatusSkipped

The Summary Page Block displaying Audit information for the Workflow Item will indicate this, as will an overlay on the Workflow Item's Status Icon, e.g.:



Resetting a Workflow Item will record a 'StatusUpdatedFromScript' entry if the item being reset had had its Status updated from a Script.

Completing Workflow Items from Script Code

Unlike Version 1 type Workflows, Actioning of Workflow Items from a Workflow Type Script is restricted.

The following methods of the Workflow Executor allow Workflow Items to be completed, i.e., their Status and Outcome to be set.

- `CompleteItem`
 - Complete a Workflow Item.
- `CompleteItemById`
 - Complete a Workflow Item using the Item's Id to identify it.
- `CompleteItemByOriginalItemId`
 - Complete a Workflow Item using the Item's Original Id to identify it.
- `CompleteCheckListItem`
 - Complete a Workflow Item.
- `CompleteCheckListItemById`
 - Complete a Workflow Item using the Item's Id to identify it.
- `CompleteCheckListItemByOriginalItemId`
 - Complete a Workflow Item using the Item's Original Id to identify it.
- `CompleteOpen`
 - Complete any currently 'Open' item (e.g., a Wait).
 - This will not return an error (i.e., `False`) if there is no 'Open' item.

WARNING: None of the above methods actually action the Workflow Item, call Script events and, in the case of Item types defining Outcomes (e.g., Questions), none of the Outcome Actions will be performed.

This is a major change from Version 1 type Workflows and means that generally, Version 2 type Workflows will require a different structure.

See the next section on 'Auto Actioning' Workflow Items.

Auto Actioning Workflow Items from Script Code

As seen in the previous section, completing Workflow Items from Script code does not actually action the item, it simply sets the item's status and, optionally, outcome.

Workflow Items can however be flagged to 'Auto Action' from Script code. This will cause an item to be actioned as if the User is manually actioning it.

The process is as follows:

- The Workflow Type Script uses one of the `AutoActionItem` methods to set the Workflow Item's `AutoAction` flag and details (e.g., the desired outcome).
 - **NOTE:** The Workflow Item is not actually actioned at this point; its status will still be 'Not Started'.
- From the Workflows form, the User will be unable to action any Workflow Items until the Workflow is processed if the current Item Group has one or more incomplete 'Auto Action' items.
 - Generally, the User will never be in this position since 'Auto Action' items will be processed automatically. Exceptions are:
 - ✘ Where an Item Group has been manually added by the User from the Items page of the Workflows form.
 - ✘ Where the User has added a new Workflow but NOT processed it and either the 'AfterInitialise' or 'AfterItemGroupAdd' events have set one or more items to 'Auto Action'.
 - ✘ Where items in future groups have been set to 'Auto Action'.

The following methods of the Workflow Executor allow Workflow Items to be flagged to 'Auto Action':

- `AutoActionItem`
 - Sets the Workflow Item's Auto Action properties.
- `AutoActionItemById`
 - Sets the Workflow Item's Auto Action properties using the Item's Id to identify it.
- `AutoActionItemByOriginalItemId`
 - Sets the Workflow Item's Auto Action properties using the Item's Original Id to identify it.

WARNING: Preventing the User from actioning items if 'Auto Action' items exist in the group is achieved via the Workflow Summary Script and is only supported in the Summary Page (Version 2) Script.

CompleteItem versus AutoActionItem

Both the `CompleteItem` and `AutoActionItem` methods complete an item, i.e., set its status to complete and set its outcome (where applicable).

The main differences are:

- `CompleteItem` sets the Workflow's status and outcome immediately; `AutoActionItem` does not.
- `CompleteItem` does NOT perform any of the Workflow Items actions or process the item in any way, i.e., NONE OF THE FOLLOWING WILL TAKE PLACE:
 - Updates to the Workflow's Flag Colour.
 - Updates to Account Monitor Categories.
 - Script events such as `BeforeItemAction` and `AfterItemAction`.
- `AutoActionItem` performs all Workflow Item actions as if the item were being actioned by the User, therefore all of the following will take place:
 - Updates to the Workflow's Flag Colour.
 - Updates to Account Monitor Categories.
 - Script events such as `BeforeItemAction` and `AfterItemAction`.

NOTE: Because `AutoActionItem` does not action the Workflow Item immediately, it can be used to flag Workflow Items that make Web Service calls (e.g., perform a Bank Account Enquiry request) to be actioned automatically, thereby avoiding any database transaction nesting issues associated with Version 1 type Workflows.

BeforeItemAction versus AfterItemAction

Both the `BeforeItemAction` and `AfterItemAction` events run when actioning a Workflow Item.

The main uses and things to be aware of are:

- `BeforeItemAction`
 - This runs before the Workflow Item has been actioned. Therefore, common uses might be:
 - ✦ Prevent the item from being run (return `False`).
 - ✦ Completing an item automatically.
- `AfterItemAction`
 - This runs after the Workflow Item has been actioned.
 - ✦ Therefore, if using the `workflow.GetCurrentItem()` method, the `finWorkflowItem` object returned WILL NOT BE THE SAME AS the `workflowItem` parameter passed to this event.
 - ✦ Returning `False` will NOT prevent the item from being actioned since it has already been actioned.
 - Use with caution if this is being used by the last item in a group since the group may well have been closed by the time this event occurs meaning that no other changes can be made to it.

Automatic Questions and Item Groups

In Version 2 type Workflows, Question and Item Groups can be flagged as 'automatic'.

These are then treated like any other 'automatic' item, e.g., when they become current (i.e., in the case of Item Groups, all other items in the group have been actioned), the 'BeforeItemAction' event will be fired and the Script can then be used to determine the outcome.

NOTE: Limited automatic completion of Item Groups was supported in Version 1 Workflows via the use of special tags, e.g. '[All Success]' which are available from the Outcome dropdown in the Workflow Item Outcome wizard.

This functionality is still supported in Version 2 Workflows and, as per Version 1 Workflows, items using these tags should not be flagged as 'automatic'.

See the [BeforeItemAction](#) section for a code example.

If the Script receives an Outcome parameter, this implies that the Workflow Item is being actioned manually by the User (or some other processes). Therefore, the following code check allows you to determine whether to attempt to automatically set the Outcome:

```
Case "BeforeItemAction"  
  Outcome = otherParameters.GetString("Outcome")  
  If Len(Outcome) = 0 Then  
    ' Automatically determine the outcome  
    otherParameters.SetString("Outcome", "My Outcome Value")  
  Else  
    ' Do nothing since this item is being actioned by the User  
  End If
```

As per actioning any other type of Workflow Item, the 'AfterItemAction' event will also be fired (providing an outcome has been specified in the 'BeforeItemAction' event).

NOTE: If the Script does not return an outcome by doing `otherParameters.SetString("Outcome", "My Outcome Value")` then the Question or Item Group will not be completed but can still be answered manually by the User.

Script Events

Workflow Types support many Script events which are described in this section.

The following events are new to Version 2 type Workflows:

- AfterItemGroupAdd
- GetTrainStops
 - This event is also available to Version 1 type Workflows.

AfterInitialise

This event is called when the Workflow is first initialised from the Workflow Type but has not yet been saved to the database.

NOTE: This event CAN access the Workflow Executor object.

Common uses are:

- Prevent the workflow from being created (e.g., the Account is in a state that does not make sense to this Workflow, e.g.):
 - `Main = False`
`finBL.Error.ErrorBegin("Workflow cannot be used for Deposits.")`
- Add Items, e.g.:
 - `Workflow.GetCurrentItemGroup().Items.AddDecisionCard.AddDecisionCard()`
- Add Item Groups
 - `Workflow.ItemGroups.AddItemGroup("Description", "ItemId")`
 - or
`WorkflowItem = Workflow.ItemGroups.CreateWorkflowItem()`
`Workflow.ItemGroups.Insert(0, WorkflowItem)`
- Skip Items, e.g.:
 - `WorkflowExecutor.ItemSetStatusSkipped`
- Set Items to Not Applicable, e.g.:
 - `WorkflowExecutor.ItemSetStatusNotApplicable`

NEVER do the following from this event:

- Save the Workflow
- Refresh the Workflow

NOTE: Setting the Workflow Type's 'Initialisation' property to 'None' on the 'New Workflows' page allows the Script to completely handle the creation of Item Groups and Items.

AfterItemGroupAdd

This event is called when an Item Group is added to the Workflow.

NOTE: This event CAN access the Workflow Executor object.

This event is called for each Item Group that is added, e.g., if an Item Group is set to repeat for each Account Client, the event will be called for each group.

NOTE: During initialisation, this event will be called once for each Item Group added to the Workflow (based on the Initialisation option on the 'New Workflows' page of the Workflow Types form) and will occur before the AfterInitialise event.

The `workflowItem` event parameter is a reference to the newly added Item Group.

Common uses are:

- Prevent the User from manually adding an Item Group (from the Workflows form, Items page), e.g.:
 - `otherParameters.GetBoolean("UserInvoked") = True`
- Update the Item Group's description
- Automatically complete items, e.g.:
 - `WorkflowExecutor.CompleteItem`
 - `WorkflowExecutor.CompleteItemById` (see warning below)
 - `WorkflowExecutor.CompleteItemByOriginalItemId` (see warning below)
- Skip Items, e.g.:
 - `WorkflowExecutor.ItemSetStatusSkipped`
 - `WorkflowExecutor.ItemSetStatusSkippedById` (see warning below)
 - `WorkflowExecutor.ItemSetStatusSkippedByOriginalItemId` (see warning below)
- Set Items to Not Applicable, e.g.:
 - `WorkflowExecutor.ItemSetStatusNotApplicable`
 - `WorkflowExecutor.ItemSetStatusNotApplicableById` (see warning below)
 - `WorkflowExecutor.ItemSetStatusNotApplicableByOriginalItemId` (see warning below)

WARNING: Do not assume that the Workflow's Current Item Group is the Item Group that has just been added, e.g., never use the 'ByItemId' or 'ByOriginalItemId' helper methods unless you are absolutely sure that the Item Group being added is now the Current Item Group.

Use the '[AfterItemGroupBegin](#)' event if you need to ensure that the various helper method that work on the current Item Group will work correctly.

If you are unsure that the new Item Group is going to be the Current Item Group, instead of referencing Workflow Items via the Workflow Executor's helper methods, e.g.:

```
' Handle Events
Select Case eventId
  Case "AfterItemGroupAdd"
    Main = workflowExecutor.ItemSetStatusNotApplicableByOriginalItemId("TEST", True)
End Select
```

Use the Items collection of the new Item Group, e.g.:

```

' Handle Events
Select Case eventId
  Case "AfterItemGroupAdd"
    Main =
workflowExecutor.ItemSetStatusNotApplicable(workflowItem.Items.ItemByOriginalId("TEST"), True)
End Select

```

NEVER do the following from this event:

- Save the Workflow
- Refresh the Workflow

This example updates the Item Group's description to the Applicant's name (assuming the Item Group is set to repeat per Applicant):

```

Dim AccountAppApplicant As finAccountAppApplicant
Dim SourceObject As Object

' Assume Success
Main = True

' Get Workflow Executor
mWorkflowExecutor = DirectCast(otherParameters.GetObject("WorkflowExecutor"), finWorkflowExecutor)

' Handle Events
Select Case eventId
  Case "AfterItemGroupAdd"
    ' Get Workflow Item's Applicant
    Main = workflowItem.GetSourceObject(SourceObject)

    ' Update Description
    If Main Then
      AccountAppApplicant = DirectCast(SourceObject, finAccountAppApplicant)
      workflowItem.Description = "Group for Applicant " & AccountAppApplicant.Name
    End If
  End Select

```

NOTE: The above can also be achieved using the [ApplicantName] tag in the Item Group's description.

BeforeProcess

This event is called before the Workflow is processed (e.g., from the Workflows form or from the Account Processes wizard).

NOTE: This event CAN access the Workflow Executor object.

Common uses are:

- Prevent the Workflow from being processed, e.g.:
 - The User does not have permission to process the Workflow.
- Inform finPOWER Connect that processing has taken place (set the `eventHandled` parameter to `True`). This allows:
 - The Script to handle the entire Workflow process, e.g., a Workflow could be defined that contains no Workflow Items but simply uses a Script to take it through various different states which could be stored in the Workflow's `UserData`.

NEVER do the following from this event:

- Save the Workflow
- Refresh the Workflow

AfterItemGroupBegin

This event is called once, when an Item Group is first started, e.g., the Workflow is processed or a Workflow Item is being actioned and the current Item Group has not yet had this event called.

The `workflowItem` event parameter is a reference to Workflow's Current Item Group.

NOTE: Using this event instead of the '[AfterItemGroupAdd](#)' event ensures that any of the Workflow Executor's helper methods that act on the current Item Group (e.g., `CompleteItemById`) will work as expected.

Unlike the '[AfterItemGroupAdd](#)' event, this event is run at the point in time at which this Item Group becomes the current Item Group; the '[AfterItemGroupAdd](#)' event occurs as soon as the Item Group is added to the Workflow which, depending on how the Workflow Type is configured, may be when the Workflow is first created.

Common uses are:

- Prevent the User from manually adding an Item Group (from the Workflows form, Items page), e.g.:
 - `otherParameters.GetBoolean("UserInvoked") = True`
- Update the Item Group's description
- Automatically complete items, e.g.:
 - `WorkflowExecutor.CompleteItem`
 - `WorkflowExecutor.CompleteItemByItemId`
 - `WorkflowExecutor.CompleteItemByOriginalItemId`
- Skip Items, e.g.:
 - `WorkflowExecutor.ItemSetStatusSkipped`
 - `WorkflowExecutor.ItemSetStatusSkippedByItemId`
 - `WorkflowExecutor.ItemSetStatusSkippedByOriginalItemId`
- Set Items to Not Applicable, e.g.:
 - `WorkflowExecutor.ItemSetStatusNotApplicable`
 - `WorkflowExecutor.ItemSetStatusNotApplicableByItemId`
 - `WorkflowExecutor.ItemSetStatusNotApplicableByOriginalItemId`

NEVER do the following from this event:

- Save the Workflow
- Refresh the Workflow

This example sets a Workflow Item in the Item Group to 'Not Applicable':

```
Dim AccountAppApplicant As finAccountAppApplicant
Dim SourceObject As Object

' Assume Success
Main = True

' Get Workflow Executor
mWorkflowExecutor = DirectCast(otherParameters.GetObject("WorkflowExecutor"), finWorkflowExecutor)

' Handle Events
Select Case eventId
    Case "AfterItemGroupBegin"
        ' Make a WorkflowItem 'Not Applicable'
        Main = workflowExecutor.ItemSetStatusNotApplicableByOriginalItemId("TEST", True)
End Select
```

NOTE: The above can also be achieved using the `[ApplicantName]` tag in the Item Group's description.

BeforeItemAction

This event is called before actioning an item.

NOTE: This event CAN access the Workflow Executor object.

Common uses are:

- Prevent an item from being actioned, e.g.:
 - The User does not have permission.
 - A previous item needs actioning before this item can be actioned.
- Perform actions that could fail (and should therefore fail the item before its status has been set), e.g., adding a Transaction to an Account.
- Update the Outcome or set the Status Notes for the item being actioned.
 - As of Version 2, this includes automatic 'Question' and 'Item Group' type items.
- Perform calls to external Web Services, e.g., to perform a Credit Enquiry.

NEVER do the following from this event:

- Save the Workflow
- Refresh the Workflow

This event can be used to automatically set the outcome of a 'Question' type Workflow Item (as of Version 2, 'Question' type items can be automatic):

```
Case "BeforeItemAction"
  Outcome = otherParameters.GetString("Outcome")
  StatusNotes = otherParameters.GetString("StatusNotes")

  If workflowItem.Automatic AndAlso workflowItem.ItemIdOriginal = "OVER20" AndAlso
    Len(Outcome) = 0 Then
    ' Automatically determine if Applicant is over 20
    If workflowItem.SourceObjectType = isefinWorkflowItemSourceObjectType.AccountAppApplicant Then
      AccountAppApplicant = workflow.AccountApp.Applicants.ItemByPk(workflowItem.SourceObjectPk)
      If AccountAppApplicant.Age >= 20 Then
        otherParameters.SetString("Outcome", "Yes")
      Else
        otherParameters.SetString("Outcome", "No")
        otherParameters.SetString("StatusNotes", "Applicant is " & AccountAppApplicant.Age)
      End If
    Else
      ' Item not configured to repeat per Applicant
    End If
  End If
```

NOTE: The check `Len(Outcome) = 0` is applied since the User might be manually actioning this item in which case the outcome will have already been specified.

The table below notes points of interest for different item types when using this event:

Item Type	Details
None	

WARNING: Do not complete the current item group from this event (e.g., using `finWorkflowExecutor.CompleteCurrentItemGroup`) since it will cause the current item to be skipped.

CouldActionItem

This event applies only to **Send Document** type Workflow items.

NOTE: This event CANNOT access the Workflow Executor object.

This event is called before attempting to action a 'Send Document' type Item. Common uses are:

- Check that all required recipients are able to receive the document, e.g., they have a postal address recorded.
 - Do `otherParameters.SetBoolean("CouldAction", False)` to indicate this item could not be processed. You can also record a reason
`otherParameters.SetString("ActionNotes", "Reason could not action")`.

NEVER do the following from this event:

- Save the Workflow
- Refresh the Workflow
- Update the Workflow in any way (this event is just an enquiry and should therefore not cause any changes to occur).

AfterItemAction

This event is called after actioning a Workflow Item.

NOTE: This event CAN access the Workflow Executor object.

Common uses are:

- Complete other, related Workflow Items.
 - E.g., using the `CompleteItem` or `AutoActionItem` methods.
- Skip or set the status of items to 'Not Applicable' based on the Outcome of the item being actioned.
- Add another Item Group based on the Outcome of the item being actioned.
- If the Workflow Item that was actioned was an Item Group then:
 - Configure the new current Item Group (`WorkflowExecutor.CurrentItemGroup`), e.g., set the status of items to 'Not Applicable'.
 - Add a new Item Group (`WorkflowExecutor.AddItemGroupFromWorkflowType`).
 - Dynamically create and populate a new Item Group.

NEVER do the following from this event:

- Save the Workflow
- Refresh the Workflow

WARNING: At the point at which this event is called, the Workflow's 'Current' Item will be the item after the Item that has just been actioned.

The table below notes points of interest for different item types when using this event:

Item Type	Details
Decision Card	<p>The following special 'Other Parameters' are available:</p> <ul style="list-style-type: none">• DecisionOutcomeLog<ul style="list-style-type: none">◦ The log object recording the Decision Outcome, e.g., a <code>finAccountLog</code> object.◦ The Script may wish to update the log's automatically generated subject, e.g.: <pre>Case "AfterItemAction" If workflowItem.ItemIdOriginal = "DC1" Then With DirectCast (otherParameters.GetObject ("DecisionOutcomeLog"), finAccountAppLog) .Subject = "A custom decision outcome" Main = .Save() End With End If</pre>
Log Outgoing Communication	<p>The following special 'Other Parameters' are available:</p> <ul style="list-style-type: none">• Log<ul style="list-style-type: none">◦ The log that was created, e.g., a <code>finAccountLog</code> object.◦ The Script may wish to update the log's automatically generated subject, e.g.:

```
Case "AfterItemAction"  
  If workflowItem.ItemIdOriginal = "LOG" Then  
    With DirectCast(otherParameters.GetObject("Log"),  
finAccountAppLog)  
      .Subject = "A custom decision outcome"  
      Main = .Save()  
    End With  
  End If
```

BeforeItemReset

This event is called before resetting a Workflow Item.

NOTE: This event CAN access the Workflow Executor object.

Common uses are:

- Prevent the Workflow Item from being reset.

NEVER do the following from this event:

- Save the Workflow
- Refresh the Workflow

AfterItemReset

This event is called before resetting a Workflow Item.

NOTE: This event CAN access the Workflow Executor object.

Common uses are:

- Prevent the Workflow Item from being reset.
 - Typically, the [BeforeItemReset](#) event would be used.
- Undo certain actions that are related to the Workflow Item, e.g.:
 - Delete a Log (or change its Notes) that was created by the Script upon first actioning this Workflow Item.

NEVER do the following from this event:

- Save the Workflow
- Refresh the Workflow

NOTE: Any unpublished Document Logs attached to a Workflow Item that is reset will automatically have their Publish Status set to 'Not Issued'.

BeforeClose

This event is called before closing a Workflow.

NOTE: This event CANNOT access the Workflow Executor object.

Common uses are:

- Prevent the Workflow from being closed.
 - Ensure that only a User with a certain permission can close the Workflow.
- Undo certain actions that are related to the Workflow if it is being cancelled, e.g.:
 - Delete a Log (or change its Notes) that was created by the Script.

NEVER do the following from this event:

- Save the Workflow
- Refresh the Workflow

CanActionItem

This event is called to see if the current User is allowed to action a Workflow Item.

NOTE: This event CANNOT access the Workflow Executor object.

NEVER do the following from this event:

- Save the Workflow
- Refresh the Workflow
- Update the Workflow in any way (this event is just an enquiry and should therefore not cause any changes to occur.

NOTE: The `finWorkflowItem` object has a `CanAction` method. This method performs various checks and this Script event is only called providing all of the built-in checks (e.g., permission keys and the item's state) are passed.

The following example prevents an item from being actioned:

```
Case "CanActionItem"
  If workflowItem.ItemIdOriginal = "DC" Then
    If finBL.CurrentUser.UserId <> "XXX" Then
      otherParameters.SetBoolean("CanAction", False)
      otherParameters.SetString("Message", "Only User 'XXX' can action this item.")
    End If
  End If
```

The following example updates the message shown due to an item not being able to be actioned, regardless of whether it could be actioned or not:

```
Case "CanActionItem"
  If workflowItem.ItemIdOriginal = "DC" Then
    If workflowItem.DecisionCardPk = 0 Then
      otherParameters.SetString("Message", "Your manager must allocate a Decision Card.")
    End If
  End If
```

BeforeAccountDecline

This event is called for **Account** type Workflows only and is called when a Refinanced Account is being declined.

NOTE: This event CANNOT access the Workflow Executor object.

This is used instead of the 'BeforeClose' Workflow event and common uses are:

- Prevent the Workflow from being closed and therefore the Refinanced Account from being declined.
 - It may be that the Workflow should be completed (or cancelled) manually by the User before attempting to decline the Refinanced Account.

NEVER do the following from this event:

- Save the Workflow
- Refresh the Workflow

UpdateDocumentsList

This event is called to update the list of Documents displayed to the User in the Action Workflows wizard.

NOTE: This event CANNOT access the Workflow Executor object.

NEVER do the following from this event:

- Save the Workflow
- Refresh the Workflow

GetTrainStops

This event is called to update the list of Train Stops generated when calling the `finWorkflow.GetTrainStops` method.

NOTE: This event CANNOT access the Workflow Executor object.

NEVER do the following from this event:

- Save the Workflow
- Refresh the Workflow

This event can access the Train Stops and either update existing items, add new ones or clear the existing collection and generate its own items.

The following example generates a completely custom list of Train Stops (unrelated to any Workflow Item Groups) and flags 'Stop B' as the current Train Stop. It also sets the Application Shortcut to run when 'Stop C' is clicked:

```
Case "GetTrainStops"
    Dim WorkflowTrainStops As finWorkflowTrainStops

    ' Get Train Stops
    WorkflowTrainStops = DirectCast(otherParameters.GetObject("TrainStops"), finWorkflowTrainStops)

    ' Overwrite with custom Train Stops
    With WorkflowTrainStops
        .Clear()
        .AddCustom("Stop A", False)
        .AddCustom("Stop B", True)
        .AddCustom("Stop C", False,
            "app://FormShow?form=Workflows&id=" & finBL.UrlEncode(workflow.WorkflowId))
    End With
```

The next example appends an Actions List to a Train Stop if it is represented by Item Group 'TEST'. In this example, the Actions are shortcuts to setting the outcome of an Item with the code 'TS' which exists within the Item Group:

```
Case "GetTrainStops"
    Dim ApplicationShortcutUrl As String
    Dim WorkflowItemGroup As finWorkflowItem
    Dim WorkflowItemTS As finWorkflowItem
    Dim WorkflowItemOutcomeItem As finWorkflowItemOutcomeItem
    Dim WorkflowTrainStop As finWorkflowTrainStop
    Dim WorkflowTrainStops As finWorkflowTrainStops

    ' Get Train Stops
    WorkflowTrainStops = DirectCast(otherParameters.GetObject("TrainStops"), finWorkflowTrainStops)

    ' Add custom Actions
    If workflow.Status = iseFinWorkflowStatus.Open Then
        For Each WorkflowTrainStop In WorkflowTrainStops
            ' Get Item Group
            WorkflowItemGroup = WorkflowTrainStop.WorkflowItemGroup

            ' Add Actions (list outcomes from Item 'TS')
            If WorkflowItemGroup.IsCurrentItemGroup() Then
                WorkflowItemTS = WorkflowItemGroup.Items.ItemByOriginalId("TS")

                If WorkflowItemTS IsNot Nothing AndAlso
                    WorkflowItemTS.Status = iseFinWorkflowItemStatus.NotStarted Then
                    With WorkflowTrainStop.ActionsMenu.MenuItems
                        ' Clear existing Actions (just in case)
                        .Clear()

                        ' Add one Action per Outcome
                        For Each WorkflowItemOutcomeItem In WorkflowItemTS.OutcomeItems
                            ' Create Action URL
                            ApplicationShortcutUrl =
                                String.Format("app://FormAction?action=WorkflowItemSetOutcome&workflow={0}&itemPk={1}&outcome={2}"
                                    , workflow.Pk, WorkflowItemTS.Pk, finBL.UrlEncode(WorkflowItemOutcomeItem.Outcome))
```

```

        ' Add
        .AddItem("", WorkflowItemOutcomeItem.OutcomeDescriptionResolved, "",
                ApplicationShortcutUrl)
    Next
End With
End If
End If
Next
End If

```

The above example loops through the Outcomes defined in Workflow Item 'TS' (providing it has not been actioned) and adds corresponding Train Stop Actions; one per Outcome. This can be achieved more simply using one of the helper methods such as `ActionsFromWorkflowItemOutcomesByItemId`, e.g.:

```

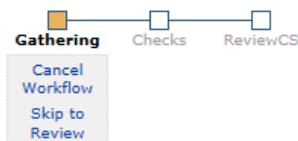
Case "GetTrainStops"
Dim WorkflowTrainStop As finWorkflowTrainStop
Dim WorkflowTrainStops As finWorkflowTrainStops

' Get Train Stops
WorkflowTrainStops = DirectCast(otherParameters.GetObject("TrainStops"), finWorkflowTrainStops)

' Add custom Actions
If workflow.Status = isefinWorkflowStatus.Open Then
For Each WorkflowTrainStop In WorkflowTrainStops
' Add Actions (list outcomes from Item 'TS')
If WorkflowTrainStop.WorkflowItemGroup.IsCurrentItemGroup() Then
WorkflowTrainStop.ActionsMenuFromWorkflowItemOutcomesByOriginalItemId("TS")
End If
Next
End If

```

This will create a Train Stops diagram something like this:



NOTE: In the above example, if you only wanted to include certain outcomes from item 'TS', you can specify a comma-separated list of outcomes, e.g., if the item contains three outcomes ('Yes', 'No', 'Maybe') but you only want to include 'Yes' and 'No':

```
WorkflowTrainStop.ActionsMenuFromWorkflowItemOutcomesByOriginalItemId("TS", "Yes,No")
```

You can also override the overall bullet size and shape (each Train Stop can also define a bullet shape so the diagram can have mixed bullet shapes), e.g.:

```

Case "GetTrainStops"
Dim WorkflowTrainStops As finWorkflowTrainStops

' Get Train Stops
WorkflowTrainStops = DirectCast(otherParameters.GetObject("TrainStops"), finWorkflowTrainStops)

' Customise
WorkflowTrainStops.BulletShape = isefinWorkflowTrainStopBulletShape.Circle
WorkflowTrainStops.BulletSize = 21

```

This will create a Train Stops diagram something like this:



GetWorkflowItemUserDataSummary

This event is called to allow a Workflow Item to produce a customised summary of its User Data.

NEVER do the following from this event:

- Save the Workflow
- Refresh the Workflow

The `finWorkflowItem` object has a `GetUserDataAsSummaryTables` method. This is called from the built-in Workflow Summary page for any Workflow Items with a status of either 'Open' or 'Complete'.

By default, this function will only return a summary if the Workflow Item is configured to have a 'Notes Entry' method of either 'Parameters (User Defined Workflow)' or 'Parameters (User Defined Workflow)' since it can use the defined Parameters to produce a summary.

Using the Script event, you can produce a customised summary, typically of the Workflow Item's User Data but the summary can include any information.

NOTE: Customised summaries can only be generated as a `ISSummaryTables` object, not as directly generated HTML.

The following example generates a customised summary for a Workflow Item:

```
Case "GetWorkflowItemUserDataSummary"
  Dim SummaryTable As ISSummaryTable
  Dim SummaryTables As ISSummaryTables

  Select Case workflowItem.ItemIdOriginal
    Case "CHK1"
      ' Initialise
      SummaryTables = finBL.CreateSummaryTables()
      SummaryTable = finBL.CreateSummaryTable()

      ' Build Summary
      With SummaryTable
        .TableClass = iseSummaryTableClass.Information
        .Columns.AddText(16).NoBreakChars = 16
        .Columns.AddText()

        With .Rows
          .AddSectionHeading2("My Custom Script Summary")
          .AddCaptionText("Custom Value:", workflowItem.UserData.GetString("CustomValue",
"[N/A]"))
        End With

        .TextAfter = "{{info|Summary generated from Workflow Type Script.}}"
      End With
      SummaryTables.Add("Main", SummaryTable)

      ' Handled
      otherParameters.SetObject("SummaryTables", SummaryTables)
      eventHandled = True
    End Select
```

NOTE: The Script must set `eventHandled` to `True` and add a 'SummaryTables' entry into the `otherParameters`.

The `KeyValueListToSummaryTables` helper method can be used if you simply want to display all of the Workflow Item's User Data, e.g.:

```
Select Case workflowItem.ItemIdOriginal
  Case "CHK1"
    ' Handled
    otherParameters.SetObject("SummaryTables",
```

```
eventHandled = True      finBL.Utilities.KeyValueListToSummaryTables(workflowItem.UserData)
End Select
```

GetWorkflowUserDataSummary

This event is called to allow a Workflow to produce a customised summary of its User Data.

NOTE: This event CANNOT access the Workflow Executor object.

This event is identical to the '[GetWorkflowItemUserDataSummary](#)' but targets the Workflow instead of a Workflow Item.

The following example generates a customised summary for a Workflow:

```
Case "GetWorkflowUserDataSummary"
' Get a User Data summary for Workflow
Dim SummaryTable As ISSummaryTable
Dim SummaryTables As ISSummaryTables

' Initialise
SummaryTables = finBL.CreateSummaryTables()
SummaryTable = finBL.CreateSummaryTable()

' Build Summary
With SummaryTable
    .TableClass = iseSummaryTableClass.Information
    .Columns.AddText(16).NoBreakChars = 16
    .Columns.AddText()

    With .Rows
        .AddSectionHeading2("My Custom Script Summary")
        .AddCaptionText("Custom Value:", workflow.UserData.GetString("CustomValue", "[not set]"))
    End With

    .TextAfter = "{{info|Summary generated from Workflow Type Script.}}"
End With
SummaryTables.Add("Main", SummaryTable)

' Handled
otherParameters.SetObject("SummaryTables", SummaryTables)
eventHandled = True
```

WARNING: By default, the built-in Workflow Summary page will not show Workflow User Data. The 'ShowUserDefined' constant must be set to True.

Script Responsibilities

For execution of a Workflow to work correctly, a Workflow Type Script has certain responsibilities. These are outlined in this section.

Recording Logs

Any Logs created as a result of processing a Workflow or actioning a Workflow Item should be recorded.

This Logs are then available for inspection (e.g., to decide what to publish) once processing is complete.

The `finWorkflowExecutor` object has the following methods to do this:

- `CreatedLogsAddAccount`
- `CreatedLogsAddAccountApp`
- `CreatedLogsAddClient`

The collection of Logs created is available via the `finWorkflowExecutor.CreatedLogs` collection.

NOTE: The `finWorkflowExecutorCreatedLog` has a `Log` property that is an Object. The object type varies depending on the `LogClass` property, e.g., it may be a `finAccountLog` or a `finClientLog`.

Recording New Workflows

New Workflows can be created automatically by actioning Workflow Items (Items that support the concept of 'Outcomes' all new Workflows to be created).

Any Workflows created as a result of Script code should also be recorded. The `finWorkflowExecutor` object has the following method to do this:

- `CreatedWorkflowsAdd`

The new Workflows are available via the `finWorkflowExecutor.CreatedWorkflows` collection.

When adding a Workflow to this collection, a `toProcess` flag can be specified. If this is `True` then at the end of processing the current Workflow, any Workflows in this collection that are flagged to be processed will also be processed.

Executing External Web Services

Calls to external Web Services, e.g., to perform a Veda Credit Enquiry, cannot be executed within a database transaction for the following reasons:

- If the transaction is rolled back, any auditing information recorded for the Web Service call will be lost.
- Web Service calls can sometimes be slow. Executing one within a transaction may result in locking occurring on the database and affect performance for other Users.

Therefore, if a Web Service type Workflow Item is actioned, the following takes place:

- A check will be made to ensure that no "outer" database transaction exists and if it does:
 - If the User is directly actioning this Workflow Item via the `finWorkflowExecutor.ExecuteWorkflowItemPerformAction` method, this method will fail and return an appropriate error message.
 - If this Workflow Item is being actioned automatically, e.g., because the Workflow is being processed of a prior Workflow Item has been actioned, no error will occur but processing

will be halted and the `finWorkflowExecutor.ExecuteWorkflowItemPerformAction` property will be set to `True`.

- The current database transaction will be committed.
- The external Web Service call will be made.
- A new database transaction will be started.
 - The new transaction is started regardless of whether the Web Service call was successful or not.

Any Workflow Type Script code wishing to perform a call to an external Web Service should:

- Perform the call in the 'BeforeItemAction' event.
- Check that a Web Service call can be made.
- Use the special `WebServiceCallBegin` and `WebServiceCallEnd` methods of the Workflow Executor either side of the Web Service call.

The following, stripped down, code sample shows the above in action and is taken from the [Workflow Type Script Examples](#) section:

```
Dim Client As finClient
Dim CreditEnquiryRequest As ISCreditEnquiryRequest
Dim CreditEnquiryRequestVeda As ISCreditEnquiryRequest VedaXmlNZ IndividualEnquiry
Dim CreditEnquiryResponse As ISCreditEnquiryResponse
Dim CreditEnquiryResponseVeda As ISCreditEnquiryResponse_VedaXmlNZ_Individual

' Assume Success
Main = True

' Get Workflow Executor
mWorkflowExecutor = DirectCast(otherParameters.GetObject("WorkflowExecutor"), finWorkflowExecutor)

' Handle Events
Select Case eventId
Case "BeforeItemAction"
    If workflowItem.ItemIdOriginal = "CE" Then
        ' Validate
        If Not mWorkflowExecutor.CanPerformWebServiceCall() Then
            Main = False
            finBL.Error.ErrorBegin("Cannot perform Web Service call.")
        End If

        ' Execute Request
        If Main Then
            mWorkflowExecutor.WebServiceCallBegin()
            Main = finBL.CreditBureau.ExecuteCreditEnquiry(CreditEnquiryRequestVeda,
                CreditEnquiryResponse, Nothing, True,
                workflow.Pk, workflowItem.pk)
            mWorkflowExecutor.WebServiceCallEnd()
        End If
    End Select
```

WARNING: The `WebServiceCallBegin` method will commit the current database transaction. It is the Script's responsibility to ensure it is written in such a way that this does not cause any issues.

The [Database Transactions](#) section discusses database transactions in more detail.

Database Transactions

Each Workflow Item is actioned within its own database transaction.

This means that if actioning of the item fails, the database transaction will be rolled back, thereby undoing any database changes made.

If a database transaction has already been started by another process (e.g., the `finAccountPayArrangementAdd` object or a Script), the Workflow will be executed within an "outer" database transaction meaning:

- If Workflow Items are actioned and then the outer database transaction is rolled back, any changes made to the Workflow will be lost.
- Any Workflow Items (or code within the Workflow Type Script) will be unable to make external Web Service calls (e.g., to the New Zealand PPSR G2B) since these are not permitted within an outer database transaction.
 - This is discussed in the [Script Responsibilities, Executing External Web Services](#) section.

NOTE: When running from the finPOWER Connect User Interface, Workflows are not processed within an outer database transaction.

However, if a Workflow is processed as a result of adding an Account Payment Arrangement (via the `finAccountPayArrangementAdd` object), the execution of the Workflow will occur within an outer database transaction.

If functionality within a Workflow Type Script requires that there is no outer transaction (e.g., the Script wants to perform a Veda Credit Enquiry), the Script can check and fail actioning of the item in the following way:

```
' Handle Events
Select Case eventId
Case "BeforeItemAction"
  If workflowItem.ItemIdOriginal = "CE" Then
    ' Validate
    If Not mWorkflowExecutor.CanPerformWebServiceCall() Then
      Main = False
      finBL.Error.ErrorBegin("Cannot perform Web Service call.")
    End If
  End If
End Select
```

Workflow Type Script Examples

Script samples are available in the [Script Events](#) section.

This section gives more task-specific examples.

Set Workflow Items to 'Not Applicable' When Initialising

This sample does the following when the Workflow is first initialised:

- Sets the 'PPSRSEARCH' Item in the current (i.e., first in this case) Item Group to 'Not Applicable'.

```
Case "AfterInitialise"  
  If Main Then Main = mWorkflowExecutor.ItemSetStatusNotApplicableById("PPSRSEARCH"), True)
```

Note the following:

- Helper methods allow the 'Not Applicable' state of an Item to be setting using one of the following methods:
 - `ItemSetStatusNotApplicable`
 - `ItemSetStatusNotApplicableById`
 - `ItemSetStatusNotApplicableByOriginalItemId`

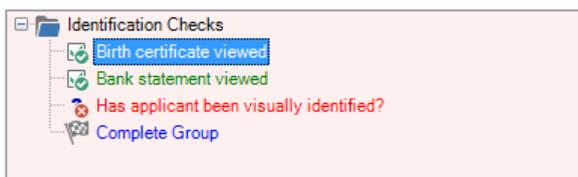
Create New Item Group with Items from Script

This sample does the following when the Workflow is first initialised (although the code could just as easily be used in the 'AfterItemAction' event):

- Adds a new Item Group
 - Can be completed non-sequentially.
 - Has two Outcomes.
 - Will automatically complete when all items have been actioned.
 - ✦ This is achieved by using the special Outcomes:
 - [All Success]
 - [Any Fail]
- Adds several Check List type Items to this Item Group
- Adds a Question type Item.

```
Case "AfterInitialise"  
  ' Add new Item Group and Items  
  With Workflow.ItemGroups.AddItemGroup("Identification Checks", "", False)  
    ' Add Outcomes  
    With .OutcomeItems  
      .AddNone("[All Success]", True, "Everything is fine")  
      .AddAllocateToSupervisor("[Any Fail]", True, "There are issues")  
    End With  
  
    ' Add Workflow Items  
    With .Items  
      ' Check List Items  
      .AddCheckListItem("Birth certificate viewed", "")  
      .AddCheckListItem("Bank statement viewed", "")  
  
      ' Question  
      With .AddQuestion("Has applicant been visually identified?", "")  
        With .OutcomeItems  
          .AddNone("Yes", True)  
          .AddNone("No", False)  
        End With  
      End With  
    End With  
  End With  
End With
```

Once added and run, a new Workflow will look something like this:



NOTE: This sample makes extensive use of helper methods added in finPOWER Connect version 2.02.04, e.g., to add Workflow Items and Outcome Items to the Item Group. These helper methods are not limited to Version 2 type Workflows.

Skip Items in Group and Complete Group

This sample does the following when the 'Test' Workflow Item is actioned:

- Skips all incomplete items in the Item Group.
- Completes the Item Group with an outcome of 'Good'.

```
Case "AfterItemAction"  
  Dim WorkflowItem2 As finWorkflowItem  
  
  If workflowItem.ItemIdOriginal = "TEST" Then  
    ' Skip items (not really required since completing the Item Group will do this anyway)  
    For Each WorkflowItem2 In workflowItem.ParentItemGroup.Items  
      If WorkflowItem2.Status = iseFinWorkflowItemStatus.NotStarted Then  
        If Not mWorkflowExecutor.ItemSetStatusSkipped(WorkflowItem2, True) Then  
          Main = False  
          Exit For  
        End If  
      End If  
    Next  
  
    ' Complete Item Group  
    If Main Then  
      Main = mWorkflowExecutor.CompleteCurrentItemGroup("Good", "Completed from test Script.")  
    End If  
  End If
```

NOTE: In the above example, the code to skip remaining items is included for example. The code is not really required since the `CompleteCurrentItemGroup` method will skip any incomplete Workflow Items before completing the Item Group.

Skip Current Item Group and Add a new Item Group

This sample does the following when the 'Test' Workflow Item is actioned:

- Skips the current Item Group.
- Add a new Item Group 'WS'.

```
Case "AfterItemAction"  
  If workflowItem.ItemIdOriginal = "TEST" Then  
    ' Skip Current Item Group and add Item Group 'WS'  
    Main = mWorkflowExecutor.SkipCurrentItemGroup("WS")  
  
    ' Just skip Current Item Group  
    'Main = mWorkflowExecutor.SkipCurrentItemGroup()  
  
    ' Skip Current Item Group and repeat it (assumes Item Group has a code)  
    'Main = mWorkflowExecutor.SkipCurrentItemGroup(workflow.GetCurrentItemGroup().ItemIdOriginal)  
  End If
```

NOTE: The `SkipCurrentItemGroup` method can also be used to repeat the current Item Group by passing in the current Item Group's code as shown in the remarked out code above.

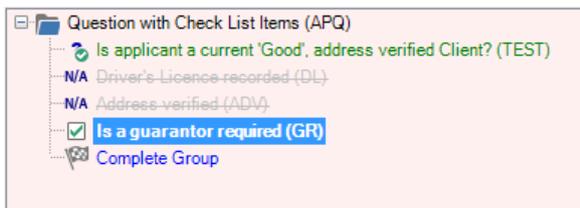
Set Check List Items Based Upon Question Outcome

This sample does the following when the 'Test' Workflow Item in Item Group 'APQ' is actioned (this Workflow Item is a 'Question' type Item with 'Yes' and 'No' outcomes):

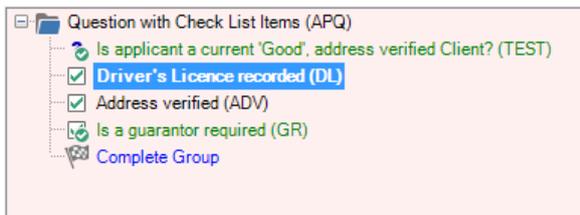
- Sets the states of various Check List Items based on whether the outcome was 'Yes' or 'No'.
 - Some Items will be completed, others set to 'Not Applicable' based on the selected outcome.
 - The 'AfterItemAction' event is used.

```
Case "AfterItemAction"  
  If mWorkflowExecutor.CurrentItemGroupId = "APQ" AndAlso workflowItem.ItemIdOriginal = "TEST"  
  Then  
    ' Get Outcome  
    Outcome = otherParameters.GetString("Outcome")  
  
    ' Set Check List Items based on Outcome  
    If Outcome = "Yes" Then  
      ' Address verified 'Good' Client  
      If Main Then Main = mWorkflowExecutor.ItemSetStatusNotApplicableByItemId("DL", True)  
      If Main Then Main = mWorkflowExecutor.ItemSetStatusNotApplicableByItemId("ADV", True)  
    Else  
      ' Other  
      If Main Then Main = mWorkflowExecutor.CompleteCheckListItemByItemId("GR", "Guarantor  
required", True)  
    End If  
  End If  
End If
```

Once an outcome of 'Yes' is selected, a new Workflow will look like this:



And an outcome of 'No' will look like this:



NOTE: The `CompleteCheckListItemByItemId` method completes Check List type Items but DOES NOT FIRE ANY SCRIPT EVENTS such as 'BeforeItemAction' and 'AfterItemAction'. This is by design.

Note the following:

- The `CurrentItemGroupId` property of the Workflow Executor can be used to check the Current Item Group.
 - This returns the `OriginalItemId` property rather than the `ItemId` since the `ItemId` may have a sequence number suffix if the Item Group has been added to the Workflow more than once.
- Helper methods allow the 'Not Applicable' state of an Item to be set using one of the following methods:

- ItemsetStatusNotApplicable
- ItemsetStatusNotApplicableByItemId
- ItemsetStatusNotApplicableByOriginalItemId
- **Check List Items can be completed using one of the following methods:**
 - CompleteCheckListItem
 - CompleteCheckListItemByItemId
 - CompleteCheckListItemByOriginalItemId

Creating Customised Documents

This sample does the following when the 'Test' Workflow Item is actioned:

- Creates the following Document Logs:
 - One for the Main Applicant.
 - One for the Account Application's Branch.
- The Account Application Advice (AAA) and Client Advice (CA) Word VBA documents are created.

```
Case "AfterItemAction"
If workflowItem.ItemIdOriginal = "TEST" Then
' Document to 'Main' Applicant
AccountAppApplicant = workflow.AccountApp.GetMainApplicant()
If AccountAppApplicant Is Nothing Then
Main = False
finBL.Error.ErrorBegin("Account Application does not have a 'Main' Applicant.")
Else
' Create Document Log (using helper method of Workflow Executor)
AccountAppLog = mWorkflowExecutor.CreateAccountAppDocumentLog("AAA",
True,
workflow.AccountAppPk,
workflowItem.pk,
AccountAppApplicant.Pk)

' Update
With AccountAppLog
.Notes = "Custom note details for Applicant."
End With

' Save
Main = AccountAppLog.Save()
End If

' Document to Branch
If Main Then
If workflow.AccountApp.BranchPk = 0 Then
Main = False
finBL.Error.ErrorBegin("Account Application does not have a Branch.")
Else
' Create Document Log (using helper method of Workflow Executor)
ClientLog = mWorkflowExecutor.CreateClientDocumentLog("CA",
True,
workflow.AccountApp.Branch.ClientPk,
workflowItem.pk)

' Update
With ClientLog
.Notes = "Custom note details for Branch."
End With

' Save
Main = ClientLog.Save()
End If
End If
End If
```

NOTE: Most of the above functionality can be achieved using a 'Send Document' type Workflow Item. However, in some cases, especially if you want to customise the Document Logs (or Email/ SMS content as per the next example) then using the Workflow Type Script is recommended.

Note the following:

- The `CreateAccountAppDocumentLog` and `CreateClientDocumentLog` helper methods of the `Workflow Executor` automatically record the created Logs to the `Workflow Executors CreatedLogs` collection providing the second parameter is `True`.

Creating Customised Email Documents

This sample does the following when the 'Test' Workflow Item is actioned:

- Creates the following Email Document Logs:
 - One for the Main Client.
 - One for the Account's Branch.
- Each of these Email has customised content and recipient lists.
- Client Email Advice (CE) documents are created.
 - Any default Subject or Message etc. defined on the Document are ignored and overwritten with custom details. The Document is just used as a placeholder for the finPOWER Connect publishing process.

```
Case "AfterItemAction"
If workflowItem.ItemIdOriginal = "TEST" Then
  ' Document to 'Main' Client
  ClientContactMethod = workflow.Account.Clients(0).Client.ContactMethods.GetCurrentEmail()
  If ClientContactMethod Is Nothing OrElse Not ClientContactMethod.HasValues() Then
    Main = False
    finBL.Error.ErrorBegin("Main Client does not have a current Email Address.")
  Else
    ' Create Document Log (using helper method of Workflow Executor)
    ClientLog = mWorkflowExecutor.CreateClientDocumentLog("CE",
                                                         True,
                                                         workflow.Account.Clients(0).ClientPk,
                                                         workflowItem.pk)

    ' Update
    With ClientLog
      ' General
      .Notes = "Custom note details for Client."

      ' Set Email details including CC address and automatic signature
      .ExtendedDataSetEmail(ClientContactMethod.Value,
                            "Test Email Subject",
                            "Test Email Message <b>HTML</b>",
                            "ph@mycompany.zzz",
                            "",
                            iseMessageTarget.OpenForEdit,
                            True,
                            "")

    End With

    ' Save
    Main = ClientLog.Save()
  End If

  ' Document to Branch
  If Main Then
    If workflow.Account.BranchPk = 0 Then
      Main = False
      finBL.Error.ErrorBegin("Client does not have a Branch.")
    ElseIf Len(workflow.Account.Branch.GetEmailForLetters()) = 0 Then
      Main = False
      finBL.Error.ErrorBegin("Branch does not have a current Email Address.")
    Else
      ' Create Document Log (using helper method of Workflow Executor)
      ClientLog = mWorkflowExecutor.CreateClientDocumentLog("CE",
                                                           True,
                                                           workflow.Account.Branch.ClientPk,
                                                           workflowItem.pk)

      ' Update
      With ClientLog
        ' General
        .Notes = "Custom note details for Branch."

        ' Set Email details including no signature
        .ExtendedDataSetEmail(workflow.Account.Branch.GetEmailForLetters(),
                              "Test Branch Email Subject",
                              "Test Branch Email Message <b>HTML</b>",
                              "",
                              "",
                              iseMessageTarget.OpenForEdit,
```

```
        True,  
        "")  
    End With  
    ' Save  
    Main = ClientLog.Save()  
End If  
End If  
End If
```

NOTE: Most of the above functionality can also be applied to SMS type Documents.

Note the following:

- Client Logs are created but there is no reason why an Account Document could not be used and Account Logs created instead.
- Email and SMS information is recorded against Logs in the `ExtendedData` property.
 - The `ExtendedDataSetEmail` method of the Logs is used to set this information.
 - ✦ The return value of this method is ignored since the only time it can fail is if you attempt to use it on an already-saved Log.
 - A similar `ExtendedDataSetSms` method exists for handling SMS Logs.
- A signature is automatically appended to the bottom of the Main Client's Email Message.
 - This is due to a signature of "*" being passed to the method.
 - This will include any Entity-specific signature providing the Log's `ClientId` property has been set prior to setting the extended data.
 - To exclude, pass a signature of "" to the method as the Branch Email does.

Creating Ad-Hoc Emails

Generally, Email and SMS messages are published via Documents (as per the previous couple of examples). However, it is possible to send them directly and record the details to a Log.

This sample does the following when the 'Test' Workflow Item is actioned:

- Sends an Email directly.
 - This assumes that User Preferences have SMTP settings configured.
 - You can use 'Open For Edit' providing the Workflow is being run from a User's PC via the finPOWER Connect User Interface.
 - ✦ This will not work for unattended execution, e.g., from a Web Service.
- Creates the following Email Logs (not Document Logs):
 - One for the Main Client.

```
Case "AfterItemAction"
  If workflowItem.ItemIdOriginal = "TEST" Then
    ' Document to 'Main' Client
    ClientContactMethod = workflow.Account.Clients(0).Client.ContactMethods.GetCurrentEmail()
    If ClientContactMethod Is Nothing OrElse Not ClientContactMethod.HasValues() Then
      Main = False
      finBL.Error.ErrorBegin("Main Client does not have a current Email Address.")
    End If

    ' Send Email
    If Main Then
      ' Initialise
      Recipients = ClientContactMethod.Value
      RecipientsCC = "ph@mycompany.zzz"
      EmailSubject = "Test Email Subject"
      EmailMessage = "Test Email Message <b>HTML</b>"

      ' Append Signature
      Signature = finBL.SettingsUser.EmailSignatureResolved(False, workflow.Account.BranchPk)
      If Len(Signature) <> 0 Then EmailMessage &= Signature

      ' Send
      Main = finBL.SendEmail(Recipients,
                            RecipientsCC,
                            "",
                            EmailSubject,
                            EmailMessage,
                            "",
                            True,
                            iseMessageTarget.Send)

    End If

    ' Create Log
    If Main Then
      ClientLog = mWorkflowExecutor.CreateClientLog("Email Sent",
                                                    True,
                                                    workflow.Account.Clients(0).ClientPk,
                                                    workflowItem.pk)

      ' Update
      With ClientLog
        ' Set Email details
        .ExtendedDataSetEmail(Recipients,
                               EmailSubject,
                               EmailMessage,
                               RecipientsCC,
                               "",
                               iseMessageTarget.Send,
                               True,
                               "")
      End With

      ' Save
      Main = ClientLog.Save()
    End If
  End If
```

NOTE: Most of the above functionality can also be applied to SMS type Documents.

Note the following:

- Client Logs are created.
 - These are NOT Document Logs, they are simply used to record the details of the Email that has already been sent.
 - Log creation could be omitted however, this is not recommended since it is a good way of auditing what is being sent.
- A signature is created via `finBL.SettingsUser.SignatureResolved`.
 - This allows an HTML or plain text signature to be created and optionally allows the primary key of a Branch to be specified so any Entity-specific is created.
 - The signature can be omitted if required.

WARNING: Because the Email is sent within a database transaction, any delays in sending the Email (e.g., communication problems with the SMTP server) may result in locks occurring on the database.

Automatic Client Credit Enquiry - Veda (New Zealand)

This example performs an automatic Credit Enquiry using the Veda (New Zealand) service for an **Account** type Workflow.

NOTE: This is unrelated to the 'Credit Enquiry' type Workflow Item (which simply shows the Credit Enquiry form to the User) and should not be used in conjunction with this type of item.

This sample does the following when the 'CE' Workflow Item is actioned:

- Uses the 'BeforeItemAction' event.
 - Since Web Service calls must first commit and then restart any database transactions, using any event other than 'BeforeItemAction' means that the Workflow Item will be completed, even if the Web Service call fails.
- Creates and executes a Veda NZ Request.
- Examines the Credit Enquiry Response and sets the Workflow Item's Outcome.
 - **NOTE:** The Outcome can be set to any value for 'None' type Workflow Items.

```
Dim Client As finClient
Dim CreditEnquiryRequest As ISCreditEnquiryRequest
Dim CreditEnquiryRequestVeda As ISCreditEnquiryRequest_VedaXmlNZ_IndividualEnquiry
Dim CreditEnquiryResponse As ISCreditEnquiryResponse
Dim CreditEnquiryResponseVeda As ISCreditEnquiryResponse_VedaXmlNZ_Individual

' Assume Success
Main = True

' Get Workflow Executor
mWorkflowExecutor = DirectCast(otherParameters.GetObject("WorkflowExecutor"), finWorkflowExecutor)

' Handle Events
Select Case eventId
Case "BeforeItemAction"
    If workflowItem.ItemIdOriginal = "CE" Then
        ' Validate
        If Not mWorkflowExecutor.CanPerformWebServiceCall() Then
            Main = False
            finBL.Error.ErrorBegin("Cannot perform Web Service call.")
        End If

        If Main Then
            ' Get Main Account Client
            Client = workflow.Account.Clients(0).Client

            ' Create Credit Enquiry Request
            CreditEnquiryRequest = finBL.CreditBureau.CreateCreditEnquiryRequest("VedaXMLNZ",
                                                                                "Consumer Enquiry")

            ' Cast to correct type
            CreditEnquiryRequestVeda = DirectCast(CreditEnquiryRequest,
                                                  ISCreditEnquiryRequest_VedaXmlNZ_IndividualEnquiry)

            ' Update from Client
            Main = finBL.CreditBureau.UpdateCreditEnquiryRequestFromClient(CreditEnquiryRequestVeda,
                                                                           Client)
        End If

        ' Update Request Options (with those defined under Global Settings)
        ' NOTE: This step is not necessary if all options are being set in the next block
        If Main Then
            Main = finBL.CreditBureau.UpdateCreditEnquiryRequestOptions(CreditEnquiryRequestVeda,
                                                                           Client)
        End If

        ' Set Options
        If Main Then
            With CreditEnquiryRequestVeda
                .AccessPurposeDescription = "Credit Decision"
                .AccountTypeDescription = "Term Account"
                .ApplicantTypeDescription = "Single Applicant"
            End With
        End If
    End Case
End Select
```

```

        .ConsentObtained = True

        '.IncludeCourtFinesData = True
    End With
End If

' Execute Request
' NOTE: Creates a Client Log and links this to this Workflow Item
If Main Then
    mWorkflowExecutor.WebServiceCallBegin()
    Main = finBL.CreditBureau.ExecuteCreditEnquiry(CreditEnquiryRequestVeda,
                                                    CreditEnquiryResponse, Nothing, True,
                                                    workflow.Pk, workflowItem.pk)

    mWorkflowExecutor.WebServiceCallEnd()
End If

' Examine the Response
If Main Then
    CreditEnquiryResponseVeda = DirectCast(CreditEnquiryResponse,
                                           ISCreditEnquiryResponse_VedaXmlNZ_Individual)

    ' Set Outcome
    If CreditEnquiryResponseVeda.BankruptcyCountUndischarged > 0 Then
        otherParameters.SetString("Outcome", "Fail")
        otherParameters.SetString("StatusNotes", String.Format("{0} undischarged bankruptcies",
                                                                CreditEnquiryResponseVeda.BankruptcyCountUndischarged))
    ElseIf CreditEnquiryResponseVeda.BankruptcyCount > 0 Then
        otherParameters.SetString("Outcome", "Refer")
        otherParameters.SetString("StatusNotes", String.Format("{0} bankruptcies",
                                                                CreditEnquiryResponseVeda.BankruptcyCount))
    Else
        otherParameters.SetString("Outcome", "Pass")
        otherParameters.SetString("StatusNotes", "No bankruptcies")
    End If
End If
End If
End Select

```

Note the following:

- Validates that a Web Service can be performed using the `WorkflowExecutor.CanPerformWebServiceCall` method.
 - NOTE:** This should only return `False` if, for some reason, this item is being called within a nested database transaction.
- When creating the `CreditEnquiryRequest` object, the Service (VedaXMLNZ) and Product (Consumer Enquiry) are as per the Credit Enquiry wizard, e.g.:

Select the Credit Bureau Service to use.

Service: VedaXMLNZ Veda (New Zealand)

Optionally select the Client you wish to enquire upon. ⓘ

Client:

Enquire on the Person Acting for this organisation?

Select the Credit Bureau Product to use.

Product: Consumer Enquiry

- The Service and Product determine the object type that we must use to perform the Credit Enquiry Request.
 - The code:

```

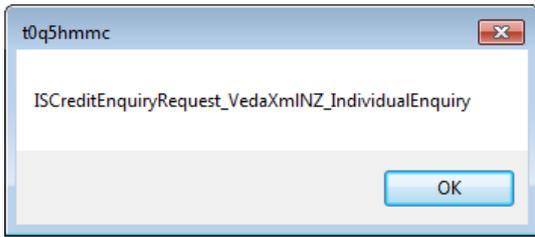
CreditEnquiryRequest = finBL.CreditBureau.CreateCreditEnquiryRequest("VedaXMLNZ",
                                                                    "Consumer Enquiry")

```

Always returns a generic `ISCreditEnquiryRequest` object.

- To determine the Service/ Product specific object type, we can use the `TypeName` function on the returned object, e.g.:

```
MsgBox (TypeName (CreditEnquiryRequest))
```



Pressing `Ctrl+C` will copy the message box text to the clipboard.

- We then cast this object to the correct type and assign it to our `CreditEnquiryRequestVeda` variable:

```
CreditEnquiryRequestVeda = DirectCast (CreditEnquiryRequest,
                                       ISCreditEnquiryRequest_VedaXmlNZ_IndividualEnquiry)
```

- Once the request has been updated from the Account's Main Client and the options updated from those defined under Global Settings, the service-specific options can be set.
 - These options vary based on the request object type.
 - The options can generally have self-explanatory names and can be determined from the Credit Enquiry wizard's 'Options' page, e.g.:

- The Request generally has 2 versions of each option, one of which is suffixed by 'Description'. We set these versions since the non-suffixed version generally requires a Credit Bureau-specific code, e.g.:

```
.AccessPurposeDescription = "Credit Decision"
```

- Since Web Service Requests cannot be executed inside a database transaction and actioning a Workflow Item occurs within a database transaction, we must use the `WebServiceCallBegin` and `WebServiceCallEnd` methods before and after executing the request.

WARNING: Failure to use these methods will result in either the Credit Enquiry request failing or a runtime exception occurring.

- Once the request has been executed, a Response object is returned.
 - Just like the request, this is a generic object. In this case an `ISCreditEnquiryResponse` object.
 - To get to service-specific properties, we must cast it to the correct type (remember, you can use the `TypeName` function to display the returned response type):

- Setting the Outcome and Status Notes is allowed for 'None' type Workflow Items, even though they do not define any Outcomes.

WARNING: The Script sample sets `.ConsentObtained = True` however, for legal reasons you may need to fail actioning of this item based on another Workflow Item, e.g., a Check List Item that determines whether the Client has given consent for a Credit Enquiry to be performed.

Automatic Client Credit Enquiry – GreenId

This example performs an automatic Credit Enquiry using the Edentiti GreenId service for an **Account** type Workflow.

The 'CE' Workflow Item is set to NOT repeat for each Client, therefore it will only be performed on the 'Main' Account Client.

See the [Automatic Client Credit Enquiry – Veda \(New Zealand\)](#) for more information.

```
Dim Client As finClient
Dim CreditEnquiryRequest As ISCreditEnquiryRequest
Dim CreditEnquiryRequestGreenId As ISCreditEnquiryRequest_GreenId_BackgroundCheckAU
Dim CreditEnquiryResponse As ISCreditEnquiryResponse
Dim CreditEnquiryResponseGreenId As ISCreditEnquiryResponse_GreenId_BackgroundCheck

' Assume Success
Main = True

' Get Workflow Executor
mWorkflowExecutor = DirectCast(otherParameters.GetObject("WorkflowExecutor"), finWorkflowExecutor)

' Handle Events
Select Case eventId
Case "BeforeItemAction"
    If workflowItem.ItemIdOriginal = "CE" Then
        ' Validate
        If Not mWorkflowExecutor.CanPerformWebServiceCall() Then
            Main = False
            finBL.Error.ErrorBegin("Cannot perform Web Service call.")
        End If

        If Main Then
            ' Get Main Account Client
            Client = workflow.Account.Clients(0).Client

            ' Create Credit Enquiry Request
            CreditEnquiryRequest = finBL.CreditBureau.CreateCreditEnquiryRequest("GreenId",
"Background Check (Australia)")

            ' Cast to correct type
            CreditEnquiryRequestGreenId = DirectCast(CreditEnquiryRequest,
                ISCreditEnquiryRequest_GreenId_BackgroundCheckAU)

            ' Update from Client
            Main =
finBL.CreditBureau.UpdateCreditEnquiryRequestFromClient(CreditEnquiryRequestGreenId,
                Client)

        End If

        ' Update Request Options (with those defined under Global Settings)
        ' NOTE: This step is not necessary if all options are being set in the next block
        If Main Then
            Main = finBL.CreditBureau.UpdateCreditEnquiryRequestOptions(CreditEnquiryRequestGreenId,
                Client)

        End If

        ' Set Options
        If Main Then
            With CreditEnquiryRequestGreenId
                .PerformAUBackgroundCheck_DnBCreditHeader = False
                .PerformAUBackgroundCheck_DvsDriversLicence = False
                .PerformAUBackgroundCheck_DvsMedicare = False
                .PerformAUBackgroundCheck_DvsPassport = False
                .PerformAUBackgroundCheck_DvsVisa = False
                .PerformAUNonDvsDriversLicence = False

                ' Clear Previous Address
                .PrevAddressFromAddressDetails(finBL.CreateAddressDetails())
            End With
        End If

        ' Execute Request
        ' NOTE: Creates a Client Log and links this to this Workflow Item
        If Main Then
            mWorkflowExecutor.WebServiceCallBegin()
            Main = finBL.CreditBureau.ExecuteCreditEnquiry(CreditEnquiryRequestGreenId,
                CreditEnquiryResponse, Nothing, True,
                workflow.Pk, workflowItem.pk)
```

```

        mWorkflowExecutor.WebServiceCallEnd()
    End If

    ' Examine the Response
    If Main Then
        CreditEnquiryResponseGreenId = DirectCast(CreditEnquiryResponse,
                                                    ISCreditEnquiryResponse_GreenId_BackgroundCheck)

        ' Set Outcome
        If CreditEnquiryResponseGreenId.OverallOutcomeState =
iseGreenIdOverallOutcomeState.Verified Then
            otherParameters.SetString("Outcome", "Verified")
            otherParameters.SetBoolean("OutcomeSuccess", True)
        Else
            otherParameters.SetString("Outcome", "NotVerified")
            otherParameters.SetBoolean("OutcomeSuccess", False)
        End If
        otherParameters.SetString("StatusNotes",
                                   CreditEnquiryResponseGreenId.OverallOutcomeStateText)
    End If
End If
End Select

```

Note the following:

- Since the Workflow Item is NOT set to repeat for each Client, the Workflow Item's, the Account's 'Main' Client is retrieved.
 - If the Workflow Item was repeated for each Client, the Workflow Item's `GetSourceObject` method could be used to retrieve the relevant Client.
- The Workflow Item is set to unsuccessful using `otherParameters.SetBoolean("OutcomeSuccess", False)`
 - This means that the item will display a 'Failed' icon.

Automatic Applicant Credit Enquiry - Centrix (New Zealand)

This example performs an automatic Credit Enquiry using the Centrix (New Zealand) service for an **Account Application** type Workflow.

The 'CE' Workflow Item is set to repeat for each Applicant.

See the [Automatic Client Credit Enquiry – Veda \(New Zealand\)](#) for more information.

```
Dim AccountAppApplicant As finAccountAppApplicant
Dim Client As finClient
Dim CreditEnquiryRequest As ISCreditEnquiryRequest
Dim CreditEnquiryRequestCentrix As ISCreditEnquiryRequest_CentrixNZ_ConsumerCreditCheck
Dim CreditEnquiryResponse As ISCreditEnquiryResponse
Dim CreditEnquiryResponseCentrix As ISCreditEnquiryResponse_CentrixNZ_Individual
Dim SourceObject As Object

' Assume Success
Main = True

' Get Workflow Executor
mWorkflowExecutor = DirectCast(otherParameters.GetObject("WorkflowExecutor"), finWorkflowExecutor)

' Handle Events
Select Case eventId
Case "BeforeItemAction"
    If workflowItem.ItemIdOriginal = "CE" Then
        ' Validate
        If Not mWorkflowExecutor.CanPerformWebServiceCall() Then
            Main = False
            finBL.Error.ErrorBegin("Cannot perform Web Service call.")
        End If

        ' Get Applicant from Workflow Item
        If Main Then
            If workflowItem.GetSourceObject(SourceObject) Then
                AccountAppApplicant = DirectCast(SourceObject, finAccountAppApplicant)
            Else
                Main = False
            End If
        End If

        ' Create Client from Applicant
        If Main Then
            Main = AccountAppApplicant.CreateClient(True, Client)
        End If

        If Main Then
            ' Create Credit Enquiry Request
            CreditEnquiryRequest = finBL.CreditBureau.CreateCreditEnquiryRequest("CentrixNZ",
                "Consumer Credit Check")

            ' Cast to correct type
            CreditEnquiryRequestCentrix = DirectCast(CreditEnquiryRequest,
                ISCreditEnquiryRequest_CentrixNZ_ConsumerCreditCheck)

            ' Update from Client
            Main =
            finBL.CreditBureau.UpdateCreditEnquiryRequestFromClient(CreditEnquiryRequestCentrix,
                Client)

        End If

        ' Update Request Options (with those defined under Global Settings)
        ' NOTE: This step is not necessary if all options are being set in the next block
        If Main Then
            Main = finBL.CreditBureau.UpdateCreditEnquiryRequestOptions(CreditEnquiryRequestCentrix,
                Client)

        End If

        ' Set Options
        If Main Then
            With CreditEnquiryRequestCentrix
                .EnquiryReasonDescription = "Credit Application"
                .ProductTypeDescription = "Consumer Finance"
                .ApplicantTypeDescription = "Single Applicant"

                '.IncludeCourtFinesData = True
            End With
        End If
    End Select
```

```

' Execute Request
' NOTE: Creates a Client Log and links this to this Workflow Item
If Main Then
  mWorkflowExecutor.WebServiceCallBegin()
  Main = finBL.CreditBureau.ExecuteCreditEnquiry(CreditEnquiryRequestCentrix,
                                                  CreditEnquiryResponse, Nothing, True,
                                                  workflow.Pk, workflowItem.pk)
  mWorkflowExecutor.WebServiceCallEnd()
End If

' Examine the Response
If Main Then
  CreditEnquiryResponseCentrix = DirectCast(CreditEnquiryResponse,
                                             ISCreditEnquiryResponse_CentrixNZ_Individual)

  ' Set Outcome
  If CreditEnquiryResponseCentrix.BankruptcyCountUndischarged > 5 Then
    otherParameters.SetString("Outcome", "Fail")
    otherParameters.SetString("StatusNotes", String.Format("{0} bankruptcies",
                                                           CreditEnquiryResponseCentrix.BankruptcyCount))
  ElseIf CreditEnquiryResponseCentrix.BankruptcyCount > 0 Then
    otherParameters.SetString("Outcome", "Refer")
    otherParameters.SetString("StatusNotes", String.Format("{0} bankruptcies",
                                                           CreditEnquiryResponseCentrix.BankruptcyCount))
  Else
    otherParameters.SetString("Outcome", "Pass")
    otherParameters.SetString("StatusNotes", "No bankruptcies")
  End If
End If
End Select

```

Note the following:

- Since the Workflow Item is set to repeat for each Applicant, the Workflow Item's `GetSourceObject` method can be used to retrieve the Applicant.
 - If this Workflow Item was not repeated for each Applicant then the `workflow.AccountApp.GetMainApplicant` method could be used to retrieve the 'Main' Applicant.
- The `finAccountAppApplicant.CreateClient` method is used to create a temporary Client object from the Applicant.
 - The Account Application Type's Script allows this to be customised.

Documents

Workflow Items can be used to publish Documents, e.g., Word VBA templates or Email or SMS messages.

All Documents in finPOWER Connect are handled in the following way:

- A Log record (e.g., a Client Log) is created defining the Document to publish.
 - The Log also contains additional information when creating Email and SMS documents.
- The Log can be published either via the Log form or the Publish Documents wizard.
 - At this point, the Word document is generated or the Email or SMS message sent.

The Workflow Type combined with the 'Document Type' and 'File Type' of the Document (e.g., Word, Email or SMS) determines the Logs that are created when actioning a 'Send Document' type Workflow Item:

Workflow Type	Document Type	File Type	Log Created
Account	Account	Word VBA Excel VBA Log Script HTML	A single Account Log per Workflow Item (e.g., if repeating for each Client, multiple Workflow Items will have been created). The Workflow Item linked to the Log details the Client if applicable.
		SMS	An Account Log for each Client that could receive the SMS.
		Email	An Account Log for each Client that could receive the Email if the generated Message, Subject, CC or BCC are NOT the same for all Clients; otherwise, a single Account Log will be created to send the Email to multiple recipients.
	Client	Word VBA Excel VBA Log Script HTML	A single Client Log per Workflow Item (e.g., if repeating for each Client, multiple Workflow Items will have been created). If not repeating for each Client, the Account's Main Client is used.
		SMS	A single Client Log per Workflow Item (e.g., if repeating for each Client, multiple Workflow Items will have been created). If not repeating for each Client, the Account's Main Client is used.
		Email	A single Client Log per Workflow Item (e.g., if repeating for each Client). If not repeating for each Client, the Account's Main Client is used.
Client	Client	Word VBA Excel VBA Log Script HTML	A single Client Log .

		SMS	A single Client Log .
		Email	A single Client Log .

NOTE: In addition to the above, a Workflow Item can have a 'Source Type' and 'Source Id' set via Script (this is the same mechanism used when repeating an Account Workflow Item for each Account Client). This also affects the type of Log created.

Workflow Type Item Wizard

Workflow Type Scripts

'Send Document' Type Items

A Workflow Type Script can affect 'Send Document' type Workflow items as follows:

- **BeforeItemAction event**
 - This event is called before actioning any type of Workflow Item and is described fully in the [BeforeItemAction](#) section.
- **CouldActionItem event**
 - This event is called before attempting to action a 'Send Document' type Item.
 - The event allows the Script to determine whether it is worth attempting to send the Document, e.g., to ensure all recipients can be contacted.
 - This event is described fully in the [CouldActionItem](#) section.

Fully Scripted Solution

A Workflow Type Script can send Documents when processing other items.

A fully scripted solution is recommended if you want full control over sending of a Document, e.g.:

- You want to send to specific Account Clients based on criteria not available when configuring the Workflow Type Item.
- You want full control over CC or BCC for an Email.
 - The basics can be handled by a Document Script but it may be more intuitive simply Script this in the Workflow Type Script.
- You want a fully customised message (e.g., Email message) without relying on smart tag replacement and the template message configured in the Document.
 - The Document Script's 'GenerateMessage' event can handle this but, again, it may be more intuitive to simply Script this in the Workflow Type Script.

'TODO: Example of blank item that does stuff, e.g., generates an AA, AAS and AAE document

Document Scripts

A Document can define a Script.

When generating SMS and Email type Documents, the Workflow Executor calls the Document's 'GenerateMessage' Script event to enable the Script to provide (or update) the message to be sent.

NOTE: Version 1 type Workflows called the 'GenerateMessage' Script event once to allow the Document Script to provide a template message (which could contain smart tags such as [Client.ClientId]) and then once for each Client.

This behaviour has been simplified for Version 2 type Workflows. The Generate Message Script is called for each Client that is to receive the SMS or Email thereby allowing a message to be generated for each Client.

WARNING: Versions prior to 2.03.02 did not always pass the correct object to the Document Script, e.g., an Account Document may have been passed a Client object.

This functionality (and the details in the sections below) has been changed so that Documents are always passed the correct object type.

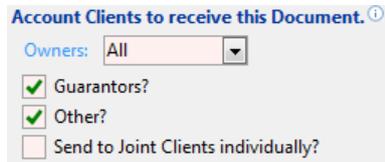
The `finWorkflowItem` has a `GetSourceObjectForDocument` method that determines the 'source' object that is sent to the 'GenerateMessage' Document Script event.

The source object will always match the Document's Type.

Account SMS

The Document's 'GenerateMessage' Script event will be called multiple times; once for each recipient (as determined by the Workflow Item's `GetDocumentRecipients` method).

For an Account Document, the 'Other' page of the Documents form determines which Clients should be included as recipients:



Account Clients to receive this Document. ⓘ

Owners: All ▼

Guarantors?

Other?

Send to Joint Clients individually?

This is used in conjunction with the Workflow Item's settings to determine whether the Document should be sent to the Branch, Dealer etc:



Send also to the following. ⓘ

Branch

Broker

Broker Employee

Dealer

Dealer Employee

The Script will receive the following parameters:

- **source**
 - A `finAccount` object defined either by the Workflow or the Workflow Item (if the Workflow Item is set to have a `SourceObjectType` of Account).
 - ✦ **NOTE:** If the Workflow (or Workflow Item) target an Account Application, a temporary `finAccount` object will be created from this Account Application.

The `eventArgs` parameter will ALWAYS contain the following:

- **Message**
 - The SMS message as defined on either the Document or the Workflow Item.
 - ✦ This may contain tag, e.g., `[Account.AccountId]` or `[Client.Name]`.
- **Workflow**
 - The Workflow (a `finWorkflow` object).
- **WorkflowItem**
 - The Workflow Item (a `finWorkflowItem` object).

The `eventArgs` parameter may OPTIONALLY contain the following:

- **AccountClient**
 - The Account Client (a `finAccountClient` object).
 - This will not exist if the recipient is a Branch, Dealer, Dealer Employee etc.
- **Client**
 - The Client (a `finClient` object).
 - The only time this will not exist is if the recipient is a Broker or Dealer Employee that does not link to a Client.
- **ClientEmployment**
 - The Client Employment (a `finClientEmployment` object).
 - The only time this will exist is if the recipient is a Broker or Dealer Employee.

Certain values can be overridden by adding the appropriate entry into the `returnValues` Key/Value List:

- **Message**

- Can be used to override the SMS message supplied in `eventArgs`.

- **Phone**

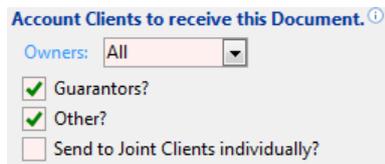
- Can be used to override or to specify an SMS phone number.

```
returnValues.SetString("message", "my custom message")
returnValues.SetString("phone", "027 12345678")
```

Account Email

The Document's 'GenerateMessage' Script event will be called multiple times; once for each recipient (as determined by the Workflow Item's `GetDocumentRecipients` method).

For an Account Document, the 'Other' page of the Documents form determines which Clients should be included as recipients:



Account Clients to receive this Document. ⓘ

Owners: All ▾

Guarantors?

Other?

Send to Joint Clients individually?

This is used in conjunction with the Workflow Item's settings to determine whether the Document should be sent to the Branch, Dealer etc:



Send also to the following. ⓘ

Branch

Broker

Broker Employee

Dealer

Dealer Employee

The Script will receive the following parameters:

- **source**
 - A `finAccount` object defined either by the Workflow or the Workflow Item (if the Workflow Item is set to have a `SourceObjectType` of Account).
 - ✦ **NOTE:** If the Workflow (or Workflow Item) target an Account Application, a temporary `finAccount` object will be created from this Account Application.

The `eventArgs` parameter will ALWAYS contain the following:

- **Message**
 - The Email message as defined on either the Document or the Workflow Item.
 - ✦ This may contain tag, e.g., `[Account.AccountId]` or `[Client.Name]`.
- **Subject**
 - The Email subject as defined on either the Document or the Workflow Item.
 - This may contain tag, e.g., `[Account.AccountId]` or `[Client.Name]`.
- **Workflow**
 - The Workflow (a `finWorkflow` object).
- **WorkflowItem**
 - The Workflow Item (a `finWorkflowItem` object).

The `eventArgs` parameter may OPTIONALLY contain the following:

- **AccountClient**
 - The Account Client (a `finAccountClient` object).
 - This will not exist if the recipient is a Branch, Dealer, Dealer Employee etc.
- **Client**
 - The Client (a `finClient` object).
 - The only time this will not exist is if the recipient is a Broker or Dealer Employee that does not link to a Client.
- **ClientEmployment**
 - The Client Employment (a `finClientEmployment` object).

- The only time this will exist is if the recipient is a Broker or Dealer Employee.

Certain values can be overridden by adding the appropriate entry into the `returnValues` Key/Value List:

- **Message**

- Can be used to override the Email message supplied in `eventArgs`.

- **Subject**

- Can be used to override the Email subject supplied in `eventArgs`.

- **To**

- Can be used to override or to specify one or more Email recipients.
- This can be either a comma or semi-colon separated list of Email addresses.

- **CC**

- Can be used to specify one or more Email CC recipients.
- This can be either a comma or semi-colon separated list of Email addresses.

- **BCC**

- Can be used to specify one or more Email BCC recipients.
- This can be either a comma or semi-colon separated list of Email addresses.

```
returnValues.SetString("message", "my custom message")
returnValues.SetString("cc", "manager@mycompany.com; sales@mycompany.com")
```

Client SMS

The Document's 'GenerateMessage' Script event will be called for the Client.

If the Workflow is an Account type Workflow then the Client will be the Account's Main Client unless the Workflow Item is set to 'Repeat this item for each Account Client'.

The Script will receive the following parameters:

- **source**
 - A `finClient` object defined either by the Workflow or the Workflow Item (if the Workflow Item is set to have a `SourceObjectType` of Client).
 - **WARNING:** If this is an Account Workflow and this Client Document has been set to be sent to the Dealer Employee or Broker Employee then this may be `Nothing` if the Employee is not linked to a Client.

The `eventArgs` parameter will ALWAYS contain the following:

- **Message**
 - The SMS message as defined on either the Document or the Workflow Item.
 - ✦ This may contain tag, e.g., `[Client.Name]`.
- **Workflow**
 - The Workflow (a `finWorkflow` object).
- **WorkflowItem**
 - The Workflow Item (a `finWorkflowItem` object).

The `eventArgs` parameter may OPTIONALLY contain the following:

- **ClientEmployment**
 - The Client Employment (a `finClientEmployment` object).
 - The only time this will exist is if the recipient is a Broker or Dealer Employee.

Certain values can be overridden by adding the appropriate entry into the `returnValues` Key/Value List:

- **Message**
 - Can be used to override the SMS message supplied in `eventArgs`.
- **Phone**
 - Can be used to override or to specify an SMS phone number.

```
returnValues.SetString("message", "my custom message")
returnValues.SetString("phone", "027 12345678")
```

Client Email

The Document's 'GenerateMessage' Script event will be called for the Client.

If the Workflow is an Account type Workflow then the Client will be the Account's Main Client unless the Workflow Item is set to 'Repeat this item for each Account Client'.

The Script will receive the following parameters:

- **source**
 - A `finClient` object defined either by the Workflow or the Workflow Item (if the Workflow Item is set to have a `SourceObjectType` of Client).
 - **WARNING:** If this is an Account Workflow and this Client Document has been set to be sent to the Dealer Employee or Broker Employee then this may be `Nothing` if the Employee is not linked to a Client.

The `eventArgs` parameter will ALWAYS contain the following:

- **Message**
 - The Email message as defined on either the Document or the Workflow Item.
 - ✎ This may contain tag, e.g., `[Account.AccountId]` or `[Client.Name]`.
- **Subject**
 - The Email subject as defined on either the Document or the Workflow Item.
 - This may contain tag, e.g., `[Account.AccountId]` or `[Client.Name]`.
- **Workflow**
 - The Workflow (a `finWorkflow` object).
- **WorkflowItem**
 - The Workflow Item (a `finWorkflowItem` object).

The `eventArgs` parameter may OPTIONALLY contain the following:

- **ClientEmployment**
 - The Client Employment (a `finClientEmployment` object).
 - The only time this will exist is if the recipient is a Broker or Dealer Employee.

Certain values can be overridden by adding the appropriate entry into the `returnValues Key/Value List`:

- **Message**
 - Can be used to override the Email message supplied in `eventArgs`.
- **Subject**
 - Can be used to override the Email subject supplied in `eventArgs`.
- **To**
 - Can be used to override or to specify one or more Email recipients.
 - This can be either a comma or semi-colon separated list of Email addresses.
- **CC**
 - Can be used to specify one or more Email CC recipients.
 - This can be either a comma or semi-colon separated list of Email addresses.
- **BCC**
 - Can be used to specify one or more Email BCC recipients.
 - This can be either a comma or semi-colon separated list of Email addresses.

```
returnValues.SetString("message", "my custom message")
returnValues.SetString("cc", "manager@mycompany.com; sales@mycompany.com")
```

Account Application SMS

The Document's 'GenerateMessage' Script event will be called multiple times; once for each recipient (as determined by the Workflow Item's `GetDocumentRecipients` method).

For an Account Application Document, the 'Other' page of the Documents form determines which Applicants should be included as recipients:



Applicants to receive this Document. ⓘ

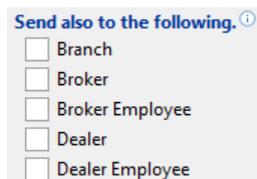
Owners: All ▼

Guarantors?

Other?

Send to Joint Applicants individually?

This is used in conjunction with the Workflow Item's settings to determine whether the Document should be sent to the Branch, Dealer etc:



Send also to the following. ⓘ

Branch

Broker

Broker Employee

Dealer

Dealer Employee

The Script will receive the following parameters:

- **source**
 - A `finAccountApp` object defined either by the Workflow or the Workflow Item (if the Workflow Item is set to have a `SourceObjectType` of `AccountApp`).

The `eventArgs` parameter will ALWAYS contain the following:

- **Message**
 - The SMS message as defined on either the Document or the Workflow Item.
 - ✎ This may contain tag, e.g., `[AccountApp.AccountAppId]` or `[Applicant.Name]`.
- **Workflow**
 - The Workflow (a `finWorkflow` object).
- **WorkflowItem**
 - The Workflow Item (a `finWorkflowItem` object).

The `eventArgs` parameter may OPTIONALLY contain the following:

- **Applicant**
 - The Applicant (a `finAccountAppApplicant` object).
 - This will not exist if the recipient is a Branch, Dealer, Dealer Employee etc.
- **Client**
 - The Client (a `finClient` object).
 - The only time this will exist is if the recipient is a Branch, Broker or Dealer or a Broker/Dealer Employee that links to a Client.
- **ClientEmployment**
 - The Client Employment (a `finClientEmployment` object).
 - The only time this will exist is if the recipient is a Broker or Dealer Employee.

Certain values can be overridden by adding the appropriate entry into the `returnValues` Key/Value List:

- **Message**

- Can be used to override the SMS message supplied in `eventArgs`.
- **Phone**
 - Can be used to override or to specify an SMS phone number.

```
returnValues.SetString("message", "my custom message")
returnValues.SetString("phone", "027 12345678")
```

Account Application Email

The Document's 'GenerateMessage' Script event will be called multiple times; once for each recipient (as determined by the Workflow Item's `GetDocumentRecipients` method).

For an Account Application Document, the 'Other' page of the Documents form determines which Applicants should be included as recipients:



Applicants to receive this Document. ⓘ

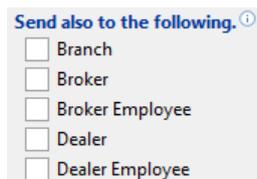
Owners: All ▼

Guarantors?

Other?

Send to Joint Applicants individually?

This is used in conjunction with the Workflow Item's settings to determine whether the Document should be sent to the Branch, Dealer etc:



Send also to the following. ⓘ

Branch

Broker

Broker Employee

Dealer

Dealer Employee

The Script will receive the following parameters:

- **source**
 - A `finAccountApp` object defined either by the Workflow or the Workflow Item (if the Workflow Item is set to have a `SourceObjectType` of `AccountApp`).

The `eventArgs` parameter will ALWAYS contain the following:

- **Message**
 - The SMS message as defined on either the Document or the Workflow Item.
 - ✦ This may contain tag, e.g., `[AccountApp.AccountAppId]` or `[Applicant.Name]`.
- **Subject**
 - The Email subject as defined on either the Document or the Workflow Item.
 - ✦ This may contain tag, e.g., `[AccountApp.AccountAppId]` or `[Applicant.Name]`.
- **Workflow**
 - The Workflow (a `finWorkflow` object).
- **WorkflowItem**
 - The Workflow Item (a `finWorkflowItem` object).

The `eventArgs` parameter may OPTIONALLY contain the following:

- **Applicant**
 - The Applicant (a `finAccountAppApplicant` object).
 - This will not exist if the recipient is a Branch, Dealer, Dealer Employee etc.
- **Client**
 - The Client (a `finClient` object).
 - The only time this will exist is if the recipient is a Branch, Broker or Dealer or a Broker/Dealer Employee that links to a Client.
- **ClientEmployment**
 - The Client Employment (a `finClientEmployment` object).
 - The only time this will exist is if the recipient is a Broker or Dealer Employee.

Certain values can be overridden by adding the appropriate entry into the `returnValues` Key/Value List:

- **Message**
 - Can be used to override the Email message supplied in `eventArgs`.
- **Subject**
 - Can be used to override the Email subject supplied in `eventArgs`.
- **To**
 - Can be used to override or to specify one or more Email recipients.
 - This can be either a comma or semi-colon separated list of Email addresses.
- **CC**
 - Can be used to specify one or more Email CC recipients.
 - This can be either a comma or semi-colon separated list of Email addresses.
- **BCC**
 - Can be used to specify one or more Email BCC recipients.
 - This can be either a comma or semi-colon separated list of Email addresses.

```
returnValues.SetString("message", "my custom message")
returnValues.SetString("cc", "manager@mycompany.com; sales@mycompany.com")
```

HTML Widgets

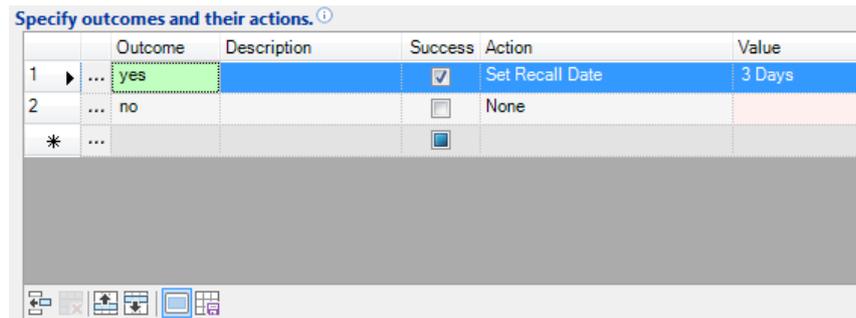
As of finPOWER Connect 3.00.06, HTML Widget' type Items were introduced for Version 2 Workflow Types.

Workflow Type Item

HTML Widget Workflow Type Items allow an HTML Widget Script to be defined, e.g.:



They also allow Outcomes to be defined, e.g.:



The reason Outcomes can be defined is that the HTML Widget can opt to either Action the Workflow Item associated with it or, if it makes more sense, set the Outcome of the Workflow Item which will, in turn, perform any Outcome Actions.

HTML Widget

When an 'HTML Widget' type item is actioned, the HTML Widget Script will be passed the primary key of the Workflow and the Workflow Item as startup parameters.

Creating a new HTML Widget and pasting template code gives the option of pasting a 'Workflow' code template. This is shown below:

```
Option Explicit On
Option Strict On

' Objects
Private mCombinedParameters As ISKeyValueList
Private mParameters As ISKeyValueList
Private mStartupParameters As ISKeyValueList

Public Function Main(eventId As String, parametersJson As String, startUpParametersJson As String,
hostingContext As iseFinHtmlWidgetHostingContext, requestInfo As finScriptRequestInfo, ByRef
returnValue As String) As Boolean

' Assume Success
Main = True

' Get Parameters
mParameters = finBL.CreateKeyValueList()
mParameters.FromSimpleJsonString(parametersJson)
mStartupParameters = finBL.CreateKeyValueList()
mStartupParameters.FromSimpleJsonString(startUpParametersJson)

' Get Combined Parameters (Startup Parameters updated with event Parameters)
mCombinedParameters = mStartupParameters.Clone()
mCombinedParameters.UpdateFromKeyValueList(mParameters, True)

' Handle Events
Select Case eventId
Case ""
' Main event, i.e., return initial HTML content (excluding html and body tags)
returnValue = ScriptInfo.TemplateText
```

```

    Case "Complete"
        ' Complete Workflow Item
        Main = Complete()

    Case Else
        Main = False
        finBL.Error.ErrorBeginFormat("Unhandled event '{0}'.", eventId)

End Select

End Function

Private Function Complete() As Boolean
    Dim Outcome As String
    Dim Success As Boolean
    Dim WorkflowPk As Integer
    Dim WorkflowItemPk As Integer
    Dim WorkflowExecutor As finWorkflowExecutor

    ' Assume Success
    Success = True

    ' Get Parameters
    With mCombinedParameters
        Outcome = .GetString("outcome")
        WorkflowPk = .GetInteger("workflowPk")
        WorkflowItemPk = .GetInteger("workflowItemPk")
    End With

    ' Create Workflow Executor
    WorkflowExecutor = finBL.CreateWorkflowExecutor()
    Success = WorkflowExecutor.WorkflowLoadPk(WorkflowPk)

    ' Action/ Set Outcome
    If Success Then
        If Len(Outcome) = 0 Then
            Success = WorkflowExecutor.ExecuteWorkflowItemPerformAction(WorkflowItemPk, True, True, "my
notes")
        Else
            Success = WorkflowExecutor.ExecuteWorkflowItemSetOutcome(WorkflowItemPk, outcome, True,
True, "my notes")
        End If
    End If

    Return Success
End Function

```

Pasting template code also supplies the following Template Text:

```

<div class="is-widget-content">
    <button id="cmdComplete" class="is-button">Complete</button>
    <button id="cmdComplete_Yes" class="is-button">Complete with Outcome "Yes"</button>
</div>

<script>
// Initialise
$(function () {
    // Handle buttons
    $("#cmdComplete").click(function () { Complete(""); });
    $("#cmdComplete_Yes").click(function () { Complete("Yes"); });
});

function Complete(outcome) {
    widget.GetString("Complete", {outcome: outcome},
        function (data) {
            // Success
            widget.RefreshParent();
            widget.Close();
        },

        function (data) {
            // Failed
            widget.UI.MsgBoxAlert(data);
        });
}
</script>

```

NOTE: The finPOWER Connect HTML Widgets document contains more details on creating and configuring HTML Widgets.

Page Sets

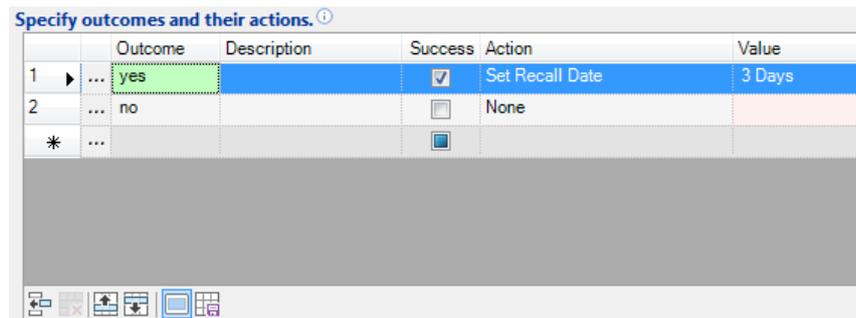
As of finPOWER Connect 2.02.06, 'Page Set' type Items were introduced for Version 2 Workflow Types.

Workflow Type Item

Page Set Workflow Type Items allow a Page Set to be defined, e.g.:



They also allow Outcomes to be defined, e.g.:



The reason Outcomes can be defined is that the Page Set can opt to either Action the Workflow Item associated with it or, if it makes more sense, set the Outcome of the Workflow Item which will, in turn, perform any Outcome Actions.

Page Set

When a 'Page Set' type item is actioned, the Page Set will be passed the primary key of the Workflow and the Workflow Item.

Creating a new Page Set and pasting template code gives the option of pasting a 'Workflow' code template. This is shown below:

```
Option Explicit On
Option Strict On

' Objects
Private mWorkflow As finWorkflow
Private mWorkflowItem As finWorkflowItem

' Reporting and User Interface Objects
Public mReports As ISfinReports
Public mUI As ISUserInterfaceBL

Public Overrides Function Initialise() As Boolean

    Dim Workflow As finWorkflow
    Dim WorkflowItem As finWorkflowItem
    Dim WorkflowPk As Integer
    Dim WorkflowItemPk As Integer

    ' Assume Success
    Initialise = True

    ' Initialise
    mReports = DirectCast(psh.Reports, ISfinReports)
    mUI = DirectCast(psh.UserInterface, ISUserInterfaceBL)

    ' Get Parameters
    Workflow = DirectCast(psh.Parameters.GetObject("workflow"), finWorkflow)
    WorkflowItem = DirectCast(psh.Parameters.GetObject("workflowItem"), finWorkflowItem)
    WorkflowPk = psh.Parameters.GetInteger("workflowPk")
    WorkflowItemPk = psh.Parameters.GetInteger("workflowItemPk")
```

```

' Get/ Create Objects
If Workflow Is Nothing Then
' Load
mWorkflow = finBL.CreateWorkflow()
Initialise = mWorkflow.LoadPk(WorkflowPk)

' Get Workflow Item
If WorkflowItemPk <> 0 Then
mWorkflowItem = mWorkflow.GetItemByPk(WorkflowItemPk)
If mWorkflowItem Is Nothing Then
Initialise = False
finBL.Error.ErrorBeginFormat("Workflow Item {0} not found.", WorkflowItemPk)
End If
End If
Else
' Objects passed to Page Set
mWorkflow = Workflow
mWorkflowItem = WorkflowItem
End If

' Validate
If Initialise Then
' E.g.
' Check mWorkflowItem is not Nothing
If mWorkflowItem Is Nothing Then
Initialise = False
finBL.Error.ErrorBegin("This Page Set requires a valid Workflow Item.")
End If
End If

' Make Page Set Read-Only?
If Initialise Then
' psh.ReadOnlySet()
End If

' Load Fields
If Initialise Then
Fields_Load()
End If

End Function

Private Sub Fields_Load()

' Workflow
With mWorkflow.UserData
' Update Page Objects, e.g.
' txtImpact.Text = .GetString("Impact")
End With

' Workflow Item
With mWorkflowItem.UserData
' Update Page Objects, e.g.
' txtReason.Text = .GetString("Reason")
End With

End Sub

Private Sub Fields_Save()

' Workflow
With mWorkflow.UserData
' Update properties from Page Objects, e.g.
' .SetString("Impact", txtImpact.Text)
End With

' Workflow Item
With mWorkflowItem.UserData
' Update properties from Page Objects, e.g.
' .SetString("Reason", txtReason.Text)
End With

End Sub

Public Sub PageSet_CommandButtonClick(sender As Object,
e As finPageSetHandlerCommandButtonClickEventArgs) Handles
Me.CommandButtonClick

Dim Ok As Boolean

```

```

Select Case e.CommandButton
Case iseFinPageSetCommandButton.Finish, iseFinPageSetCommandButton.Ok
' Assume Success
Ok = True

' Save Fields
Fields_Save()

' Save Workflow
If Ok Then Ok = mWorkflow.Save()

' Action Workflow Item
If Ok Then
Ok = psh.WorkflowItemPerformAction(mWorkflow.Pk, mWorkflowItem.Pk, "Status notes.")
' Ok = psh.WorkflowItemSetOutcome(mWorkflow.Pk, mWorkflowItem.Pk, "Outcome", "Status
notes.")
End If

' Error?
If Not Ok Then
mUI.ErrorMessageShow()

' Do not close this Page Set
e.Cancel = True
End If

End Select

End Sub

```

Of note in the template code are the following:

- The Page Set code can also accept 'workflow' and 'workflowItem' object parameters.
 - This code has been added for future flexibility and is not yet used.
- When the 'OK' or 'Finish' button is clicked, the following is done:
 - The Workflow is saved.
 - The Workflow item is actioned (or the outcome set).
 - ✦ You may wish for this part to be conditional upon certain information having been entered, i.e., save the Workflow but not action the item at this point.

NOTE: The finPOWER Connect Page Sets document contains more details on creating and configuring Page Sets.

Appendix A – Helper Functions

Workflow Functions (`finWorkflowFunctions`)

Although use of `finBL.WorkflowFunctions` is generally prohibited from within a Workflow Type Script, certain functions can still be used from outside of the Workflow Type Script, e.g., from conversion and other Scripts.

Certain functions are accessed from within `finPOWER Connect`, e.g., to skip Item Groups from a Summary Page or to add a new Item Group from the Items page of the Workflows form.

Commonly used Workflow Functions that can be used outside of the Workflow Type Script are:

- **WorkflowDeleteItemGroup**
 - Used to delete a Workflow Item Group from the User Interface, i.e., flag it as Deleted.
- **WorkflowItemStatusNotApplicableToggle**
 - Used to toggle the 'Not Applicable' status of a Workflow Item, e.g., from Summary Page links.
- **WorkflowItemStatusSkippedToggle**
 - Used to toggle the 'Skipped' status of a Workflow Item, e.g., from Summary Page links.

Adding and Inserting Workflow Items

The following methods exist in `finWorkflowItems` and will add items to the end of the collection:

Method	Details
<code>AddAllocateToUser</code>	
<code>AddBankAccountEnquiry</code>	
<code>AddBankAccountEnquiryReview</code>	
<code>AddCancelWorkflow</code>	
<code>AddCloseWorkflow</code>	
<code>AddCheckListItem</code>	
<code>AddCreditEnquiry</code>	
<code>AddDecisionCard</code>	
<code>AddDocument</code>	
<code>AddNone</code>	Typically used to add an item with an <code>ItemId</code> that can be processed by the Script.
<code>AddOutgoingCommunication</code>	
<code>AddPaymentArrangement</code>	
<code>AddQuestion</code>	Use helper methods of the returned <code>finWorkflowItem.OutcomeItems</code> collection to add outcomes.
<code>AddRegisterSecurityStatement</code>	
<code>AddReview</code>	The period can be a description of mnemonic, e.g., "2d" or "3 Months".
<code>AddSecurityRegisterSearch</code>	
<code>AddRegisterSecurityStatement</code>	
<code>AddTest</code>	Use the returned <code>finWorkflowItem</code> to set test criteria relevant to the Workflow's target object type, e.g., <code>MonitorCategoryATest</code> and <code>MonitorCategoryARange</code> .
<code>AddWait</code>	The period can be a description of mnemonic, e.g., "2d" or "3 Months".

Each of the above `Add` methods has a corresponding `Insert` method, e.g., `InsertWait`. The insert methods take the same parameter plus a first parameter of `index` where `index` is the zero-based index of the item to insert before or can be one of the following, special values:

- -1
 - Add to the end of the collection (does exactly the same as the corresponding `Add` method).
- -2

- Inserts the item BEFORE the current item if this is the current Item Group.
- -3
 - Inserts the item AFTER the current item if this is the current Item Group.

WARNING: When inserting items from the 'AfterItemAction' event, the current item will not be the item that was just actioned but the new current item (usually the next item).

You can also add and insert copies of a Workflow Item using the `Add` and `Insert` methods together with the Workflow Item's `Clone` method, e.g.:

```
Case "BeforeItemAction"
  If workflowItem.ItemIdOriginal = "CHKA" Then
    ' Insert cloned item AFTER current item
    With mWorkflowExecutor.CurrentItemGroup.Items
      .Insert(mWorkflowExecutor.CurrentItemGroup.CurrentItemIndex + 1,
        .ItemByOriginalId("WAIT1").Clone())
    End With
  End If
```

NOTE: The `Insert` method does not recognise the special -1, -2 and -3 values for index that the helper methods details above use.

WARNING: The `Clone` method copies nearly all Workflow Item properties so ensure the clone is not used on an already actioned item, including the item being actioned in the 'AfterItemAction' event where properties such as `Status` have already been updated.

The following methods exist in `finWorkflowExecutor` to add items to the Workflow:

Method	Details
<code>AddItemGroupFromWorkflowType</code>	<p>Adds a new Item Group from those defined on the Workflow Type.</p> <p>The <code>index</code> parameter determines where to insert the Item Group and has the following special values:</p> <ul style="list-style-type: none"> • -1 <ul style="list-style-type: none"> ○ To the end of the Workflow.
<code>AddItemGroupItemsFromWorkflowType</code>	<p>Inserts Workflow Items, as defined on the Workflow Type, into the current Item Group.</p> <p>The <code>index</code> parameter determines where to insert the items and has the following special values:</p> <ul style="list-style-type: none"> • -1 <ul style="list-style-type: none"> ○ Add to the end of the collection. • -2 <ul style="list-style-type: none"> ○ Inserts the item BEFORE the current item. • -3 <ul style="list-style-type: none"> ○ Inserts the item AFTER the current item.

Skipping Items and Item Groups

The following methods exist in `finWorkflowExecutor` and allow skipping of Workflow Items and Item Groups:

Method	Details
<code>ItemSetStatusSkipped</code>	Skip or unskip the specified Workflow Item.
<code>ItemSetStatusSkippedByItemId</code>	Skip or unskip an Item in the Current Item Group with the specified Item Id. NOTE: Only the first matched Workflow Item will be skipped.
<code>ItemSetStatusSkippedByOriginalItemId</code>	Skip or unskip an Item in the Current Item Group with the specified Original Item Id. NOTE: Only the first matched Workflow Item will be skipped.
<code>ItemGroupSetStatusSkipped</code>	Skip the specified Workflow Item Group.
<code>ItemGroupSetStatusSkippedByItemId</code>	Skip an Item Group with the specified Item Id and all unactioned items within it. NOTE: Only the first matched Item Group with a status of 'Not Started' or 'Open' will be skipped.
<code>ItemGroupSetStatusSkippedByItemIdOriginal</code>	Skip an Item Group with the specified Original Item Id and all unactioned items within it. NOTE: Only the first matched Item Group with a status of 'Not Started' or 'Open' will be skipped.
<code>SkipCurrentItemGroup</code>	Skip the current Item Group and optionally add a new Item Group to the end of the Workflow.

Setting Items and Item Groups to 'Not Applicable'

The following methods exist in `finWorkflowExecutor` and allow setting Workflow Items and Item Groups to 'Not Applicable':

Method	Details
<code>ItemSetStatusNotApplicable</code>	Set the specified Workflow Item to 'Not Applicable' or 'Not Started' (if it was already 'Not Applicable').
<code>ItemSetStatusNotApplicableByItemId</code>	Set the specified Workflow Item to 'Not Applicable' or 'Not Started' (if it was already 'Not Applicable'). The Item must be in the Current Item Group and have the specified Item Id. NOTE: Only the first matched Workflow Item will be affected.
<code>ItemSetStatusNotApplicableByOriginalItemId</code>	Set the specified Workflow Item to 'Not Applicable' or 'Not Started' (if it was already 'Not Applicable'). The Item must be in the Current Item Group and have the specified Original Item Id. NOTE: Only the first matched Workflow Item will be affected.
<code>ItemGroupSetStatusNotApplicable</code>	Set the specified Workflow Item Group to 'Not Applicable'.
<code>ItemGroupSetStatusNotApplicableByItemId</code>	Set an Item Group with the specified Item Id and all unactioned items within it to 'Not Applicable'. NOTE: Only the first matched Item Group with a status of 'Not Started' or 'Open' will be affected.
<code>ItemGroupSetStatusNotApplicableByItemIdOriginal</code>	Set an Item Group with the specified Original Item Id and all unactioned items within it to 'Not Applicable'. NOTE: Only the first matched Item Group with a status of 'Not Started' or 'Open' will be affected.

Appendix B – Frequently Asked Questions

Workflow Items

How do I get the Outcome of a Workflow Item?

- The `finWorkflowItem` object has a `StatusOutcome` String property.
- `WorkflowExecutor.GetStatusOutcomeByItemIdOriginal`
 - Works on the current item group and is useful in determining the outcome of another workflow item in the same group.
 - Returns a String value and does not error if no item with the specified original item id is found.

NOTE: When an item is completed from Script code, the outcome need not match any of the items in the `ItemOutcomes` collection.

Also, even items that do not have an `ItemOutcomes` collection such as Check List Items can have their `StatusOutcome` property set this way.

Why Does Completing an Item Group from 'BeforeItemAction' Skip the Current Item?

- When completing an Item Group, e.g., using `finWorkflowExecutor.CompleteCurrentItemGroup`, any incomplete items in the group are set to 'Skipped'.
- Because the current item's status during the 'BeforeItemAction' event is still set to 'Not Started', the current item will be skipped.
- The solution is to either:
 - Use the 'AfterItemAction' event instead. This way, the item's status is 'Completed' so the item will not be skipped.

Workflow Item Groups

How do I add an Item Group from outside of the Workflow Type Script?

Version 1 type Workflows can use the `finWorkflowFunctions.WorkflowAddItemGroupFromWorkflowType` method to add Item Groups to a Workflow from outside of the Workflow Type Script.

This method is not allowed for Version 2 type Workflows. Instead, a Workflow Executor must be used as per the following example:

```
Dim Ok As Boolean
Dim WorkflowExecutor As finWorkflowExecutor

' Assume Success
Ok = True

' Initialsie
WorkflowExecutor = finBL.CreateWorkflowExecutor()

' Load Workflow
Ok = WorkflowExecutor.WorkflowLoadPk(workflow.Pk)

' Add Item Group
If Ok Then Ok = WorkflowExecutor.AddItemGroupFromWorkflowType("TEST", True, Index, True)

' Save
If Ok Then Ok = WorkflowExecutor.WorkflowSave()

' Refresh Workflow (since Workflow Executor has its own copy of the Workflow)
' NOTE: Not necessary unless you need to access the up-to-date Workflow again
If Ok Then workflow.Refresh()
```

Miscellaneous

How Do I Refresh Workflows on another Form, e.g., the Task Manager?

- Forms support the concept of "Notification Actions". This allows other forms (or Scripts) to send messages to them, e.g., to tell the form to refresh a Workflows grid.
- Notification Actions are sent to a form using a "FormAction" type Application Shortcut.
- The format of the data that a Notification Action receives is not easily available however, the Form Details report (right-clicking on a Form's tab and selecting Special, Form Details) allows you to see a list of Notification Actions that the form will respond to.
- The Task Manager form has a special "WorkflowRefresh" notification action which the following Script sample shows how to use:

```
Public Function Main(parameters As ISKeyValueList) As Boolean
    ' Assume Success
    Main = True

    Dim ApplicationShortcut As ISApplicationShortcut
    Dim Params As ISKeyValueList

    Params = finBL.CreateKeyValueList()
    Params.SetBoolean("Added", True)

    ApplicationShortcut = finBL.CreateApplicationShortcutFormAction("TaskManager",
                                                                    "WorkflowRefresh",
                                                                    Params.ToXmlString(),
                                                                    False,
                                                                    True)

    Main = finBL.ExecuteApplicationShortcut(ApplicationShortcut)

End Function
```