

finPOWER Connect 3 Smart Lists

Version 3.05
16th March 2021

Table of Contents

Disclaimer.....	3
Version History.....	4
Introduction	5
Samples	5
Security	5
Smart Lists Overview	6
Tutorial	7
Adding a new "Smart List" Script	7
finSmartListHandler	8
Returning a List of Clients	9
Customising Columns	11
Showing a Row Summary	14
Making the Row Summary Optional	16
Allow Filtering by City	18
Adding a "Send Email" Action	22
Adding a "Request Address Update" Action.....	24
File Uploads	29
Notification Actions	31
Targeting a Specific User and Date As At	32
User	32
Date As At	33
Row Actions	34
Form Open	34
Forms with an id Parameter.....	34
Forms without an id Parameter	34
Quick Lists.....	35
Custom Quick Lists.....	36
Handling Multiple Lists	36

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, 2021.

Version History

[illegible]

Introduction

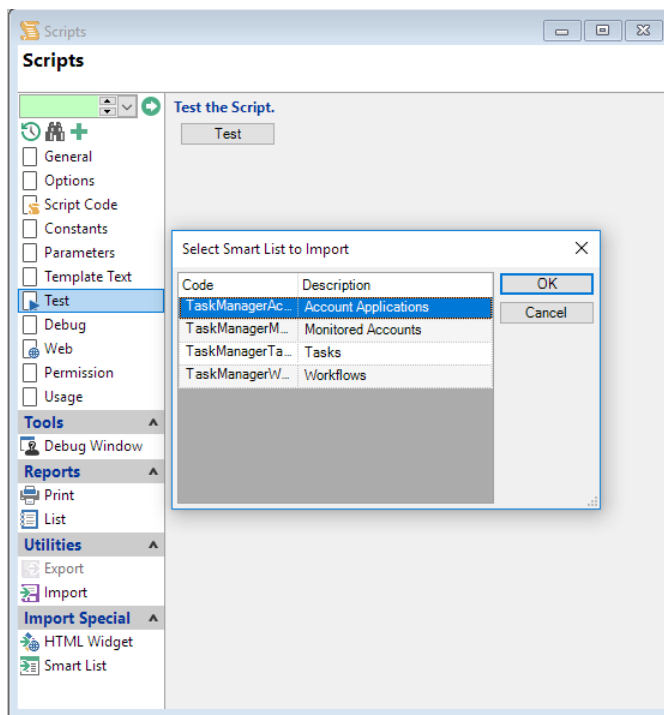
This document discusses finPOWER Connect Smart Lists and is focused on the creation and configuration of Smart Lists.

NOTE: The majority of this document is a tutorial detailing how to create a new Smart List.

Samples

Several Smart List samples are available. These are included within the finPOWER Connect setup and are stored in the **[ApplicationFolder]/Templates/SampleScripts** folder.

Also, system-supplied Smart Lists can be imported via the Scripts form, Import Special, Smart List action:



Security

When querying the finPOWER Connect database, always ensure that your SQL WHERE clause contains any filters relevant to the current User, e.g.:

```
sqb.sqlWhere.Append(finBL.CurrentUserInformation.FilterClientSqlWhere)
```

Variations on this exist for Accounts, Account Applications and Security Statements.

NOTE: Smart Lists displayed in the Tasks View in finPOWER Connect Cloud can be written to cater for viewing of another User's records.

See the section [Targeting a Specific User](#) for a code example.

Smart Lists Overview

- Smart Lists allow custom Lists (plus corresponding record previews and actions) to be created in finPOWER Connect.
 - These can be used in finPOWER Connect Cloud and will, in the future, be supported within the desktop version of finPOWER Connect.
- Smart Lists rely heavily on Scripting.
 - Smart List Scripts have access to the full finPOWER Connect business layer functionality.
- Smart Lists are used in the finPOWER Connect Cloud "Tasks View". They provide Task Manager like functionality that is better suited to a Web-based environment, e.g.:

The screenshot displays the 'Tasks View' in finPOWER Connect. The interface includes a top navigation bar with tabs for 'Today', 'Tasks', 'Workflows', 'Monitored', and 'Applications'. A left sidebar contains navigation options like 'All Tasks', 'My Logs', 'Review Dates', 'Account Quotes Outstanding', and filters for 'Outstanding Days' (0-10, 1-20, 21+), 'Tests', 'Napier Clients', and 'Clients by City'.

The main area shows a table of tasks under the '0-10 Days' filter. The table has columns for 'Type', 'Code', 'Name', 'Subject', and 'O/D Days'. Two tasks are listed:

Type	Code	Name	Subject	O/D Days
Account Application Log	ADEP0001	Deposit Test	Outgoing SMS	4
Account Application Log	ADEP0001	Deposit Test	Account Application Letter	9

On the right, a detailed view for the 'Outgoing SMS' task is shown. It includes the following information:

- Subject:** Outgoing SMS
- Date:** 21/06/2016 6:19PM (UTC+10:00) Hawaii
- Action:** account app sms
- Alert:** To Action by Admin on 12/01/2017 10:00AM Auckland
- SMS details:**
 - Phone: (27) 940 7647
 - Message: account app sms

Tutorial

This tutorial goes through the creation and gradual enhancement of a Smart List which displays finPOWER Connect Clients.

It is split into the following steps:

- [Adding a new "Smart List" Script](#)
- [Returning a List of Clients](#)
- [Customising Columns](#)
- [Showing a Row Summary](#)
- [Making the Row Summary Optional](#)
- [Allow Filtering by City](#)
- [Adding a "Send Email" Action](#)
- [Adding a "Request Address Update" Action](#)

Adding a new "Smart List" Script

In this section we will create a new "Smart List" Script and add the basic template code.

- From the Admin menu, select Scripts to open the Scripts form.
- Click the "Add" button to create a new Script and set the various properties as shown below:

- Select the "Script Code" page and use the "Paste template Script code" button (📋) to provide a starting point for the Smart List Script.
- Save the Script.

The Script now contains the following code:

```
Option Explicit On
Option Strict On

' Objects
Private mEventArgs As ISKeyValueList
Private mSmartListHandler As finSmartListHandler
```

```

Public Function Main(smartListHandler As finSmartListHandler,
                    eventId As String,
                    eventArgs As ISKeyValueList,
                    hostingContext As iseFinSmartListHostingContext,
                    scriptRequestInfo As finScriptRequestInfo) As Boolean

    ' Assume Success
    Main = True

    ' Initialise
    mEventArgs = eventArgs
    mSmartListHandler = smartListHandler

    ' Handle Events
    Select Case eventId
        Case "Initialise"
            ' Initialise
            Main = Initialise()

        Case "Execute"
            ' Execute
            Main = Execute()

        Case "GetRowSummary"
            ' Get Row Summary
            GetRowSummary()

        Case "ExecuteRowAction"
            ' Execute Row Action
            Main = ExecuteRowAction()

    End Select
End Function

Private Function Execute() As Boolean
End Function

Private Function Initialise() As Boolean

    Return True

End Function

Private Sub GetRowSummary()
End Sub

Private Function ExecuteRowAction() As Boolean
End Function

```

The remainder of this tutorial involves implementing and enhancing the various highlighted functions called by the `Main()` method.

finSmartListHandler

The most important parameter passed to the Script is `smartListHandler`.

This is a `finSmartListHandler` object that has been initialised from the Script and is the main object that the Script will use to communicate with the smart list, e.g., return a Data Table or get a Row Summary (e.g., an HTML Summary Page for a Client).

By the end of this section, the Smart List will look like this when viewed in finPOWER Connect Cloud:

Copy the following code and replace the `Execute()` method.

When run from the Scripts form, the following will be displayed:

	ClientId	Name	DateOfBirth	Status
1	M	Main Branch		0
2	HO	Head Office		0
3	NR	Northern Region		0
4	SR	Southern Region		0
5	CR	Central Region		0
6	C10000	Sample, Amelia...	29/08/1953	20
7	C10001	Morris, Peter Da...	12/08/1981	5
8	C10002	Browne, William...	23/02/1962	0
9	C10002.1	Browne, Anne El...	14/09/1971	0
10	C10003	Devon Shoppe L...		0
11	C10004	Davis, Claire	12/12/1979	0
12	C10005	Weldon, Richard	16/02/1969	0
13	C10006	Browne, David...	19/06/1982	45
14	AB	AB Appliances L...		5
15	IH	In House Finance		0
16	TOP	Top Town Motors		0

And, when used as a Task Group Item in finPOWER Connect Cloud:

Tasks

All Tasks

My Logs

Review Dates

Clients

Clients

Type to filter list

	Client Id	Name	Date Of Birth	Status	
<input type="checkbox"/>	1	M	Main Branch	0	
<input type="checkbox"/>	2	HO	Head Office	0	
<input type="checkbox"/>	3	NR	Northern Region	0	
<input type="checkbox"/>	4	SR	Southern Region	0	
<input type="checkbox"/>	5	CR	Central Region	0	
<input type="checkbox"/>	6	C10000	Sample, Amelia Ingrid	29/08/1953	20
<input type="checkbox"/>	7	C10001	Morris, Peter David	12/08/1981	5
<input type="checkbox"/>	8	C10002	Browne, William Charles	23/02/1962	0
<input type="checkbox"/>	9	C10002.1	Browne, Anne Elizabeth	14/09/1971	0
<input type="checkbox"/>	10	C10003	Devon Shoppe Ltd		0
<input type="checkbox"/>	11	C10004	Davis, Claire	12/12/1979	0
<input type="checkbox"/>	12	C10005	Weldon, Richard	16/02/1969	0
<input type="checkbox"/>	13	C10006	Browne, David William	19/06/1982	45
<input type="checkbox"/>	14	AB	AB Appliances Ltd		5

NOTE: In both interfaces, the Client Id column provides a hyperlink to display the Client form. This is built-in functionality for recognised field names such as ClientId, AccountId etc.

The code does the following:

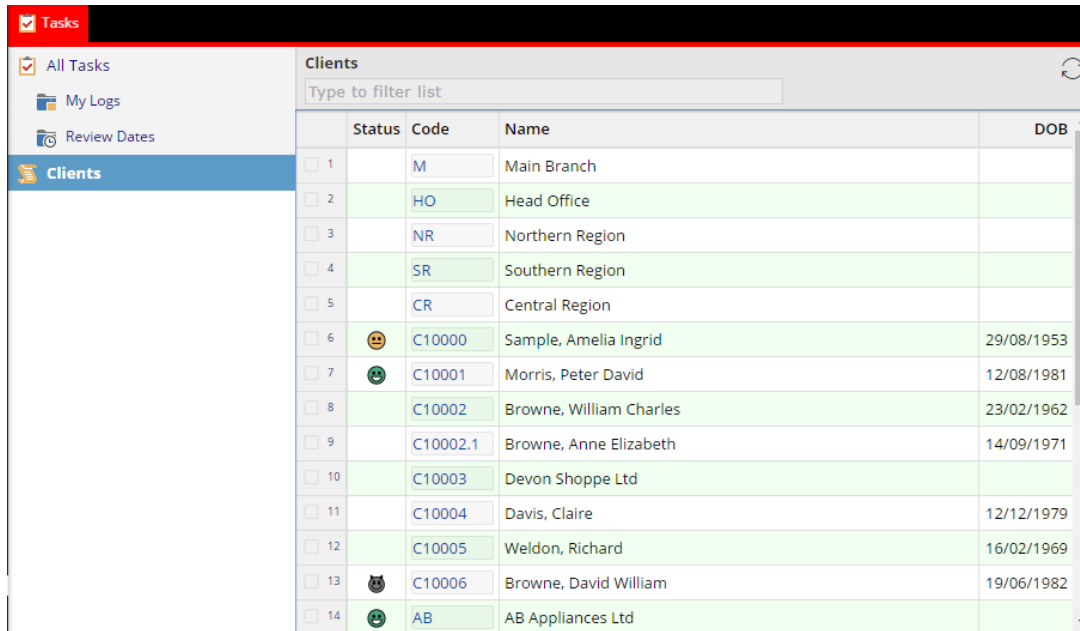
- Creates a Select Query Builder to get a list of all Clients.
- Executes the query and returns a `DataTable`.
- Sets Smart List Handler Results to this `DataTable`.

IMPORTANT: The highlighted code applies the current User's filters to the Client List; always include User Filters.

Customising Columns

As it stands, the Smart List displays all Clients and always shows the Client Id, Name and Date of Birth (in that order). It also displays the Client Status as a number which is not ideal.

By the end of this section, the Smart List will look like this when viewed in finPOWER Connect Cloud:



	Status	Code	Name	DOB
<input type="checkbox"/> 1		M	Main Branch	
<input type="checkbox"/> 2		HO	Head Office	
<input type="checkbox"/> 3		NR	Northern Region	
<input type="checkbox"/> 4		SR	Southern Region	
<input type="checkbox"/> 5		CR	Central Region	
<input type="checkbox"/> 6	😊	C10000	Sample, Amelia Ingrid	29/08/1953
<input type="checkbox"/> 7	😊	C10001	Morris, Peter David	12/08/1981
<input type="checkbox"/> 8		C10002	Browne, William Charles	23/02/1962
<input type="checkbox"/> 9		C10002.1	Browne, Anne Elizabeth	14/09/1971
<input type="checkbox"/> 10		C10003	Devon Shoppe Ltd	
<input type="checkbox"/> 11		C10004	Davis, Claire	12/12/1979
<input type="checkbox"/> 12		C10005	Weldon, Richard	16/02/1969
<input type="checkbox"/> 13	😺	C10006	Browne, David William	19/06/1982
<input type="checkbox"/> 14	😊	AB	AB Appliances Ltd	

You can allow the User to choose which columns to display (and in which order) and also control column captions and formatting by using the `Initialise()` method to define the Smart List columns.

Copy the following code and replace the `Initialise()` method.

```
Private Function Initialise() As Boolean

    Dim Success As Boolean

    ' Assume Success
    Success = True

    ' Define Columns
    With mSmartListHandler.Columns
        .AddClientStatus("Status")
        .AddCode("ClientId", "Code",,, "Clients")
        .AddString("Name", "Name")
        .AddDate("DateOfBirth", "DOB")
    End With

    Return Success

End Function
```

In finPOWER Connect Cloud, the Smart List now appears as:

Tasks				
<div> <div> <div>All Tasks</div> <div>My Logs</div> <div>Review Dates</div> <div>Clients</div> </div> <div> <div>Clients</div> </div> </div>				
Clients				
Type to filter list				
	Status	Code	Name	DOB
<input type="checkbox"/> 1		M	Main Branch	
<input type="checkbox"/> 2		HO	Head Office	
<input type="checkbox"/> 3		NR	Northern Region	
<input type="checkbox"/> 4		SR	Southern Region	
<input type="checkbox"/> 5		CR	Central Region	
<input type="checkbox"/> 6	😊	C10000	Sample, Amelia Ingrid	29/08/1953
<input type="checkbox"/> 7	😊	C10001	Morris, Peter David	12/08/1981
<input type="checkbox"/> 8		C10002	Browne, William Charles	23/02/1962
<input type="checkbox"/> 9		C10002.1	Browne, Anne Elizabeth	14/09/1971
<input type="checkbox"/> 10		C10003	Devon Shoppe Ltd	
<input type="checkbox"/> 11		C10004	Davis, Claire	12/12/1979
<input type="checkbox"/> 12		C10005	Weldon, Richard	16/02/1969
<input type="checkbox"/> 13	😺	C10006	Browne, David William	19/06/1982
<input type="checkbox"/> 14	😊	AB	AB Appliances Ltd	

Note the following:

- The columns now appear in the order that the Smart List Handler Columns are defined rather than the order they are retrieved in the Select Query.
- A special `AddClientStatus()` method indicates to the User Interface that the Client Status should be displayed as an icon rather a number.
- The "Date of Birth" column has a custom caption of "DOB".

And, when configuring the Task Group Item, columns can now be selected:

finPOWER Connect Cloud Connect Task Group Item

Columns

Specify Columns to include.

Specify Columns to include. ⓘ

DateOfBirth

ClientId

Name

Status

>

<

>>

<<

⬆

⬇

Cancel

< Back

Next >

Finish

	Code	Name	Status
<input type="checkbox"/> 1	M	Main Branch	
<input type="checkbox"/> 2	HO	Head Office	
<input type="checkbox"/> 3	NR	Northern Region	
<input type="checkbox"/> 4	SR	Southern Region	
<input type="checkbox"/> 5	CR	Central Region	
<input type="checkbox"/> 6	C10000	Sample, Amelia Ingrid	😞
<input type="checkbox"/> 7	C10001	Morris, Peter David	👤
<input type="checkbox"/> 8	C10002	Browne, William Charles	

Showing a Row Summary

Currently, the Smart List shows a simple grid of Clients. In this section we will add a Client summary for the selected row.

By the end of this section, the Smart List will look like this when viewed in finPOWER Connect Cloud:

Clients			
Type to filter list			
	Code	Name	Status
<input type="checkbox"/> 1	M	Main Branch	
<input type="checkbox"/> 2	HO	Head Office	
<input type="checkbox"/> 3	NR	Northern Region	
<input type="checkbox"/> 4	SR	Southern Region	
<input type="checkbox"/> 5	CR	Central Region	
<input type="checkbox"/> 6	C10000	Sample, Amelia Ingrid	😊
<input checked="" type="checkbox"/> 7	C10001	Morris, Peter David	😊
<input type="checkbox"/> 8	C10002	Browne, William Charles	
<input type="checkbox"/> 9	C10002.1	Browne, Anne Elizabeth	
<input type="checkbox"/> 10	C10003	Devon Shoppe Ltd	
<input type="checkbox"/> 11	C10004	Davis, Claire	
<input type="checkbox"/> 12	C10005	Weldon, Richard	
<input type="checkbox"/> 13	C10006	Browne, David William	🐱

Summary

Code: **C10001**

External Id:

Name: **Morris, Peter David**

Type: **I, Individual**

Date of Birth: **12/08/1981** 35 years old

Gender: **Male**

Occupation: **Account Manager**

Credit Rating:

Status: **Excellent** 😊

Contact Details

Home: **(08) 2254 1224**

Work: **(08) 2555 2215**

Mobile: **(021) 12312 3123** 📱

Email: **pmorris@biglake.com**

Accounts

We can easily use the built-in Client Summary page to provide a summary of the selected row.

Update the Execute() method to include the Pk column in the Select Query:

```
' Create Query
With sqb
    .Table = "Client"
    .Fields.AddList("Pk,ClientId,Name,DateOfBirth,Status")
```

Update the Initialise() method to always return the Pk column and also indicate that this Smart List now supports displaying a row summary:

```
' Define Columns
With mSmartListHandler.Columns
    .AddClientStatus("Status")
    .AddCode("ClientId", "Code",,, "Clients")
    .AddString("Name", "Name")
    .AddDate("DateOfBirth", "DOB")

    ' Hidden (will always be included in results)
    .AddHidden("Pk")
End With

' Summary Support
mSmartListHandler.SupportsSummary = True
mSmartListHandler.SummaryColumns = "Pk"

Return Success
```

Copy the following code and replace the GetRowSummary() method.

```
Private Sub GetRowSummary()

    Dim Pk As Integer
    Dim SummaryColumns As ISKeyValueList

    ' Get Event Args
    SummaryColumns = mEventArgs.GetKeyValueList("SummaryColumns")

    ' Get Summary Column Values that uniquely identify a record
    Pk = SummaryColumns.GetInteger("Pk")
```

```

' Get HTML
mSmartListHandler.RowSummary = finBL.SummaryPageFunctions.ClientSummary(False, Pk, Nothing,
ScriptInfo.Target)

End Sub

```

In finPOWER Connect Cloud, the Smart List now appears as:

Clients			
Type to filter list			
	Code	Name	Status
<input type="checkbox"/> 1	M	Main Branch	
<input type="checkbox"/> 2	HO	Head Office	
<input type="checkbox"/> 3	NR	Northern Region	
<input type="checkbox"/> 4	SR	Southern Region	
<input type="checkbox"/> 5	CR	Central Region	
<input type="checkbox"/> 6	C10000	Sample, Amelia Ingrid	😊
<input checked="" type="checkbox"/> 7	C10001	Morris, Peter David	😊
<input type="checkbox"/> 8	C10002	Browne, William Charles	
<input type="checkbox"/> 9	C10002.1	Browne, Anne Elizabeth	
<input type="checkbox"/> 10	C10003	Devon Shoppe Ltd	
<input type="checkbox"/> 11	C10004	Davis, Claire	
<input type="checkbox"/> 12	C10005	Weldon, Richard	
<input type="checkbox"/> 13	C10006	Browne, David William	🐱

Summary

Code: [C10001](#)

External Id:

Name: **Morris, Peter David**

Type: I, Individual

Date of Birth: **12/08/1981** 35 years old

Gender: **Male**

Occupation: **Account Manager**

Credit Rating:

Status: **Excellent** 😊

Contact Details

Home: [\(08\) 2254 1224](#)

Work: [\(08\) 2555 2215](#)

Mobile: [\(021\) 12312 3123](#) 📶

Email: pmorris@biglake.com

Accounts

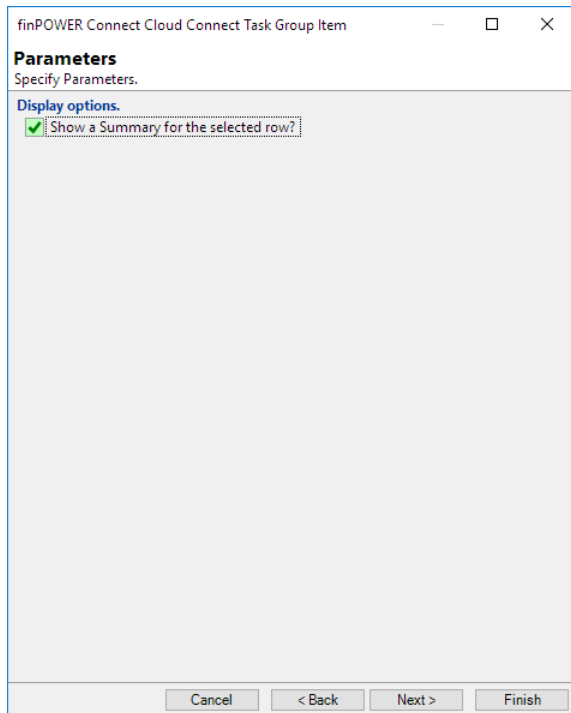
Note the following:

- We need to uniquely identify each row, therefore we always include the Pk column.
 - But this is meaningless to Users so we add it as a hidden column.
 - We could have used the ClientId column instead.
- In the `Initialise()` method, setting `mSmartListHandler.SummaryColumns = "Pk"` instructs the Smart List to always send the Pk fields value to the Script when requesting a Row Summary.
 - If we needed multiple fields, we can comma-separate them, e.g. "Pk,ClientId"
- The `GetRowSummary()` method is passed a special "SummaryColumns" parameter.
 - This allows us to determine the Pk of the row to summarise and pass this to the built-in `SummaryPageFunctions.ClientSummary()` method.

Making the Row Summary Optional

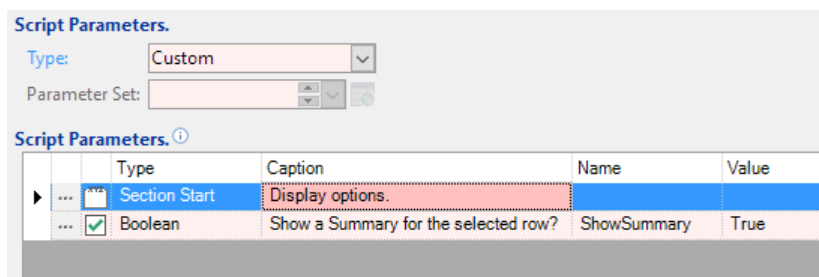
Currently, the row summary is always displayed.

In this section we'll see how to make this optional, i.e., the Smart List can be configured to show or hide the summary when configuring a Task Group Item:



Script Parameters are used to define options such as this.

- Select the Parameters page on the Scripts form
- Set "Type" to custom and add a "Section Start" and a "Boolean" Parameter as shown below:



Now, when configuring a Task Group Item, these new Parameters will be displayed as shown at the start of this section.

Update the `Execute()` method to use this new Parameter:

```
Dim dt As DataTable
Dim Parameters As ISKeyValueList
Dim sqb As ISSelectQueryBuilder
Dim Success As Boolean

' Assume Success
Success = True

' Initialise
Parameters = mSmartListHandler.ExecutionParameters
sqb = finBL.Database.CreateSelectQueryBuilder()

' Create Query
With sqb
    .Table = "Client"
    .Fields.AddList("Pk,ClientId,Name,DateOfBirth,Status")
```

```
' SQL Where
With .SqlWhere
    ' User Filters
    .Append(finBL.CurrentUserInformation.FilterClientSqlWhere)
End With
End With

' Execute
Success = sqb.ExecuteDataTable(dt)

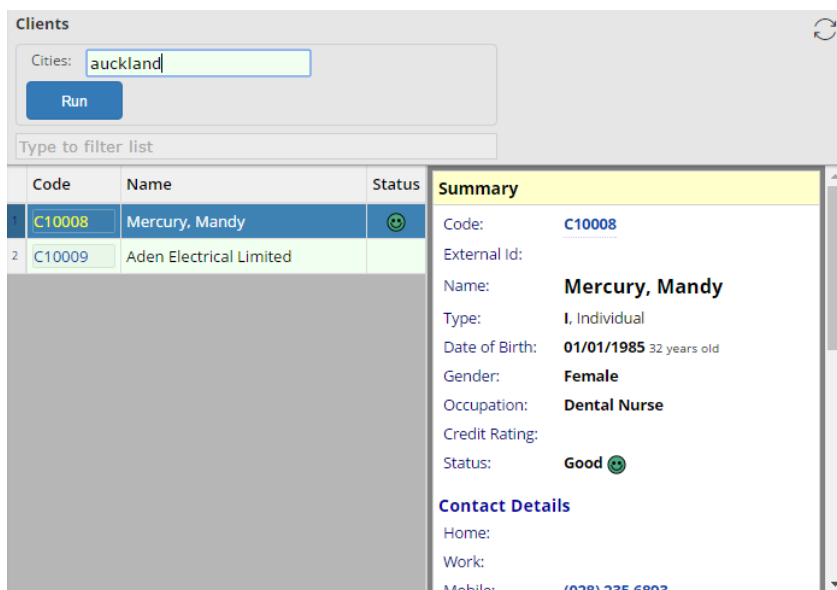
' Set Smart List Results
If Success Then
    mSmartListHandler.Results.DataSetFromDataTable(dt)
End If

' Displaying a Summary?
mSmartListHandler.SupportsSummary = Parameters.GetBoolean("ShowSummary")

Return Success
```

Allow Filtering by City

Parameters can also be used to add functionality such as filtering the list by City (either by displaying a field for the User to enter or by configuring a Task Group Item):



Again, Script Parameters are used to add functionality such as this.

- Add 2 new Parameters as shown below:

	Type	Caption	Name	Value
...	XVZ	Section Start	Display options.	
...	Boolean	Show a Summary for the selected row?	ShowSummary	True
...	XVZ	Section Start	Specify any Range filters you require.	
▶	abl	Text	Cities	CitiesRange

Update the Execute () method to use the new "CitiesRange" Parameter:

```
Dim CitiesRange As String
Dim dt As DataTable
Dim Parameters As ISKeyValueList
Dim sqb As ISSelectQueryBuilder
Dim sqbCities As ISSelectQueryBuilder
Dim Success As Boolean

' Assume Success
Success = True

' Initialise
Parameters = mSmartListHandler.ExecutionParameters
sqb = finBL.Database.CreateSelectQueryBuilder()
sqbCities = finBL.Database.CreateSelectQueryBuilder()

' Get Parameters
With Parameters
    CitiesRange = .GetString("CitiesRange")
End With

' Create Query
With sqb
    .Table = "Client"
    .Fields.AddList("Pk,ClientId,Name,DateOfBirth,Status")

' SQL Where
With .SqlWhere
    If Len(CitiesRange) = 0 Then
        ' Return no records if not filtered
        .AppendComparisonInteger("Pk", "=", -1)
    Else
        ' Cities (include Suburb)
        With sqbCities
            .Table = "ClientContactAddress"
```

```

.Fields.Add("ClientPk")

With .SqlWhere
    .AppendComparisonField("ClientContactAddress.ClientPk", "=", "Client.Pk")

    ' City/ Suburb
    .BlockBegin(iseSqlWhereBuilderNestedBlockType.OrBlock)
    .AppendRange("City", CitiesRange)
    .AppendRange("Suburb", CitiesRange)
    .BlockEnd()

    ' Current Addresses Only
    .AppendComparisonIntegerBoolean("Historic", "=", False)
    .BlockBegin(iseSqlWhereBuilderNestedBlockType.OrBlock)
    .AppendComparisonNull("DateStop")
    .AppendComparisonDate("DateStop", "<",
finBL.TimeZoneFunctions.GetCurrentDatabaseDate().AddDays(1), False)
    .BlockEnd()
End With
End With

.AppendInSubQuery("Pk", sqbCities)
End If

' User Filters
.Append(finBL.CurrentUserInformation.FilterClientSqlWhere)
End With

' Order By
.OrderByFields.Add("ClientId")
End With

' Execute
Success = sqb.ExecuteDataTable(dt)

' Set Smart List Results
If Success Then
    mSmartListHandler.Results.DataSetFromDataTable(dt)
End If

' Displaying a Summary?
mSmartListHandler.SupportsSummary = Parameters.GetBoolean("ShowSummary")

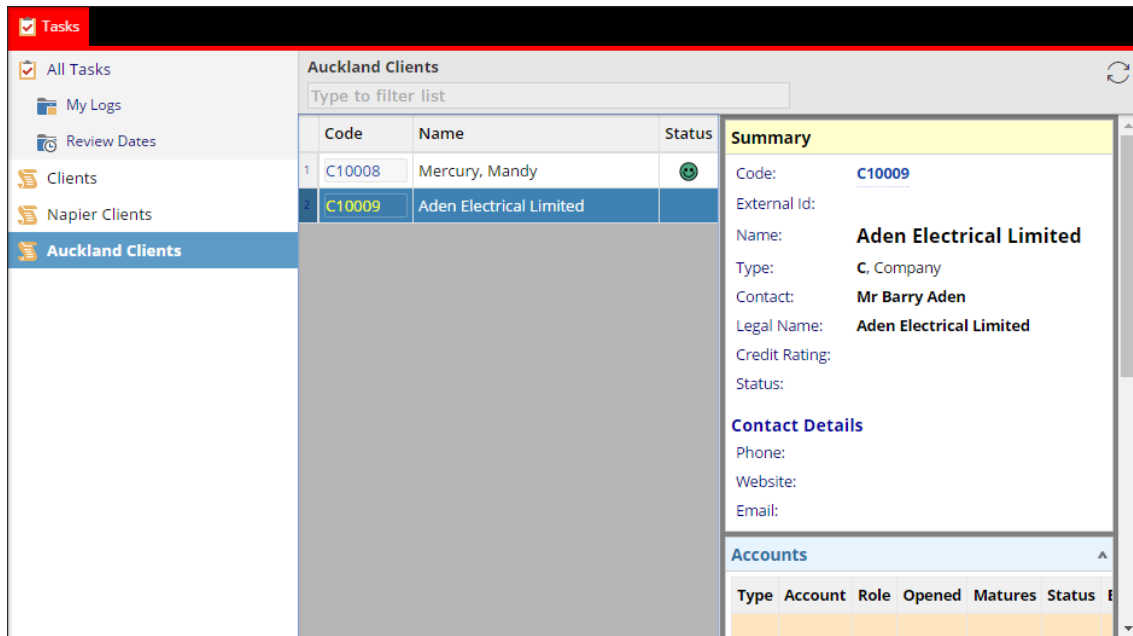
Return Success

```

Note the following:

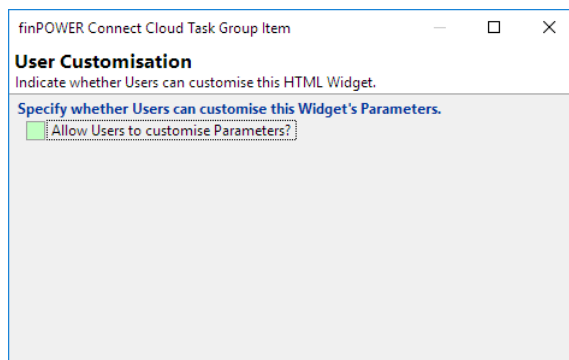
- The "CitiesRange" Parameter is used to build a sub-query to restrict returned records.
 - If this Parameter is blank, an empty Data Table is returned.
- The Select Query has also been updated so that the results are now ordered by ClientId.

When configuring a Task Group Item to use this Smart List, by default you can enter the Cities Parameter on a per-item basis, i.e., not allow the Client to enter it. This allows you to produce custom lists, e.g.:

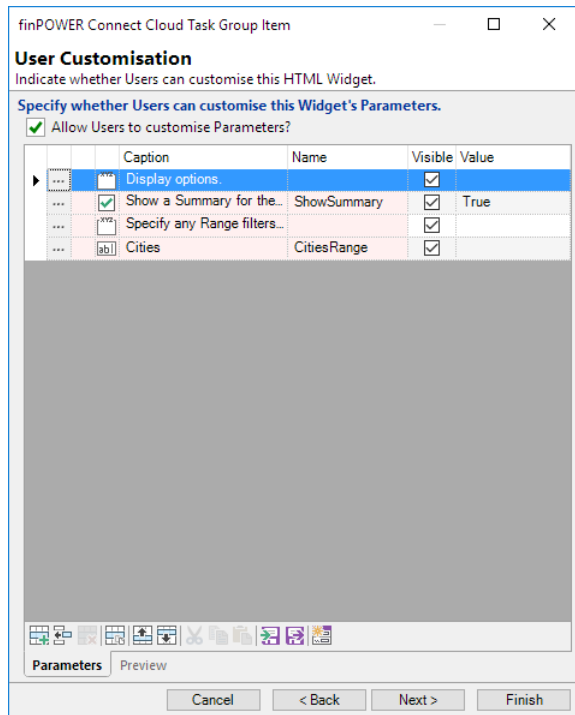


To allow the Client to enter a City (as per the screenshot at the start of this section), you need to do the following:

- From the finPOWER Connect Cloud Configuration form, Tasks group, Task Groups page, edit the Smart List item:
 - You can either drilldown the Task Group to locate the item or click the link in the summary in the "Task Group Items" section.
- Move to the "User Customisation" page of the finPOWER Connect Cloud Task Group Item wizard:

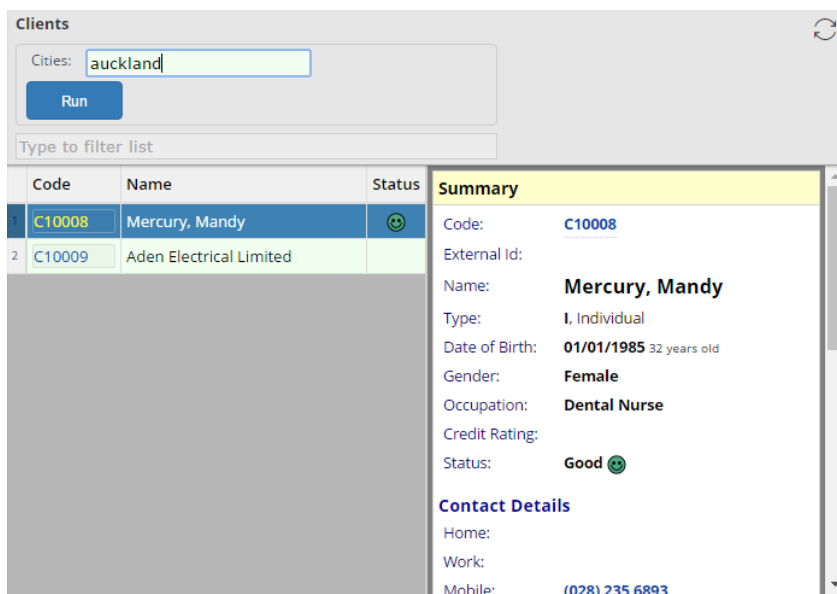


- Check the "Allow Users to customise Parameters?" box.
- Click the "Add Standard Parameters" button (🔧) to add all Parameters defined on the Smart List Script:



- Remove all Parameters except "CitiesRange" since this is the only one you want to allow Users to enter.

The Smart List will now allow the User to filter by City (or a range of Cities):



Adding a "Send Email" Action

Actions can be added below the results grid, e.g., to send a Client an Email:

The screenshot shows a web application interface for 'Napier Clients'. It features a table with columns for Code, Name, and Status. A client record for 'Sample, Amelia Ingrid' with Code 'C10000' is selected. To the right of the table, a detailed view of the client is shown, including their name, type, date of birth, gender, occupation, credit rating, and status. At the bottom left of the record, there is an '@' icon representing the 'Send Email' action.

When clicked, this Action will open the "Send Email Message" wizard for the Client, allowing the User to easily enter and send an Email:

The screenshot shows a 'Send Email Message' wizard. It has a sidebar with navigation options like 'Tasks', 'My Logs', 'Review Dates', 'Clients', 'Napier Clients', and 'Auckland Clients'. The main area is divided into sections: 'Email' (Enter Email details), 'Specify Recipient details' (Client: C10000, Name: Sample, Amelia Ingrid, Email: dropdown, CC: text box, BCC: text box), and 'Enter the Email Subject and Message' (Test Email from Smart List, text area). At the bottom, there are buttons for 'Cancel', '< Back', 'Next >', and 'Send'.

Firstly, all Forms are opened using the record's Id (e.g., C10000) and NOT the Integer primary key. Therefore, we need to update the `Initialise()` method to always include the `ClientId` column, regardless of whether it is being displayed or not:

```
' Define Columns
With mSmartListHandler.Columns
  .AddClientStatus("Status")
  .AddCode("ClientId", "Code",,,, "Clients").AlwaysInclude = True
  .AddString("Name", "Name")
  .AddDate("DateOfBirth", "DOB")

  ' Hidden (will always be included in results)
  .AddHidden("Pk")
End With
```

Next, we need to define an Action that will open the Send Email Message wizard so add the following to the bottom of the `Initialise()` method:

```
' Summary Support
mSmartListHandler.SupportsSummary = True
mSmartListHandler.SummaryColumns = "Pk"

' Define Actions
With mSmartListHandler.Actions
    .AddRowFormOpen("SendEmail", "EmailMessage", "ClientId", "type=Client&subject=Test Email from Smart List", "Email", "", "Open the Send Email Message wizard")
End With

Return Success
```

Note the following:

- The `finSmartListHandler.Actions.AddRowFormOpen()` method takes the following parameters:

- `actionId`
- `formKey`

- ✧ The Form Key that identifies the Send Email Message wizard.
- ✧ This can be seen as part of the URL, e.g., when showing the Send Email Message wizard from an action on the Client form, the URL is as follows:

```
/finC/?V?id=Records/Main_Tasks?page=TASKS&item=Test2/Records?list=Client.C10000&sel=&sbs=n&p=/EmailMessage?type=Client&id=C10000
```

- `id`
 - ✧ The field name of the record id.
- `parametersUrl`
 - ✧ Any other URL-encoded Parameters to send to the form being opened.
- `icon`
 - ✧ The Icon for the Action button.
- `caption`
 - ✧ The Caption text for the Action button. In this case, we have left it blank so only the icon shows.
- `description`
 - ✧ Appears as a Tooltip on the Action button.

NOTE: Because this Action simply opens a form (the Send Email Message wizard) in the User Interface, there is nothing else we need to add to the Script.

Adding a "Request Address Update" Action

The previous Action simply opened the Send Email Message wizard in the User Interface.

We will now add an Action that is processed by the Smart List Script and will send an Email automatically to the Client to confirm their existing address information:

Auckland Clients			
Type to filter list			
	Code	Name	Status
✓ 1	C10008	Mercury, Mandy	😊
✓	C10009	Aden Electrical Limited	

@ @ Confirm Address

When clicked, this Action will automatically send an Email to the selected Client (or Clients) detailing their currently recorded address.

Firstly, we need to define a custom Action so add the following to the `Initialise()` method:

```
' Summary Support
mSmartListHandler.SupportsSummary = True
mSmartListHandler.SummaryColumns = "Pk"

' Define Actions
With mSmartListHandler.Actions
    .AddRowFormOpen("SendEmail", "EmailMessage", "ClientId", "type=Client&subject=Test Email from Smart List", "Email", "", "Open the Send Email Message wizard")
    .AddRowCustom("RequestAddressUpdate", True, True, True, "", "Email", "Confirm Address", "Send an Email for Client to confirm their Address details")
End With

Return Success
```

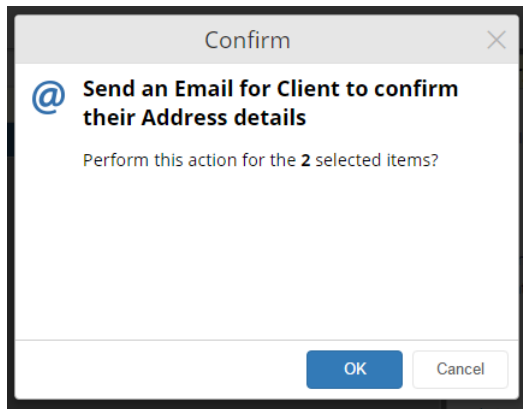
Note the following:

- The `finSmartListHandler.Actions.AddRowCustom()` method takes the following parameters:
 - `actionId`
 - `allowMultiple`
 - ✧ Indicates whether this action will be available if multiple rows are selected. In this case we have set to True.
 - `prompt`
 - ✧ This is set to True so the User will be prompted before the action is executed. Even if only a single row is selected.
 - `promptIfMultiple`
 - ✧ This is also set to True.

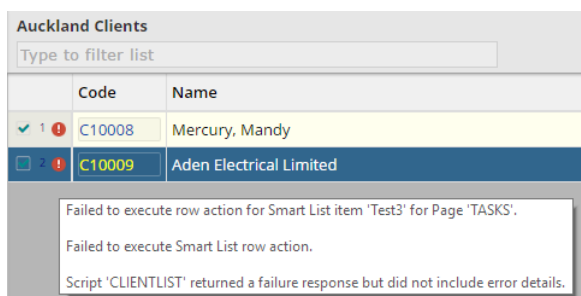
✧ **NOTE:** This is separate from the 'prompt' parameter since it may be desirable to NOT prompt unless multiple rows are selected.

- parametersUrl
- icon
- caption
- description

Now, if several grid rows are selected and this Action is clicked, a prompt will be displayed:



Upon clicking "OK", each of the grid rows will display an error icon with a tooltip detailing the error:



This is because we are not handling the new Action in the Script.

Replace the `ExecuteRowAction()` method with the following:

```
Private Function ExecuteRowAction() As Boolean

    Dim Action As finSmartListAction
    Dim ActionId As String
    Dim Parameters As ISKeyValueList
    Dim Success As Boolean
    Dim SummaryColumns As ISKeyValueList
    Dim Pk As Integer

    ' Assume Success
    Success = True

    ' Get Event Args
    With mEventArgs
        ActionId = .GetString("ActionId")
        Parameters = .GetKeyValueList("Parameters")
        SummaryColumns = mEventArgs.GetKeyValueList("SummaryColumns")
    End With

    ' Get Summary Column Values that uniquely identify a record
    Pk = SummaryColumns.GetInteger("Pk")

    ' Get Action
    If mSmartListHandler.Actions.Exists(ActionId) Then
        Action = mSmartListHandler.Actions(ActionId)
```

```

Else
    Success = False
    finBL.Error.ErrorBeginFormat("Action '{0}' not found.", ActionId)
End If

' Execute Action
If Success Then
    Select Case Action.ActionId
        Case "RequestAddressUpdate"
            Success = ExecuteRowAction_RequestAddressUpdate(Pk)

        Case Else
            Success = False
            finBL.Error.ErrorBeginFormat("Action '{0}' not handled.", Action.ActionId)
    End Select
End If

Return Success
End Function

```

And, add the following function to the bottom of the Script:

```

Private Function ExecuteRowAction_RequestAddressUpdate(clientPk As Integer) As Boolean

    Dim Client As finClient
    Dim CurrentAddress As ISAddressDetails
    Dim EmailAddress As String
    Dim Message As String
    Dim Recipients As finMessageRecipients
    Dim sb As StringBuilder
    Dim Subject As String
    Dim Success As Boolean

    ' Assume Success
    Success = True

    ' Load Client
    Client = finBL.CreateClient()
    Success = Client.LoadPk(clientPk)

    ' Locate current Email
    If Success Then
        EmailAddress = Client.ContactMethods.GetCurrentEmail().Value

        ' Validate
        If Len(EmailAddress) = 0 Then
            Success = False
            finBL.Error.ErrorBegin("Client does not have a current Email address.")
        End If
    End If

    ' Locate current Physical Address
    If Success Then
        CurrentAddress = Client.ContactAddresses.GetCurrentPostalAddress()
    End If

    ' Build Email message
    If Success Then
        Subject = "Address Confirmation"

        sb = New StringBuilder()
        With sb
            .AppendLine(String.Format("Dear {0}", Client.SalutationResolved))
            .AppendLine()
            If CurrentAddress.HasValue Then
                .AppendLine("We have the following postal address recorded for you:")
                .AppendLine()
                .AppendLine(finBL.Addressing.FormatAddressFromAddressDetails(CurrentAddress))
                .AppendLine()
                .AppendLine("If your address has changed, please could you email your new address details so we can update our records.")
            Else
                .AppendLine("We do not have a current postal address recorded for you.")
                .AppendLine("Please could you email your address details so we can update our records.")
            End If

            .AppendLine()
            .AppendLine("Regards,")
        End With
    End If
End Function

```

```

        .AppendLine()
        .AppendLine(finBL.CurrentUserInformation.UserName)
    End With

    Message = sb.ToString()
End If

' Send Email
' NOTE: This also generates a Client Log
If Success Then
    Recipients = finBL.CreateMessageRecipients()
    With Recipients
        ' Initialise
        .AddClient(Client.ClientId, Client.Name, EmailAddress)
        .GenerateMessages(Message, finBL.TimeZoneFunctions.GetCurrentDatabaseDate(), Subject)

        ' Send (append signature and use standard HTML Email template)
        Success = .SendEmail(iseMessageTarget.Send, Nothing, True, "", "*", "")
    End With
End If

Return Success
End Function

```

Now, running this Action sends an Email and produces the following (providing the Client has been give an Email address and the SMTP Server details in finPOWER Connect are configured correctly):

The screenshot shows the 'Napier Clients' application interface. On the left, a table lists clients. The client 'Sample, Amelia Ingrid' with code 'C10000' is selected and highlighted with a cross-hatch pattern, indicating it is 'expired'. The right pane displays a detailed summary for this client, including their name, type (Individual), date of birth (29/08/1953), gender (Female), occupation (Nurse), credit rating, and status (Caution). It also lists contact details (Home, Work, Mobile, Email) and accounts.

The problem now is that the row for which the Email has been sent has been flagged as "expired", hence the cross-hatching and the fact no more Actions can be performed on it without refreshing the Smart List.

"Expiring" a row is the default when running an Action since it is assumed that the Action is updating the record, e.g., setting the Flag Colour of a Workflow.

However, in this case, the record is not being affected; the Action is simply sending an Email.

Adding the following to the end of the `ExecuteRowAction_RequestAddressUpdate()` method solves this issue:

```

' Do not expire row
mSmartListHandler.ResultsRowAction.Expired = False

Return Success

```

End Function

File Uploads

Uploading Files is also possible via Smart Lists actions.

For example, your Smart List may list un-actioned Account Logs that require a scanned document to be attached.

In this case, you may wish to provide an "Upload File" Action for the selected Row.

The following partial example demonstrates how to achieve this:

```
Private Function Initialise() As Boolean

    Dim Success As Boolean

    ' Assume Success
    Success = True

    ' Define Columns
    With mSmartListHandler.Columns
        ' Hidden (always included in results, e.g., for drilldowns and summaries)
        .AddHidden("Pk")
    End With

    ' Define Actions
    With mSmartListHandler.Actions
        .AddRowCustomFileUpload("UploadFile")
    End With

    ' Summary Support
    ' NOTE: Summary Columns also provide the primary key for row-based Actions
    mSmartListHandler.SupportsSummary = True
    mSmartListHandler.SummaryColumns = "Pk"

    Return Success
End Function

Private Function ExecuteRowAction() As Boolean

    Dim AccountLog As finAccountLog
    Dim Action As finSmartListAction
    Dim ActionId As String
    Dim Parameters As ISKeyValueList
    Dim Pk As Integer
    Dim Success As Boolean
    Dim SummaryColumns As ISKeyValueList

    ' Assume Success
    Success = True

    ' Get Event Args
    With mEventArgs
        ActionId = .GetString("ActionId")
        Parameters = .GetKeyValueList("Parameters")
        SummaryColumns = mEventArgs.GetKeyValueList("SummaryColumns")
    End With

    ' Get Summary Column Values that uniquely identify a record
    Pk = SummaryColumns.GetInteger("Pk")

    ' Get Action
    If mSmartListHandler.Actions.Exists(ActionId) Then
        Action = mSmartListHandler.Actions(ActionId)
    Else
        Success = False
        finBL.Error.ErrorBeginFormat("Action '{0}' not found.", ActionId)
    End If

    ' Load Log
    If Success Then
        AccountLog = finBL.CreateAccountLog()
        Success = AccountLog.Load(Pk)
    End If

    ' Execute Action
    If Success Then
        Select Case Action.ActionType
            Case isefinSmartListActionType.RowCustom
                ' Custom
```

```

        Select Case Action.ActionId
            Case "UploadFile"
                ' Upload File
                Success = ExecuteRowAction_UploadFile(AccountLog)

            Case Else
                Success = False
                finBL.Error.ErrorBeginFormat("Action '{0}' not handled.", Action.ActionId)
            End Select

        Case Else
            Success = False
            finBL.Error.ErrorBeginFormat("Action '{0}' not handled.", Action.ActionId)
        End Select
    End If

    Return Success
End Function

Private Function ExecuteRowAction_UploadFile(accountLog As finAccountLog) As Boolean

    Dim Account As finAccount
    Dim AddedAsFileName As String
    Dim Success As Boolean

    ' Assume Success
    Success = True

    ' Validate
    If mScriptRequestInfo.Files.Count = 0 Then
        Success = False
        finBL.Error.ErrorBegin("No file specified.")
    End If

    ' Load Account
    If Success Then
        Account = finBL.CreateAccount()
        Success = Account.LoadPk(accountLog.AccountPk)
    End If

    ' Upload File
    If Success Then
        Success = Account.DocumentFiles.Add(mScriptRequestInfo.Files(0).FileData,
                                            "",
                                            mScriptRequestInfo.Files(0).FileName,
                                            AddedAsFileName,
                                            True)
    End If

    ' Update Log
    If Success Then
        With AccountLog
            .DocumentFileName = AddedAsFileName
            .ActionCompleteUtcDate = finBL.TimeZoneFunctions.GetCurrentUtcDateTime()

            Success = .Save()
        End With
    End If

    Return Success
End Function

```

This provides the following Action for a Smart List Row:

<input type="checkbox"/>	AccountId	Date	Subject
<input type="checkbox"/> 1	L10035	10/02/2017 04:36	Upload Document Client Photo

Notification Actions

When an action is performed on a row, the row is automatically flagged as being "Invalid" or "Expired", i.e., the information it is displaying may not be up-to-date.

Smart Lists can also respond to Notification Actions to invalidate rows, e.g., if a Client Log is edited, you may wish for any Smart List rows referencing that Log to become invalid.

Built-in Smart Lists such as Tasks and Workflows contain examples of how to achieve this.

Targeting a Specific User and Date As At

User

Prior to version 3.00.00 of finPOWER Connect, Smart Lists (as shown in the Tasks view) were always intended to deal with the currently logged in User.

However, an update to finPOWER Connect Cloud allows other Users' Tasks Views to be viewed within finPOWER Connect. The available Users that can be viewed is as per the Users page of the Task Manager form within finPOWER Connect, e.g.:

Code	Name
ADMIN	Administrator
AB-DB1	Blakey, Dan
GUEST	Guest User
USER	General User
WEBUSER	Web User

Code: AB-DB1
Name: Blakey, Dan
Job Title: Sales Rep
User Level: Normal
Log Email

Responsibilities

- ✓ Business Development Manager
- ✓ Can Allocate Account Applications
- ✓ Can Authorise new Accounts

External User

Dealer: AB, AB Appliances Ltd
Employee: AB-DB1, Blakey, Dan

OK Cancel

The `finSmartListHandler` object can be initialised with a `UserId` property that can be used by Smart List Scripts to target a specific User.

This can then be accounted for when querying the database to supply a different SQL Where clause based on the User, e.g.:

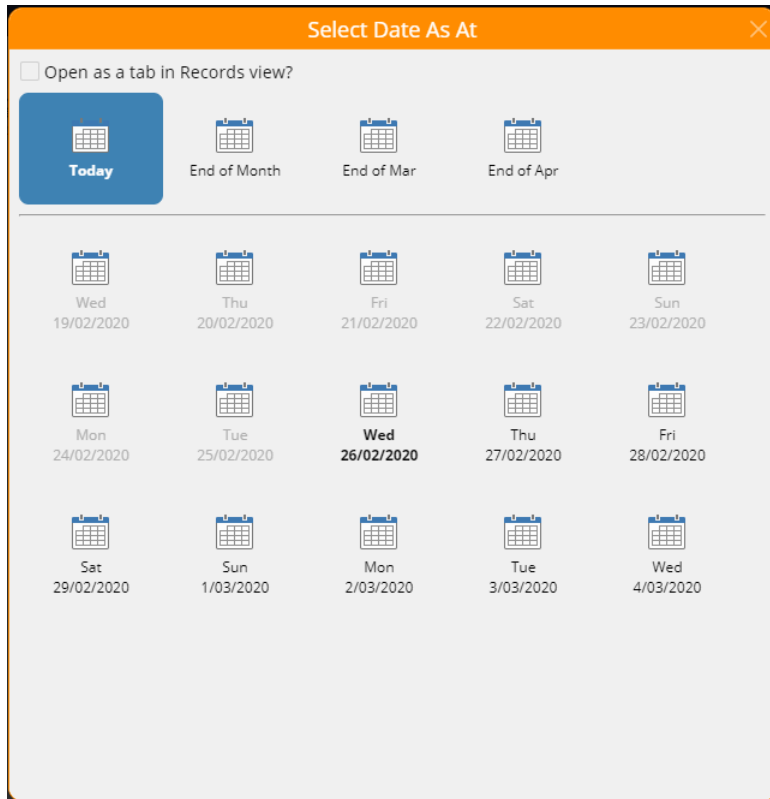
```
' Multi-User Support (i.e., Allow Tasks to be viewed for a different User)
If mSmartListHandler.UserId = finBL.CurrentUserInformation.UserId Then
    FilterAccountSqlWhere = finBL.CurrentUserInformation.FilterAccountSqlWhere
Else
    ' Viewing for another User
    If finBL.Users(mSmartListHandler.UserId).GetCurrentUserInformation(CurrentUserInformation) Then
        FilterAccountSqlWhere = CurrentUserInformation.FilterAccountSqlWhere
    Else
        ' ERROR
    End If
End If
```

WARNING: The `finAccount` object will still fail to load if the currently signed in User does not have permission to view the Account, regardless of the User that tasks are being shown for.

Date As At

Prior to version 3.03.00 of finPOWER Connect, Smart Lists (as shown in the Tasks view) were always intended to deal with the current date (in the Database Time Zone).

The Tasks view in finPOWER Connect Cloud now allows user's to select the 'Date As At' to use, e.g.:



The `finSmartListHandler` object can be initialised with a `DateAsAt` property that can be used by Smart List Scripts, e.g., when querying the database:

```
' Date As At Support
If mSmartListHandler.DateAsAt <> Nothing Then
    Sqb.SqlWhere.AppendComparisonDate("AccountLog.ActionDate", "<=", mSmartListHandler.DateAsAt)
End If
```

Row Actions

Form Open

Forms with an id Parameter

Many forms take an id parameter to represent the record id, e.g., to show the "EmailMessage" form for a Client, an action can be added as follows:

```
.AddRowFormOpen("SendEmail",  
"EmailMessage",  
"ClientId",  
"type=Client&subject=Test Email from Smart List",  
"Email",  
"",  
"Open the Send Email Message wizard")
```

This tells the action to pass the content of the "ClientId" field as the id parameter when opening the form.

Forms without an id Parameter

Many forms, typically HTML Widgets, do not accept an id parameter.

Instead, the Widget may accept a parameter such as `accountId`.

In these cases, a replaceable tag should be used, e.g.:

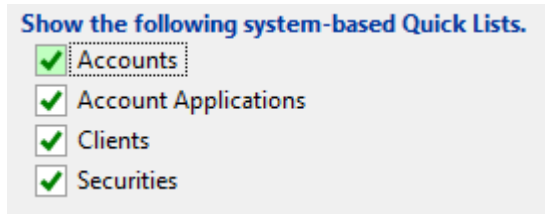
```
.AddRowFormOpen("PendingWithdrawal",  
"AccountPendingWithdrawal",  
"AccountId",  
"accountId=[AccountId]&showHistory=false",  
"Withdrawal",  
"Drawdown",  
"Make a request for an Additional Drawdown")
```

This tells the action to replace the "[AccountId]" tag with the content of the "AccountId" field as the id parameter when opening the form.

Quick Lists

Quick Lists are a special type of Smart List which are displayed on the "Quick Lists" page of the "Search view" in finPOWER Connect Cloud to present filtered lists to the User.

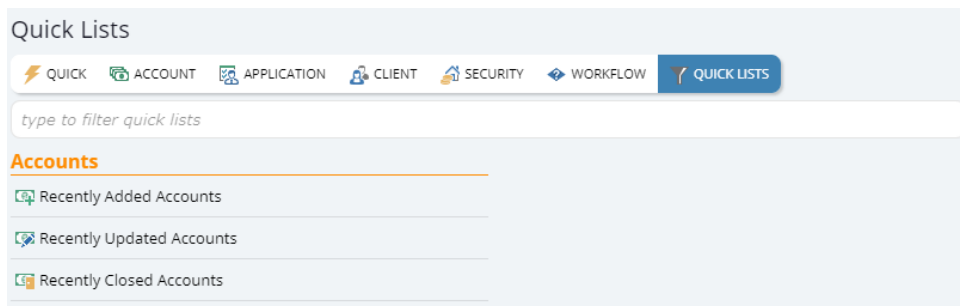
Under the finPOWER Connect Configuration form, Search group, Quick Lists page, there is the option to include various system-defined Quick Lists:



Show the following system-based Quick Lists.

- ☒ Accounts
- ☒ Account Applications
- ☒ Clients
- ☒ Securities

Checking one of these options means that one or more entries will be included on the Quick Lists page, e.g.:



Quick Lists

QUICK ACCOUNT APPLICATION CLIENT SECURITY WORKFLOW QUICK LISTS

type to filter quick lists

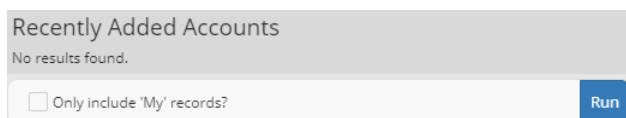
Accounts

- Recently Added Accounts
- Recently Updated Accounts
- Recently Closed Accounts

NOTE: The three entries above are the result of checking the "Accounts" box which uses the system supplied `SmartList_QuickList_Accounts` Smart List.

This Smart List has a special parameter named "QuickListType". This allows the Script to handle multiple Quick Lists.

Quick Lists will only display a single Parameter named "MyRecords" (if the list defines this), e.g.:



Recently Added Accounts

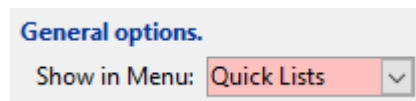
No results found.

☐ Only include 'My' records?

Run

Custom Quick Lists

Smart List Scripts can be configured to be used as Quick Lists by selecting the "Show in Menu" option on the Options page of the Scripts form:



General options.

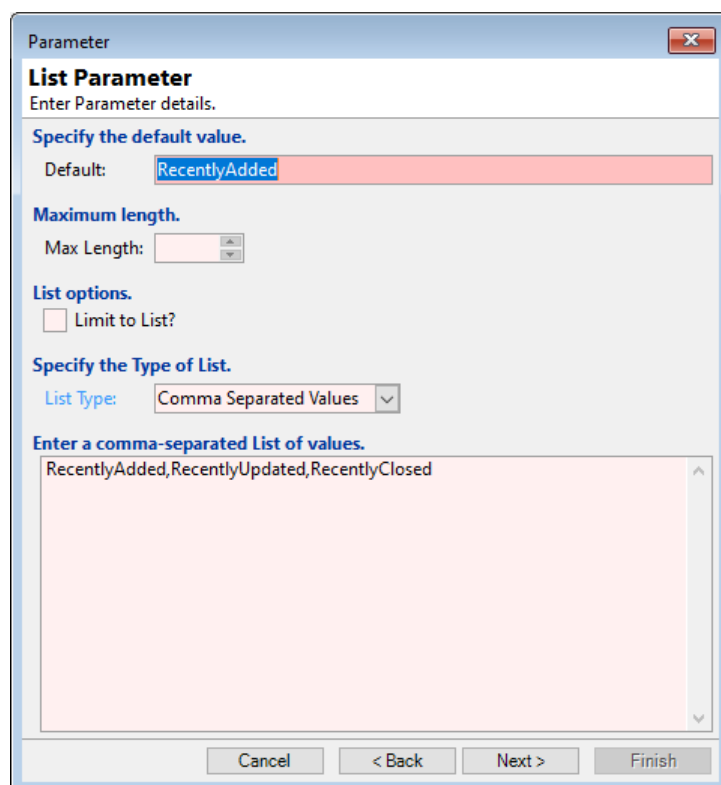
Show in Menu: Quick Lists

The Script's Category as defined on the General page of the Scripts form determines how Quick Lists are grouped.

Handling Multiple Lists

Each of the system-supplied Quick Lists can handle multiple lists.

They do this by providing a special parameter named "QuickListType". This parameter is never shown to the User, e.g.:



Parameter

List Parameter
Enter Parameter details.

Specify the default value.
Default: RecentlyAdded

Maximum length.
Max Length:

List options.
☐ Limit to List?

Specify the Type of List.
List Type: Comma Separated Values

Enter a comma-separated List of values.
RecentlyAdded,RecentlyUpdated,RecentlyClosed

Cancel < Back Next > Finish

They also defined a special parameter named "Configuration" which is a tilde-separated list of entries for each option that the User will see. This is in the format:

Icon~Caption~QuickListType

For example, the system supplied Accounts Quick List defines the following configuration options:

Parameter

List Parameter

Enter Parameter details.

Specify the default value.

Default:

Maximum length.

Max Length:

List options.

☐ Limit to List?

Specify the Type of List.

List Type:

Enter a comma-separated List of values.

Account|Add~Recently Added Accounts|RecentlyAdded
Account|Edit~Recently Added Accounts|RecentlyUpdated
Account|Closed~Recently Added Accounts|RecentlyClosed

Cancel < Back Next > Finish

NOTE: The icon "Account|Add" indicates that the "Account" icon should be shown with the "Add" overlay (a green plus symbol).