finPOWER Connect Word Document and Excel Workbook Processes and Scripting Guide

Version 1.25

06 September 2023

Documents/Development/finPOWER Connect/Version 4/Documents/Word Document and Excel Workbook Processing and Scripting Guide.docx

Contents

Introduction

This document discusses finPOWER Connect Documents and in particular the Documents with the "Word Document" file type introduced in finPOWER Connect version 3.

This Document focuses on "Word Document" file types but the majority of the code can also be used for Documents with an "Excel Workbook" file type.

"Word Document" type Documents will eventually replace most existing Word VBA type Documents and in the context of this Document, refer to finPOWER Connect Documents that have a file type of "Word Document" rather than actual MS Word Documents (.docx files).

NOTE: Although these Documents are referred to as "Word Documents", in actual fact the MS Word Document is just the base (or intermediate) format and all of the supplied Documents actually produce PDF files.

Word Documents Overview

Word Documents have many advantages over Microsoft Word VBA Documents:

- They do no rely on Microsoft Word being installed on a User's PC in order to run.
- They are much faster than using Word VBA.
    - In many cases, hundreds of times faster.
- They use finPOWER Connect Scripting which is full VB.NET rather than VBA.
- They can work from an existing MS Word Document (.docx) as a template.
    - This allows existing Word VBA templates to be used and the VBA code modules removed.
- The .docx file can be embedded within the Document record in the finPOWER Connect database which means that the Document does not require access to the file system (and in particular, the Templates folder) in order to be used.
    - This is highly beneficial in a Web environment where the Web Server may not have access to the Templates folder.
- A PDF file can be produced instead of a Word Document.
    - All supplied Templates will generate a PDF rather than a Word file.
- They do not rely on the finPOWER Connect User Interface to be published, i.e., they can be published "Unattended" like Emails and SMS messages.
    - This is highly beneficial in a Web environment where the finPOWER Connect Windows User Interface is not available.

WARNING: finPOWER Connect uses an external component called GemBox to produce Word Documents and PDF files.

GemBox may have formatting issues that are outside of Intersoft's control and therefore, by working with "Word Document" and "Excel Workbook" file types, any formatting functionality relies on GemBox's implementation being correct.

Also, whenever the version of GemBox used by finPOWER Connect is updated, it is recommended that all applicable Documents are tested for potential, newly introduced, formatting issues.

Importing a Supplied Word Document

Prior to version 3 of finPOWER Connect, all supplied Documents were MS Word VBA templates, i.e., .dot files in the /Templates folder. Setting up one of these as a finPOWER Connect Document involved adding a new record to the Documents admin library and configuring all or the document options correctly.

With version 3, the /Templates folder now contains .xml files for each of the supported documents. These XML files can now be imported via the "Import" action on the Documents form.

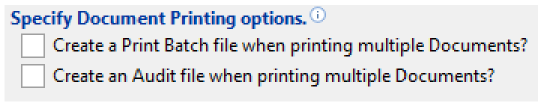
The XML file contains all configuration options of the Document and also has the MS Word Document (.docx) file embedded.

NOTE: The /Templates folder also contains the non-embedded .docx file for reference. Out of the box, this should match the embedded file in the Document.

The Documents Form section includes more information on the configuration options applicable to Word Documents.

Global Settings

The Global Settings form contains a section on the General, Documents and Templates page that relates to Document Printing:



Specify Document Printing options.

☐ Create a Print Batch file when printing multiple Documents?

☐ Create an Audit file when printing multiple Documents?

NOTE: When Scripting, these values can be overridden on the `finDocumentsPrint` object.

These settings are used by the Documents Print wizard to decide whether to create Batch files in the Document Manager '(documents)' folder when multiple items are printed.

These files can be used to reprint a batch or, in the case of the 'Audit' file, review the reason that certain items failed to process.

Permission Keys

The following Permission Keys exist to control Document Printing:

- DocumentsPrint.AllowDraftPrint
  - This is "Deny" by default and will show/ hide the "Draft Print?" checkbox on the Print Documents wizard.

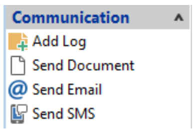
Publishing and Printing

NOTE: This section applies to Documents that are being generated via a Log, e.g., a Client or Account Log.

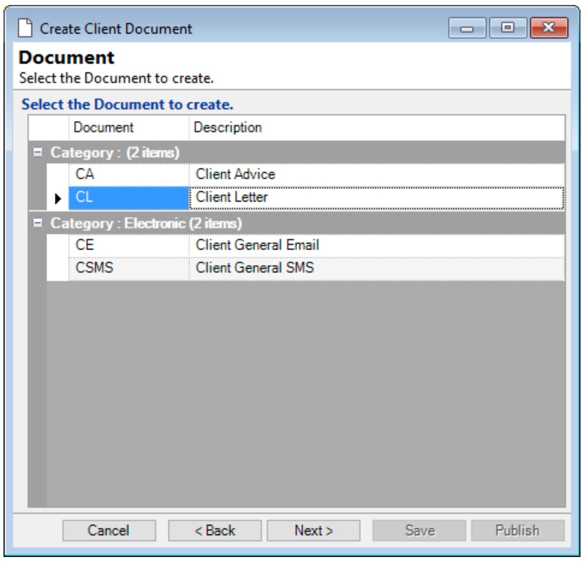
Creating, Publishing and Printing a Document involves up to 3 separate steps which are detailed in this section.

Creating a Document Log

A Document Log is typically created by the "Send Document" action on a form such as Clients or Accounts, e.g.:



This launches the Create Document wizard from which a Document can be selected, e.g.:


Create Client Document

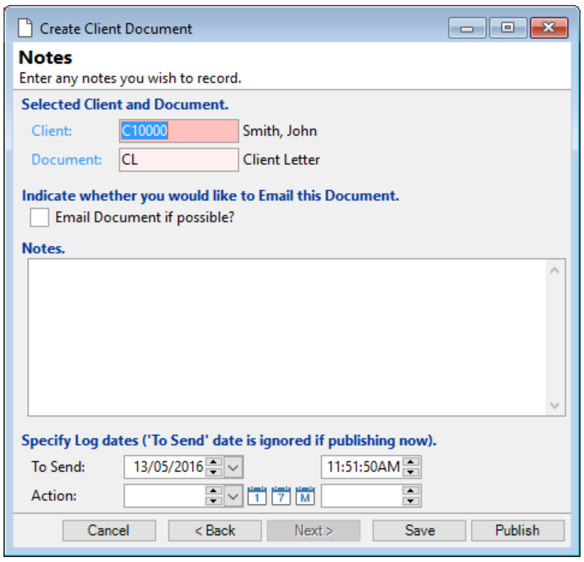
Document

Select the Document to create.

Select the Document to create.

Document	Description
Category : (2 items)	
CA	Client Advice
CL	Client Letter
Category : Electronic (2 items)	
CE	Client General Email
CSMS	Client General SMS

The final page of this wizard summaries the Document that is to be created, e.g.:


Create Client Document

Notes

Enter any notes you wish to record.

Selected Client and Document.

Client: C10000 Smith, John

Document: CL Client Letter

Indicate whether you would like to Email this Document.

☐ Email Document if possible?

Notes.

Specify Log dates ('To Send' date is ignored if publishing now).

To Send: 13/05/2016 11:51:50AM

Action:

NOTE: In the above example, an "Email Document if possible?" checkbox is available.

This is visible because the Document is configured with an Email setting of "Prompt". If the Email setting on the Document was "Yes", this checkbox would still be shown but would be checked by default.

At this point, either the Save or Publish button can be used to complete the wizard. Both of these buttons will create a Log for the Client:

- Save
  - The Client Log will be saved but not published at the moment.
  - The Log's Publish Status will be set based on the "Default Publish Status" setting defined on the Options page of the Documents form.
  - The Log can be published at a later time either by the Documents Publish wizard or via the "Publish" action on the Client Log form.
- Publish
  - The Client Log will be saved.
  - The Client Log will be published.
    - This is the same as using the "Publish" action on the Client Log form or using the `finClientLog.Publish()` business layer method.

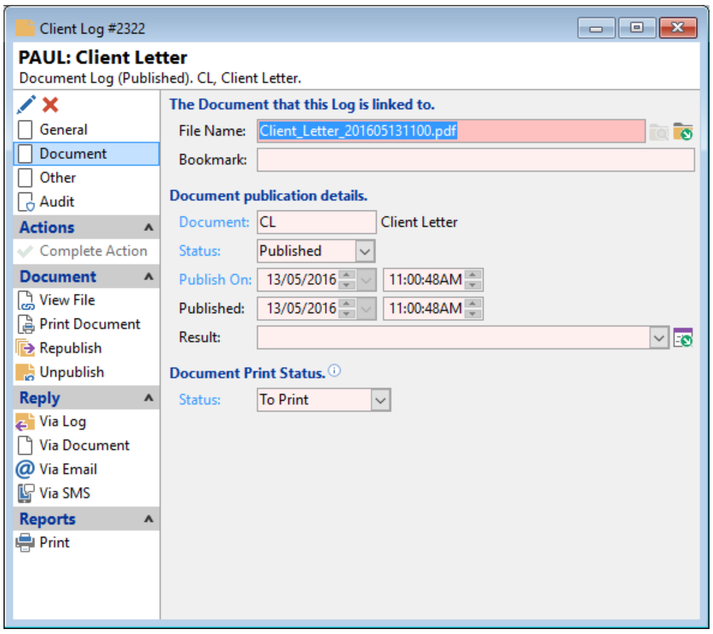
NOTE: Emails are always sent as part of the Publish process, NOT the Printing process.

Publishing Document Logs

Unless the Publish button was used on the Create Document wizard, the Log must now be published. This can be done via the Client Log form or the Publish Documents wizard.

Log form

The "Publish" action can be used to publish the Document (i.e., generate the document file):



Client Log #2322

PAUL: Client Letter

Document Log (Published). CL, Client Letter.

The Document that this Log is linked to.

File Name: Client_Letter_201605131100.pdf

Bookmark:

Document publication details.

Document: CL Client Letter

Status: Published

Publish On: 13/05/2016 11:00:48AM

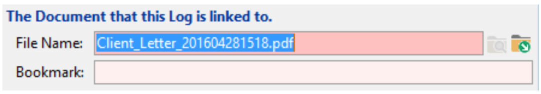
Published: 13/05/2016 11:00:48AM

Result:

Document Print Status.

Status: To Print

Once published, the Log is updated to display the file name of the Document that has been created, e.g.:

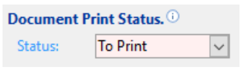


The Document that this Log is linked to.

File Name: Client_Letter_201604281518.pdf

Bookmark:

And, because this is a "Word Document" type Document, and the Document Script has set the Print Status of the Log to "To Print", the Print Documents wizard will automatically be opened to print the document:



Document Print Status.

Status: To Print

NOTE: The Print Documents wizard is only shown if a document file has been produced. If the Document Script simply emailed the document as a file attachment then the Print Status may have been set to "Not Applicable" instead.

Assume that the User has responded to not print the document at this point. If this is the case then the Print Documents wizard would generally be used to print the document.

Publish Documents wizard

The Publish Documents wizard has not changed since the Publish stage and the Print stage are completely separate.

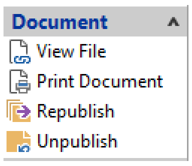
However, if any Logs were published that have a Print Status of 'To Print', the wizard will prompt the User to open the Print Documents wizard which will then show a list of Logs published in the Publish Documents wizard.

Printing Document Logs

Either the "Print Document" action on the Log form or the Print Documents wizard can be used to print the document:

Log Form

A "Print Document" action on Log forms where the Log's Print Status is "To Print".



Selecting this action will open the Print Documents wizard allowing the Document to be quickly printed.

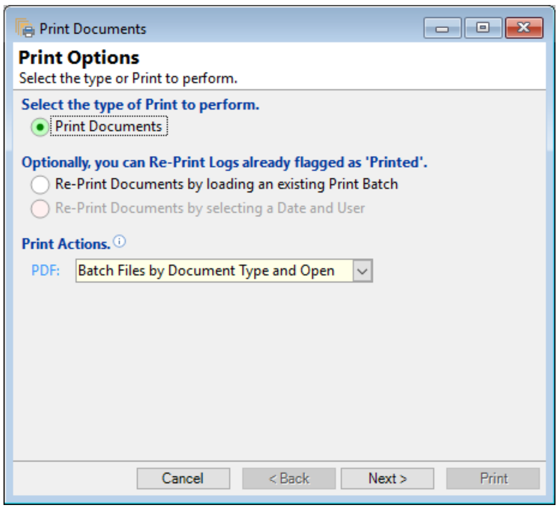
Print Documents wizard

The Print Documents wizard is used to print Document Logs.

NOTE: This wizard does not actually print the Documents but rather batches documents into a single file (or files) which is then opened and from which they can then easily be printed.

Only Logs with a Print Status of "To Print" will be included:

Print Options



Print Documents

Print Options

Select the type or Print to perform.

Select the type of Print to perform.

(●) Print Documents

Optionally, you can Re-Print Logs already flagged as 'Printed'.

( ) Re-Print Documents by loading an existing Print Batch

( ) Re-Print Documents by selecting a Date and User

Print Actions.

PDF: Batch Files by Document Type and Open

Typically, publishing Documents generates PDF files that are either referenced by file name on the Log or, in rarer cases, embedded in the Log.

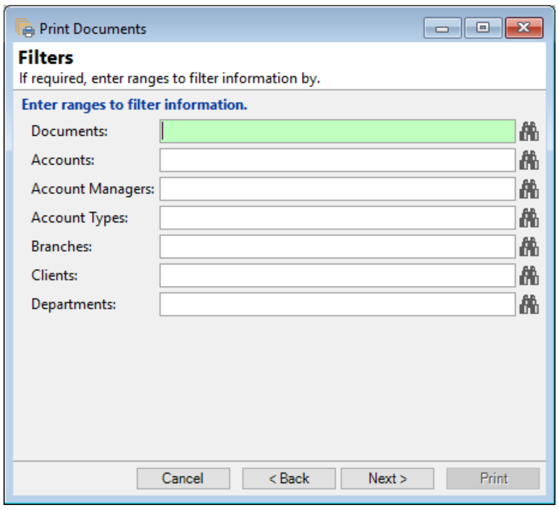
The "Print Actions" section determines how files, in particular, PDF files will be handled:

- **Print Actions**
  - PDF:
    - This determines how the individual PDF files stored with each Log will be combined into a single file (or files):
      - Batch Files by Document Type and Open (the default)
        - All PDF files for a particular Document type will be combined into a single bulk PDF file which will then be opened for printing.
        - NOTE: Once a bulk file exceeds 1024MB, another file will be created.
      - Batch all Files and Open
        - All PDF files will be combined into a single bulk PDF file which will then be opened for printing.
        - NOTE: Once a bulk file exceeds 1024MB, another file will be created.
      - Open Individual Files
        - All PDF files will be opened individually for printing.
        - NOTE: Typically you would only use this option if there are very few Logs to be published since printing the PDF documents would involve activating each one in Windows and using the PDF view to print each document.

Filters

This page will not be displayed when printing a single Log or when this wizard has been launched at the end of the Publish Documents wizard.

It allows filters to be entered:


Print Documents

Filters

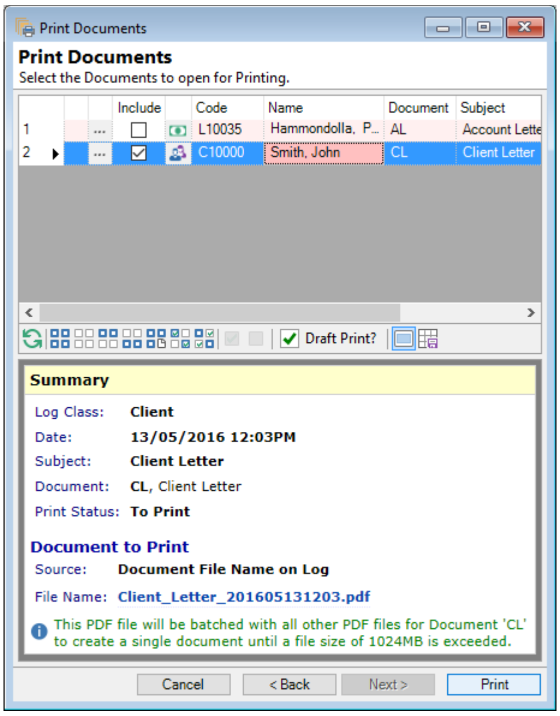
If required, enter ranges to filter information by.

Enter ranges to filter information.

Documents:	
Accounts:	
Account Managers:	
Account Types:	
Branches:	
Clients:	
Departments:	

Print Documents

This page presents a grid of items (Logs) to print.


Print Documents

Print Documents

Select the Documents to open for Printing.

	Include	Code	Name	Document	Subject
1	☐	L10035	Hammondolla, P...	AL	Account Lette
2	☑	C10000	Smith, John	CL	Client Letter

Draft Print? ✓

Summary

Log Class:	**Client**
Date:	**13/05/2016 12:03PM**
Subject:	**Client Letter**
Document:	**CL**, Client Letter
Print Status:	**To Print**

Document to Print

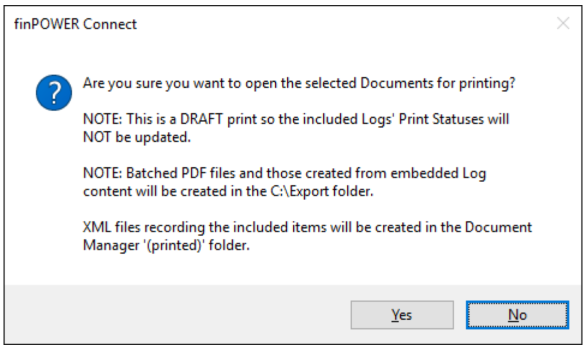
Source: **Document File Name on Log**

File Name: **Client_Letter_201605131203.pdf**

This PDF file will be batched with all other PDF files for Document 'CL' to create a single document until a file size of 1024MB is exceeded.

A summary of the selected Log shows details of the Document to print.

Upon printing, the User receives an information message and the summary is updated to include details of the file that has been opened for printing:

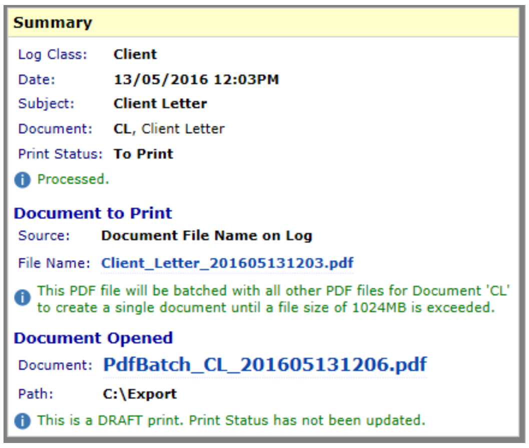

finPOWER Connect

Are you sure you want to open the selected Documents for printing?

NOTE: This is a DRAFT print so the included Logs' Print Statuses will NOT be updated.

NOTE: Batched PDF files and those created from embedded Log content will be created in the C:\Export folder.

XML files recording the included items will be created in the Document Manager '(printed)' folder.



Summary

Log Class:	Client
Date:	13/05/2016 12:03PM
Subject:	Client Letter
Document:	CL, Client Letter
Print Status:	To Print

Processed.

Document to Print

Source:	Document File Name on Log
File Name:	Client_Letter_201605131203.pdf

This PDF file will be batched with all other PDF files for Document 'CL' to create a single document until a file size of 1024MB is exceeded.

Document Opened

Document:	PdfBatch_CL_201605131206.pdf
Path:	C:\Export

This is a DRAFT print. Print Status has not been updated.

NOTE: In the above examples, the "Draft Print?" checkbox was selected. This causes the documents to be opened for printing but does not update the "Print Status" on the Logs.

Emailed Documents

When a Document Emails the generated document file as an attachment, a record of an Emails sent is also recorded.

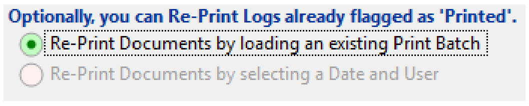
For Client and User Logs, the Email details will always be recorded against the main Log since there is no concept of having multiple recipients. The one exception to this is if the Log has already been published and therefore contains existing Email details. In this case, a child Log will be created to record the Email details.

For Account and Account Application Logs, child Logs will be created for each Client/ Applicant, regardless of whether they could actually be Emailed or not.

This allows for personalised, printed copies of Documents available.

Reprinting Documents via the Print Documents wizard

The first page of the Print Documents wizard has two options to allow Logs to be loaded for re-printing:



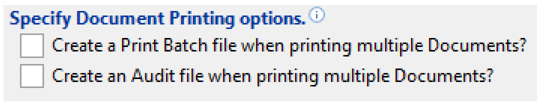
Optionally, you can Re-Print Logs already flagged as 'Printed'.

[● Re-Print Documents by loading an existing Print Batch]

○ Re-Print Documents by selecting a Date and User

Re-Print Documents by loading an existing Print Batch

Under Global Settings, General, Documents and Templates, two options are available to allow the Print Documents wizard to save XML files when printing multiple Logs.



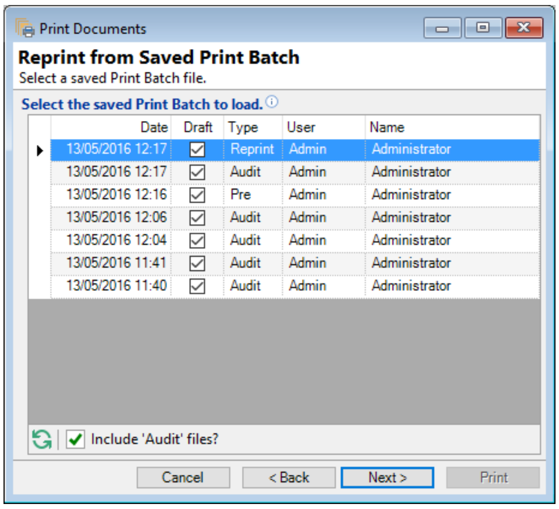
Specify Document Printing options.

☐ Create a Print Batch file when printing multiple Documents?

☐ Create an Audit file when printing multiple Documents?

When these options are checked (they are unchecked by default), XML files are saved to the Document Manager '(printed)' folder.

When this re-print options is selected, the wizard presents a list of all of these XML files (the 'Audit' files are not listed by default), e.g.:



Reprint from Saved Print Batch

Select a saved Print Batch file.

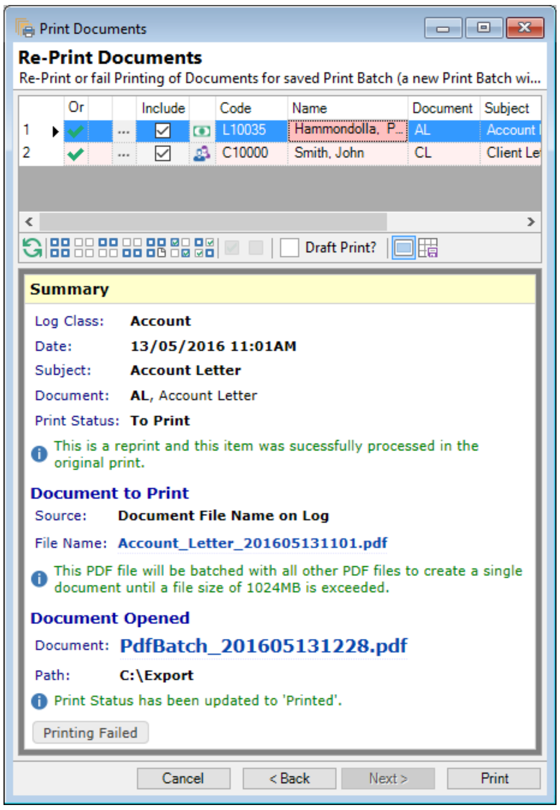
Select the saved Print Batch to load.

Date	Draft	Type	User	Name
13/05/2016 12:17	✓	Reprint	Admin	Administrator
13/05/2016 12:17	✓	Audit	Admin	Administrator
13/05/2016 12:16	✓	Pre	Admin	Administrator
13/05/2016 12:06	✓	Audit	Admin	Administrator
13/05/2016 12:04	✓	Audit	Admin	Administrator
13/05/2016 11:41	✓	Audit	Admin	Administrator
13/05/2016 11:40	✓	Audit	Admin	Administrator

☑ Include 'Audit' files?

WARNING: Any 'Pre' files indicate that the status of this print batch is unknown, e.g., the User's PC was shut down before a 'Reprint' file could be created.

Documents can then be reprinted, e.g.:



Re-Print Documents

Re-Print or fail Printing of Documents for saved Print Batch (a new Print Batch wi...

		Or		Include		Code	Name	Document	Subject
1	▶	✓	...	☑		L10035	Hammondolla, P...	AL	Account l
2		✓	...	☑		C10000	Smith, John	CL	Client Le

Summary

Log Class:	**Account**
Date:	**13/05/2016 11:01AM**
Subject:	**Account Letter**
Document:	**AL**, Account Letter
Print Status:	**To Print**

This is a reprint and this item was sucessfully processed in the original print.

Document to Print

Source:	**Document File Name on Log**
File Name:	**Account_Letter_201605131101.pdf**

This PDF file will be batched with all other PDF files to create a single document until a file size of 1024MB is exceeded.

Document Opened

Document:	**PdfBatch_201605131228.pdf**
Path:	**C:\Export**

Print Status has been updated to 'Printed'.

NOTE: The 'Orig' column displays a tick icon for items that were originally processed successfully.

Re-Print Documents by selecting a Date and User

This is not yet implemented.

Standalone Documents

TODO:

Documents Form

The various pages on the Documents form that relate to Word Documents are described in this section. Many of these properties are expanded on in later sections.

Document

Document details.

- **Document File Type**
  - File Type:
    - This will always be "Word Document".
- **File Name and Location**
  - File Name:
    - The file name (relative to the Templates folder) of the MS Word Document (.docx file) that will be used as a template for new Word Documents.
  - Embedded:
    - The file named above can be embedded within the Document record so that access to the Templates folder is not required to use this Document.

      NOTE: All Word Documents supplied with finPOWER Connect have the docx file embedded within them.
    - The various buttons can be used to embed the .docx file, save the embedded file to the file system or validate that the embedded .docx file matches the file named above (generally a copy of the original .docx file that resides in the Templates folder).
- **Template Email Message for us in Script**
  - Email details that can be used by the Document Script, generally to send the generated PDF file as an Email attachment.

Options

Miscellaneous options. Many of these are for use by the Document Script only.

- **General options**
  - Allow Standalone running of this Document?
    - Generally, Documents are generated from Logs, e.g., an Account Log. However, when producing bulk Documents, e.g., for a marketing campaign, it is useful to be able to generate Documents en masse (generally via the Report Explorer).

      NOTE: It is the responsibility of the Document Script to support this functionality.
    - Show in the Report Explorer?
      - Indicates whether this Standalone Document should appear in the Report Explorer.
    - Allow Document to be run from a Log.
      - Indicates whether this Document can be run from a Log, e.g., whether it should appear in the "Create Account Document" wizard that is run from the Accounts form, "Send Document" action.
  - **Default Publish Status**
    - Status:
      - The default "Publish Status" to set on any Logs generated for this Document.
    - Print Document upon Publish?

        - Indicates whether, upon publishing a Document from the User Interface (e.g., the Create Client Document wizard), the Document will be automatically printed (i.e., go through the Print Documents process, not actually printed).

- **Preferences to be used by Document Script**
    - Save Document automatically?
        - Indicates that the Script should save the generated Document to the Document Manager folder.
    - Create Log record when running Standalone?
        - Indicates that when the Document is run Standalone, a Log record (e.g., a Client or Account Log) should be generated for each Document recipient.
    - Embed Document in Log record?
        - Indicates whether the Document should be embedded in the Log.

          NOTE: Typically you would not embed Documents in a Log record since it is likely to bloat the database. However, for certain small Documents that may be required to be accessed from a Web application that does not have access to the Document Manager folder, this can be advantageous.
    - **Email Preferences**
        - Email:
            - Indicates whether the Script supports Emailing of the Document (i.e., Yes and Prompt) and, whether the option to Email the Document should be checked by default (i.e., Yes).
            - Emails contain personalised details so do not combine for 'Joint' records?
                - If the Email contains Client or Applicant-specific information, e.g., a Client Id or a Client Name then this option should be checked. This prevents combined Emails from being sent to 'Joint' Clients or Applicants; instead, each will be sent their own, personalised, Email.
        - Always Email a copy of the primary document?
            - Indicates whether to always Email to primary (usually the first, i.e., for the 'Main' Client) document that was created to all Clients.
                - NOTE: This does not prevent personalised documents being created and saved for each Account Client; simply the document that is Emailed. The Log notes indicate whether a document for another recipient was attached to the Email.
        - Print Document even if Emailed?
            - Indicates that the Print Status of the Log should be set to "To Print", even if the Document was emailed by the Document Script.
                - By default, the Print Status will be set to "Not Applicable" if the Document was emailed.
    - **Account Clients to receive this Document**
        - Owners:
            -
        - Guarantors?
            -
        - Other?
            -
        - Send to Joint Clients individually?
            -

NOTE: The finPOWER Connect business layer can be used to automatically build a list of Account Clients (or Account Application Applicants) to receive a Document based on the "Account Clients to receive this Document" options.

Creating an MS Word Document

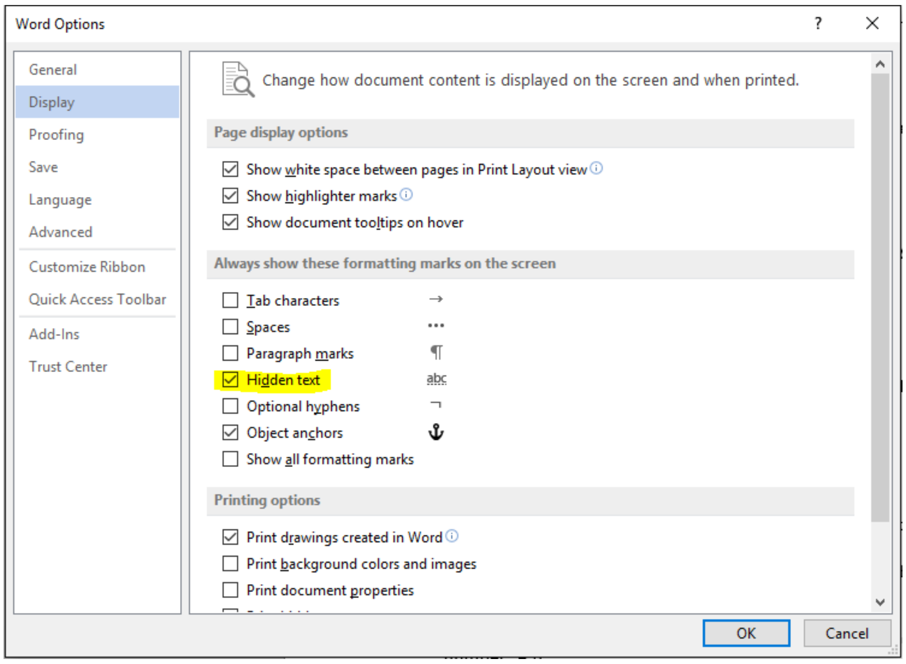
Typically, all "Word Document" type Documents will have an MS Word Document (.docx file) which they use as a template.

See Appendix C – Converting an Existing MS Word VBA Template for information on converting existing MS Word Templates (.dot files).

Bookmarks

Bookmarks are used to insert customised values into the Document or remove sections of the Document.

To display Bookmarks in MS Word, select File, Options, Display and check "Hidden Text":



Word Options

General
Display
Proofing
Save
Language
Advanced
Customize Ribbon
Quick Access Toolbar
Add-Ins
Trust Center

Change how document content is displayed on the screen and when printed.

Page display options

☑ Show white space between pages in Print Layout view
☑ Show highlighter marks
☑ Show document tooltips on hover

Always show these formatting marks on the screen

☐ Tab characters →
☐ Spaces ···
☐ Paragraph marks ¶
☑ Hidden text abc
☐ Optional hyphens ¬
☑ Object anchors ⚓
☐ Show all formatting marks

Printing options

☑ Print drawings created in Word
☐ Print background colors and images
☐ Print document properties

OK Cancel

There are several types of Bookmark:

- **Standard "substitute" bookmarks:**
    - Updated via `finDocumentRecipientDetailsItem.UpdateStandardBookmarks` method.
    - See Appendix B – Standard Bookmarks for a full list of standard Bookmarks that the finPOWER Connect business layer can replace automatically.
    - NOTE: Standard Bookmarks can still be customised after being in the `UpdateStandardBookmarks` method.
- **Custom "substitute" bookmarks:**
    - Handled via custom Script code.
        - In the supplied samples, all custom bookmarks are handled in a `ReplaceBookmarks` function.
- **Special "markers" to add or delete text etc.:**
    - For example, to remove a clause if the document should not include it.

Bookmark Naming

All bookmarks within an MS Word Document must adhere to the following:

- Bookmark names must:
    - Be unique.

    - Only consist of letters and numbers.
- Bookmarks should be named as per the content their default content.
    - For example, a bookmark to display the Id of an Account should be named **AccountAccountId**
        - The content of "substitute" bookmarks should be the bookmark name enclosed in square brackets and be in the format "Table.Field", e.g.: **[Account.AccountId]**
    - If you need multiple occurrences of the bookmark to contain the same value, append a number, e.g.: **AccountAccountId1, AccountAccountId2, AccountAccountId3**

Bookmark Replacement

Replacing of MS Word bookmarks is performed in the Document Script.

Various methods exist to allow text or HTML to be inserted into bookmarks and for the content of a bookmark to be removed (useful for removing optional paragraphs and clauses).

Text

Plain text.

```
.SetContentString("LetterContent", "My content")
```

Removing Bookmarks

Bookmarks can be used to mark an optional paragraph or clause in a document.

This can then be removed via Script code, e.g.:

```
.Remove("LetterContent")
```

HTML

HTML support relies entirely on what is supported by the GemBox component used to generate Word Documents and also MS Word itself.

As of version 3.00.00, this functionality is quite limited, particularly when dealing with tables.

Some table limitations include:

- Only percentage cell widths appear to work reliably; px, em and pt either do not work or are not reliable.

```
.SetContentHtml("LetterContent", "My <b>HTML</b> content")
```

Note, HTML do not currently support some functionality, including:

- HTML Templates, i.e. templates enclosed in {{ and }}.

Images

Images can be inserted into a bookmark.

A height in points can optionally be specified, e.g.:

```
.SetContentImageFileName("Logo", Branch.LogoFileNameResolved, 40)
```

Summary Tables

Summary Tables allow an `ISSummaryTable` object to be inserted into a bookmark.

There are two methods that Summary Tables can be inserted.

SetContentHtmlSummaryTable(s)

The `SetContentHtmlSummaryTable` method simply piggy-backs the HTML bookmark functionality and therefore has all the same limitations relating to cell widths and other formatting.

Warnings:

- SetContentSummaryTable(s) is the preferred method.
- As with SetContentHtml, there are limitations to supported functionality.

```
.SetContentHtmlSummaryTable("Transactions", SummaryTable)
.SetContentHtmlSummaryTables("Transactions", SummaryTables)
```

SetContentSummaryTable(s)

This is the preferred method of adding a table or tables into a Word Document.

- Include additional options used to determine how to format the Table, including Rows and Cells.
  - The Summary Table includes a WordDocumentOptions object.
    - ISSummaryTableWordDocumentOptions
  - Rows include a WordDocumentOptions object.
    - ISSummaryTableRowWordDocumentOptions
  - Cells include a WordDocumentOptions object.
    - ISSummaryTableCellWordDocumentOptions.
  - These define properties such as fonts, borders, padding and paragraph spacing.

```
.SetContentSummaryTable("Transactions", SummaryTable)
.SetContentSummaryTables("Transactions", SummaryTables)
```

Note, Summary Tables do not currently support some functionality, including:

- HTML Templates, i.e. templates enclosed in {{ and }}.
- AddCheckBox
  - Instead of using checkboxs consider using special characters, e.g. ☐, ☑ and ☒.

Tables

Tables are fundamental to the layout of many Word documents.

The GemBox software used by finPOWER Connect to handle Word Documents and, in particular to convert an MS Word document to a PDF document, may render tables slightly differently that MS Word. This can lead to alignment issues.

This may be particularly apparent when using nested tables, i.e., tables within tables.

Note, repeating Heading Rows on each page does NOT work where a nested table is used.

Following these guidelines will help minimise these issues:

- Position the cursor in the table.
- Right-click and select Table Properties
- Use a table percentage width. Preferably 100%.
    - Select the Table tab:

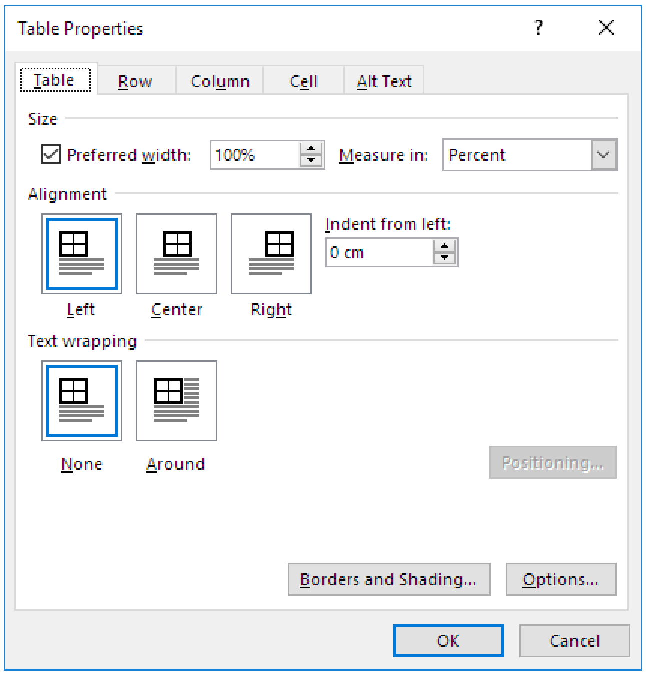


Table Properties

Table | Row | Column | Cell | Alt Text

Size

[✓] Preferred width: 100% Measure in: Percent

Alignment

[✓ Left] Center Right

Indent from left: 0 cm

Text wrapping

[✓ None] Around

Positioning...

Borders and Shading... Options...

OK Cancel

- Set table default cell padding to zero:
    - From the Table tab, click the Options... button:

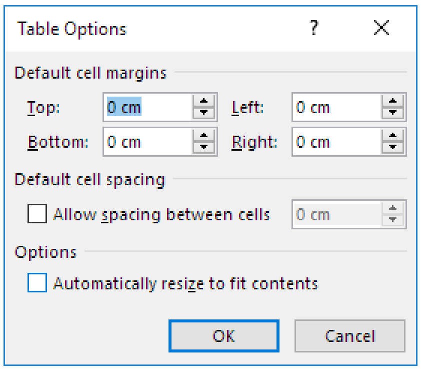


Table Options

Default cell margins

Top: 0 cm Left: 0 cm

Bottom: 0 cm Right: 0 cm

Default cell spacing

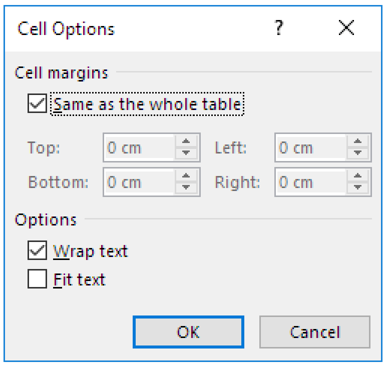
[ ] Allow spacing between cells 0 cm

Options

[ ] Automatically resize to fit contents

OK Cancel

- Set all cell paddings to the table default:
    - From the Cell tab, click the Options... button:


Cell Options

Cell margins

☑ Same as the whole table

Top: 0 cm Left: 0 cm

Bottom: 0 cm Right: 0 cm

Options

☑ Wrap text

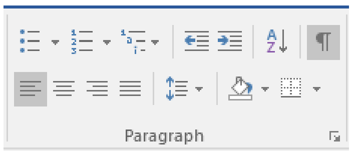
☐ Fit text

This will produce a compact table where the table content is bumped hard-up against the edge of the cell, e.g.:

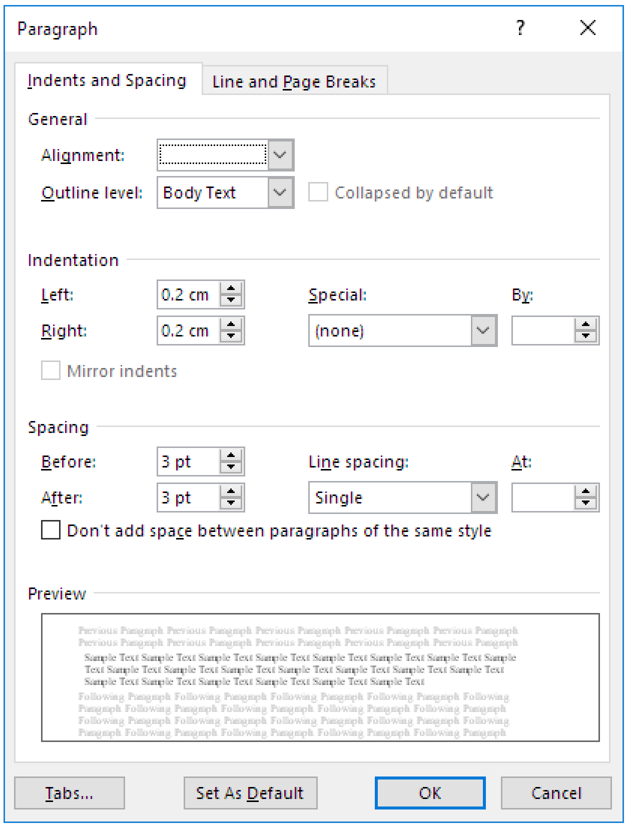
Cell A	Cell B	Cell C
Cell D	Cell E	Cell F

Use Paragraph padding to pad the cell contents:

- Select all Table Cells.
- From the Home tab on the MS Word Ribbon, click the small arrow to the right of Paragraph:



- This will display the Paragraph Properties dialog:


Paragraph

Indents and Spacing | Line and Page Breaks

General

Alignment:

Outline level: Body Text ☐ Collapsed by default

Indentation

Left: 0.2 cm Special: By:

Right: 0.2 cm (none)

☐ Mirror indents

Spacing

Before: 3 pt Line spacing: At:

After: 3 pt Single

☐ Don't add space between paragraphs of the same style

Preview

- From here, you can set the Left, Right indentation and, if required the Before and After spacing.

Cell A	Cell B	Cell C
Cell D	Cell E	Cell F

Document Script

The Document Script for Word Documents only supports a single event: "Publish".

It is the responsibility of this event to generate the document file and either save it to the Document Manager folder or, less commonly, embed the file content in the Log record.

Many of the supplied Word Documents (e.g., CL, "Client Letter") support both publishing via a Log or as a Standalone document.

Others such as R1, "Overdue Reminder 1" only support being published from a Log.

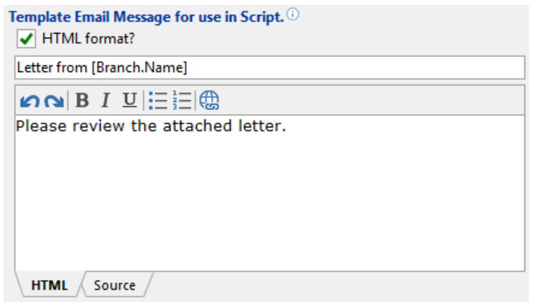
NOTE: Supporting Standalone publishing (e.g., via the Report Explorer) requires that the Script implement alternate functionally, therefore unless specified, the examples in this section only refer to the Script code used for Log publishing.

Only Client and Account type Documents are discussed in this section. Account Application Documents are very similar to Account Documents and User Documents are similar to Client Documents.

Client Document

This section uses the sample CL, "Client Letter".

This Document supports emailing the generated PDF file as a file attachment. A template email is defined on the Document page:



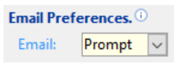
Template Email Message for use in Script.

✓ HTML format?

Letter from [Branch.Name]

Please review the attached letter.

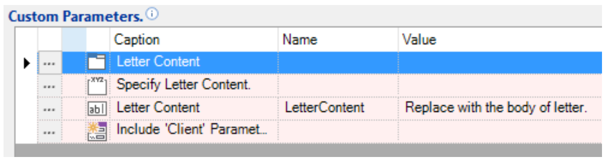
And the Options page specifies that the Document supports emailing:



Email Preferences.

Email: Prompt

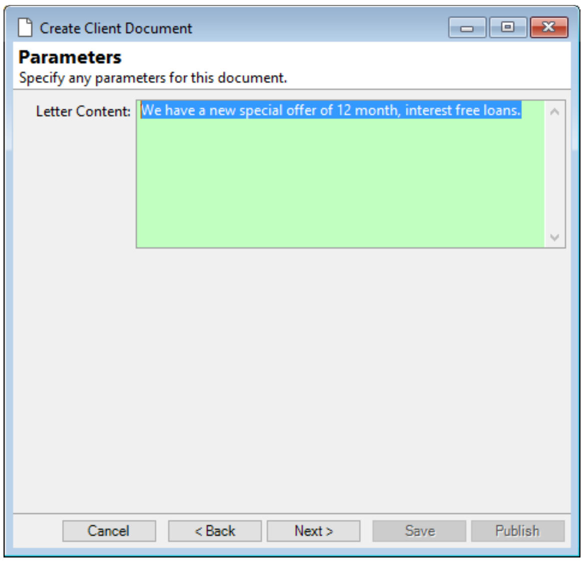
This Document defines a single parameter that is applicable to Log publishing: "LetterContent".



Custom Parameters.

Caption	Name	Value
Letter Content		
Specify Letter Content.		
Letter Content	LetterContent	Replace with the body of letter.
Include 'Client' Paramet...		

This allows a User (or Script) to enter some text that is used for the letter content, e.g.:



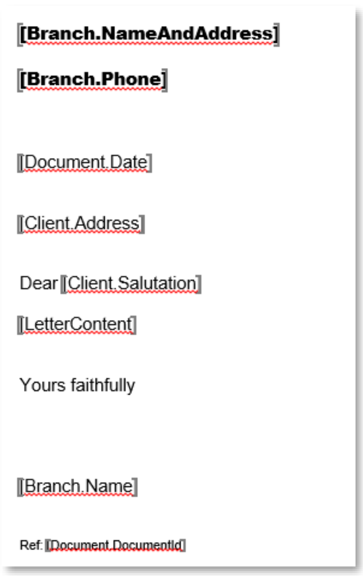
Create Client Document

Parameters

Specify any parameters for this document.

Letter Content: We have a new special offer of 12 month, interest free loans.

The Script simply replaces the "LetterContent" bookmark that is defined in the MS Word Document (.docx file) embedded in the Document record:



[Branch.NameAndAddress]

[Branch.Phone]

[Document.Date]

[Client.Address]

Dear [Client.Salutation]

[LetterContent]

Yours faithfully

[Branch.Name]

Ref: [Document.DocumentId]

Main method

The following is a stripped down version of the code from the Main method (all code relating to Standalone publishing has been removed):

```
Public Function Main(source As Object,
                     eventId As String,
                     eventArgs As ISKeyValueList,
                     ByRef handled As Boolean,
                     parameters As Object,
                     ByRef returnValues As ISKeyValueList,
                     ByRef text As String) As Boolean

  Dim Success As Boolean

  ' Assume Success
  Success = True

  ' Initialise
  mDocument = finBL.Documents(Mid(ScriptInfo.ScriptId, 10))
  mWordDocuments = New List(Of ISWordDocument)

  ' Handle Events
  Select Case eventId
    Case "Publish"
      ' Publish
      Success = Publish_Log(DirectCast(source, finLogPublishBatch)(0))
  End Select

  Return Success

End Function
```

This method does the following:

- Gets a reference to the finPOWER Connect Document (`mDocument`) since this contains various pieces of information and helper functions required by the Script.
- Create a list of `ISWordDocument` objects. This is used later in the Script.
- Handles the "Publish" event which simply involves calling the `Publish_Log` method.
    - NOTE: Just like the older style MS Word VBA templates, the Script is passed a `finLogPublishBatch` object containing information about the Logs that are being published. However, unlike MS Word VBA templates, this object will only ever contain a single item (item 0).

NOTE: The `ScriptInfo.ScriptId` property for Documents has a "Document_" prefix which is removed to get the actual Document Id.

Publish_Log method

This method gets details and handles the entire Log publication process and will not generally need to be updated from the default code supplied with the Document.

```
Private Function Publish_Log(logPublishBatchItem As finLogPublishBatchItem) As Boolean

  Dim di As finDocumentRecipientDetailsItem
  Dim DocumentBinaryData As Byte()
  Dim DocumentManagerFileName As String
  Dim DocumentManagerFileNameRelative As String
  Dim Success As Boolean

  ' Assume Success
  Success = True

  ' Validate

  ' Get Document Recipient Details for Client
  If Success Then
    Success =
mDocument.GetClientDocumentRecipientDetailsItemFromLogPublishBatchItem(logPublishBatchItem, di)
  End If

  ' Create Document
  If Success Then
    Success = CreateDocument(di)
  End If

  ' Save Document to Document Manager?
  If Success AndAlso mDocument.AutoSaveDocument Then
    Success =
finBL.DocumentManagerFunctions.SaveWordDocumentToDocumentManagerPdfClient(di.ClientId,
di.WordDocument, mDocument.FileName, DocumentManagerFileName, DocumentManagerFileNameRelative)
  End If

  ' Embed in Log?
  If Success AndAlso mDocument.EmbedDocumentInLog Then
    Success = di.WordDocument.SavePdfToByteArray(DocumentBinaryData)
  End If

  ' Update Log Publish details
  If Success Then
    With logPublishBatchItem
      .PublishSuccess = True
      .PublishDocumentFileName = DocumentManagerFileNameRelative
      .PublishDocumentBinaryDataFileType = isefinLogEmbeddedFileType.Pdf
      .PublishDocumentBinaryData = DocumentBinaryData
      .PrintStatus = di.PrintStatus
    End With
  End If

  ' Record Email details
  If Success AndAlso di.EmailDetails.EmailSent Then
    If Not di.EmailDetails.ClientLogUpdateForEmailSend(logPublishBatchItem.LogPk) Then
      ' Ignore Errors
    End If
  End If

  Return Success

 End Function
```

This method does the following:

- Calls the `finDocumentRO.GetClientDocumentRecipientDetailsItemFromLogPublishBatchItem` helper method to retrieve information such as Branch, Client Address and Client Email that will be useful in creating the document.
    - IMPORTANT: The returned `finDocumentRecipientDetailsItem` also contains a `WordDocument` property which is an `ISWordDocument` object pre-loaded with the .docx file that is either embedded in of referenced by the finPOWER Connect Document.
- Calls the `CreateDocument` method.
    - This generates an `ISWordDocument` and adds it into the `mWordDocuments` list

- Save the generated Word Document to the Document Manager as a PDF file.
  - This is done only if the finPOWER Connect Document is configured to `AutoSaveDocument` (defined on the Options page of the Documents form).
- Embeds the generated Word Document as a PDF file in the Log record.
  - This is only if the finPOWER Connect Document is configured to `EmbedDocumentInLog` (defined on the Options page of the Documents form).
  - NOTE: It would not be usual to embed document content within a Log since it may cause the database to become bloated.
- The `finLogPublishBatchItem` is updated to record the status of the publish.
  - This is used by the finPOWER Connect publishing functionality and is used to update the Log, charge fees etc at the end of the publish process.
- If an email was sent (in the `CreateDocument` method), the details of this email are recorded on the Log object for auditing purposes.

CreateDocument method

This method generates the Word Document and would not generally need to be modified in any of the supplied Documents unless customising Email content.

```
Private Function CreateDocument(di As finDocumentRecipientDetailsItem) As Boolean

  Dim Success As Boolean
  Dim WordDocumentBookmarks As ISWordDocumentBookmarks

  ' Assume Success
  Success = True

  ' Get Bookmarks
  WordDocumentBookmarks = di.WordDocument.GetBookmarks()

  ' Replace Standard Bookmarks
  Success = di.UpdateStandardBookmarks(WordDocumentBookmarks)

  ' Replace Custom Bookmarks
  If Success Then
    ReplaceBookmarks(di, WordDocumentBookmarks)
  End If

  ' Update Bookmarks
  If Success Then
    Success = di.WordDocument.UpdateBookmarks(WordDocumentBookmarks)
  End If

  ' Record Word Document
  If Success Then
    mWordDocuments.Add(di.WordDocument)
  End If

  ' Update di.EmailDetails if required
  ' NOTE: By default, these are set to the Email template details defined on this Document

  ' Email Document?
  If di.EmailDetails.PublishEmail Then di.EmailDetails.SendEmailPdf()

  Return Success

End Function
```

This method does the following:

- Gets a collection of all bookmarks defined in the MS Word document (.docx file).
- Replaces all Standard bookmarks.
    - See Appendix B for a full list of standard bookmarks.
- Calls the ReplaceBookmarks method to replace Custom, (i.e., document-specific) bookmarks.
- Update Bookmarks, i.e., actually update the Word Document with the contents of the bookmarks set in the above blocks.
- Adds the Word Document to the `mWordDocuments` list.
- Emails the Document if necessary.

NOTE: The Email Address, Subject and Message defined on `di.EmailDetails` (`finDocumentRecipientDetailsItem.EmailDetails`) can be overridden by the Script prior to the email being sent.

This means that no email template details are required to be defined on the Document and, if required, the Email Address can be updated or overridden or appended.

ReplaceBookmarks method

This method updates any document-specific bookmarks and is generally the only method that will need modifying in any of the supplied Documents.

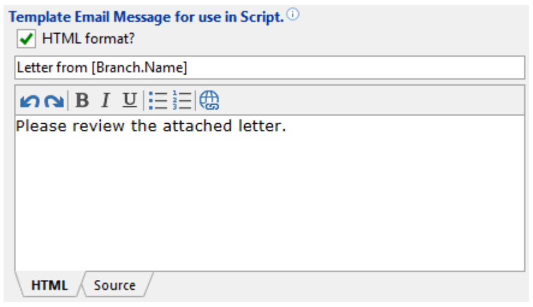
```
Private Function ReplaceBookmarks(di As finDocumentRecipientDetailsItem,
                                  wordDocumentBookmarks As ISWordDocumentBookmarks) As Boolean

  Dim Success As Boolean

  ' Assume Success
  Success = True

  ' Parameters
  If Success Then
    With wordDocumentBookmarks
      .SetContentString("LetterContent", di.Parameters.GetString("LetterContent"))
    End With
  End If

  Return Success

End Function
```

Account Document

This section uses the sample R1, "Overdue Reminder 1".

WARING: Account Documents are inherently more complicated than Client Documents since they must potentially handle sending a document and/ or email to multiple Account Clients and must take into consideration "Joint" Clients.

This Document supports emailing the generated PDF file as a file attachment. A template email is defined on the Document page:



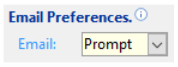
Template Email Message for use in Script.

✓ HTML format?

Letter from [Branch.Name]

Please review the attached letter.

And the Options page specifies that the Document supports emailing:



Email Preferences.

Email: Prompt

Main method

The following is a stripped down version of the code from the Main method (all code relating to Standalone publishing has been removed):

```
Public Function Main(source As Object,
                     eventId As String,
                     eventArgs As ISKeyValueList,
                     ByRef handled As Boolean,
                     parameters As Object,
                     ByRef returnValues As ISKeyValueList,
                     ByRef text As String) As Boolean

  Dim Success As Boolean

  ' Assume Success
  Success = True

  ' Initialise
  mDocument = finBL.Documents(Mid(ScriptInfo.ScriptId, 10))
  mWordDocuments = New List(Of ISWordDocument)
  mWordDocumentsAccount = New List(Of ISWordDocument)

  ' Handle Events
  Select Case eventId
    Case "Publish"
      ' Publish
      Success = Publish_Log(DirectCast(source, finLogPublishBatch)(0))
  End Select

  Return Success

End Function
```

This method does the following:

- Gets a reference to the finPOWER Connect Document (`mDocument`) since this contains various pieces of information and helper functions required by the Script.
- Create a list of `ISWordDocument` objects. This is used later in the Script.
- Create a list if `ISWordDocument` objects for a particular Account. This is used later in the Script.
  - NOTE: This line is highlighted since it is the only line that differs from the `Main` method defined for Client Documents.
- Handles the "Publish" event which simply involves calling the `Publish_Log` method.
  - NOTE: Just like the older style MS Word VBA templates, the Script is passed a `finLogPublishBatch` object containing information about the Logs that are being published. However, unlike MS Word VBA templates, this object will only ever contain a single item (item 0).

NOTE: The `ScriptInfo.ScriptId` property for Documents has a "Document_" prefix which is removed to get the actual Document Id.

Publish_Log method

This method gets details and handles the entire Log publication process and will not generally need to be updated from the default code supplied with the Document.

```
Private Function Publish_Log(logPublishBatchItem As finLogPublishBatchItem) As Boolean

  Dim dis As finDocumentRecipientDetailsItems
  Dim Success As Boolean

  ' Assume Success
  Success = True

  ' Validate

  ' Get Document Recipient Details for relevant Account Clients
  If Success Then
    Success =
mDocument.GetAccountDocumentRecipientDetailsItemFromLogPublishBatchItem(logPublishBatchItem,
PersonalisedEmails, dis)
  End If

  ' Create Document
  If Success Then
    If Not CreateAccountDocument(dis, logPublishBatchItem, logPublishBatchItem.LogPk, Nothing)
Then
      ' Ignore errors since the status of logPublishBatchitem will be updated to record this
    End If
  End If

  Return Success

End Function
```

NOTE: This differs significantly from the method used for Client Documents since an Account must handle multiple Clients.

This method does the following:

- Calls the `finDocumentRO.GetAccountDocumentRecipientDetailsItemFromLogPublishBatchItem` helper method to retrieve information for each applicable Account Client such as Branch, Client Address and Client Email that will be useful in creating the document.
    - IMPORTANT: Each of the returned `finDocumentRecipientDetailsItem` also contains a `WordDocument` property which is an `ISWordDocument` object pre-loaded with the .docx file that is either embedded in of referenced by the finPOWER Connect Document.
- Calls the `CreateAccountDocument` method.

CreateAccountDocument method

This is a stripped-down version of the code from the `CreateAccountDocument` method (all code relating to Standalone publishing has been removed).

This method handles generating a Document and Emailing the Document for each applicable Account Client.

```
Private Function CreateAccountDocument(dis As finDocumentRecipientDetailsItems,
                                       logPublishBatchItem As finLogPublishBatchItem,
                                       parentLogPk As Integer,
                                       ByRef accountLog As finAccountLog) As Boolean

  Dim di As finDocumentRecipientDetailsItem
  Dim DocumentManagerFileName As String
  Dim DocumentManagerFileNameRelative As String
  Dim DocumentBinaryData As Byte()
  Dim Success As Boolean

  ' Assume Success
  Success = True

  ' Initialise
  mWordDocumentsAccount.Clear()

  ' Create Document for all relevant Account Clients
  For Each di In dis
    ' Create Document
    If Not CreateDocument(di) Then
      Success = False
      Exit For
    End If
  Next

  ' Save Combined Document to Document Manager?
  ' NOTE: This is for ALL Account Clients, regardless of whether individual documents will be
saved for each Account Client
  If Success AndAlso mDocument.AutoSaveDocument Then
    Success =
finBL.DocumentManagerFunctions.SaveWordDocumentsToDocumentManagerPdfAccount(di.AccountId,
mWordDocumentsAccount, mDocument.FileName, DocumentManagerFileName,
DocumentManagerFileNameRelative)
  End If

  ' Embed Combined Document in Log?
  ' NOTE: This is for ALL Account Clients, regardless of whether individual documents will be
saved for each Account Client
  If Success AndAlso mDocument.EmbedDocumentInLog Then
    Success = finBL.PdfUtilities.CombineWordDocumentsToPdfByteArray(mWordDocumentsAccount,
DocumentBinaryData)
  End If

  ' Email Account Clients?
  ' NOTE: This will create additional Account Logs (for each Account Client) where neccessary
  If Success AndAlso dis(0).EmailDetails.PublishEmail Then
    Success = dis.SendAccountEmailsPdf(parentLogPk)
  End If

  ' Update Log Publish details
  If Success AndAlso logPublishBatchItem IsNot Nothing Then
    With logPublishBatchItem
      ' Success
      .PublishSuccess = True
      .PublishDocumentFileName = DocumentManagerFileNameRelative
      .PublishDocumentBinaryDataFileType = isefinLogEmbeddedFileType.Pdf
      .PublishDocumentBinaryData = DocumentBinaryData
    End With
  End If

  Return Success

End Function
```

This method does the following:

- Clears the `mWordDocumentsAccount` list.

    - NOTE: Clearing this list is only necessary when Documents for multiple Accounts are being created, i.e., when running Standalone from the Report Explorer.
- Calls the `CreateDocument` method for each recipient to create the Word Document for the recipient.
    - The also adds the Word Document to the `mWordDocumentsAccount` list.
- Saves the combined Document (i.e., a file containing the Word Documents for all entries in the `mWordDocumentsAccount` list).
- Embeds the combined Document in the Account Log if necessary.
- Calls the `finDocumentRecipientDetailsItems.SendAccountEmailsPdf` method which will, if possible, send an Email to each Account Client.
    - An additional Account Log is created to record each Email sent and the Document for that Account Client is saved to the Log (as opposed to the combined Document that is saved to the main Log).
    - NOTE: It may be that an Account Client cannot be Emailed (e.g., no Email Address recorded). In this case, the Log will be created with a Print Status of 'To Print'.

CreateDocument method

This method generates the Word Document for an Account Client (or Joint Clients) and would not generally need to be modified in any of the supplied Documents unless customising Email content.

```
Private Function CreateDocument(di As finDocumentRecipientDetailsItem) As Boolean

  Dim Success As Boolean
  Dim WordDocumentBookmarks As ISWordDocumentBookmarks

  ' Assume Success
  Success = True

  ' Get Bookmarks
  WordDocumentBookmarks = di.WordDocument.GetBookmarks()

  ' Replace Standard Bookmarks
  Success = di.UpdateStandardBookmarks(WordDocumentBookmarks)

  ' Replace Custom Bookmarks
  If Success Then
    ReplaceBookmarks(di, WordDocumentBookmarks)
  End If

  ' Update Bookmarks
  If Success Then
    Success = di.WordDocument.UpdateBookmarks(WordDocumentBookmarks)
  End If

  ' Record Word Document
  If Success Then
    mWordDocuments.Add(di.WordDocument)
    mWordDocumentsAccount.Add(di.WordDocument)
  End If

  ' Update di Email details if required
  ' NOTE: By default, these are set to the Email template details defined on this Document

  Return Success

End Function
```

NOTE: Account documents differ from Client Documents in that you would typically use Account-specific functionality (including standard bookmarks such as AccountSalutation) since a Document may be targeting Joint Clients.

Also, Emailing of the Document is not handled in this method, it is handled in the `CreateAccountDocument` method instead.

This method does the following:

- Gets a collection of all bookmarks defined in the MS Word document (.docx file).
- Replaces all Standard bookmarks.
  - See Appendix B for a full list of standard bookmarks.
- Calls the ReplaceBookmarks method to replace Custom, (i.e., document-specific) bookmarks.
- Update Bookmarks, i.e., actually update the Word Document with the contents of the bookmarks set in the above blocks.
- Adds the Word Document to the `mWordDocuments` list.
- Adds the Word Document to the `mWordDocumentsAccount` list.
  - This list contains Documents for all applicable Account Clients.

NOTE: The Email Address, Subject and Message defined on `di.EmailDetails` (`finDocumentRecipientDetailsItem.EmailDetails`) can be overridden by the Script prior to the email being sent.

This means that no email template details are required to be defined on the Document and, if required, the Email Address can be updated or overridden (multiple email addresses can be specified by separating then by semi-colons).

ReplaceBookmarks method

This method updates any document-specific bookmarks and is generally the only method that will need modifying in any of the supplied Documents.

```
Private Function ReplaceBookmarks(di As finDocumentRecipientDetailsItem,
                                  wordDocumentBookmarks As ISWordDocumentBookmarks) As Boolean

  Dim Success As Boolean

  ' Assume Success
  Success = True

  ' Parameters
  If Success Then
    With wordDocumentBookmarks
      .SetContentString("LetterContent", di.Parameters.GetString("LetterContent"))
    End With
  End If

  Return Success

End Function
```

Customising Document Scripts

The sample Document Scripts for Client and User Documents (e.g., Client_Letter.xml and User_Letter.xml) are reasonably self-explanatory since they are only ever creating a Document for a single recipient).

However, Accounts and Account Applications increase complexity by needing to support multiple recipients, Joint Clients etc.

This section details how to create either completely custom Documents Scripts or make modifications to existing Scripts to cater for situations that can't be handled using the supplied Scripts and/ or optional configurable on a Document.

Email CC the Dealer

An Account's Dealer can easily be CC'd (or BCC'd) in Emails.

NOTE: These examples are based on the supplied Account Letter sample.

For an Account Document, update the `CreateDocumentAccount` method as follows:

```
' Update di Email details if required
' NOTE: By default, these are set to the Email template details defined on this Document
Dim DealerClient As finClient
Dim DealerEmail As String

If di.Account.DealerExternalParty.GetClient(DealerClient) Then
  ' Get Dealer Email
  DealerEmail = DealerClient.ContactMethods.GetCurrentEmail().Value

  ' CC Dealer
  If Len(DealerEmail) <> 0 Then
    di.EmailDetails.EmailAddressesCc.Add(DealerEmail)
  End If
Else
  Success = False
End If

Return Success
```

This would CC the Dealer on the Email for each and every included Account Client.

Vanilla Client Document

This sample creates a basic Client Document that must be published from a Document Log.

It assumes that the supplied Client_Letter.docx document is embedded in the Document record and a parameter named 'LetterContent' is added to the Document.

Although this is a full Document Script, for clarity, much of the validation and option support included in the sample Client Letter (Client_Letter.xml) Document has been removed.

```
Option Explicit On
Option Strict On

' ################################################################
' Vanilla Client Letter
'
' Version: 1.00 (06/05/2016)
'
' Usage: Sample
' ################################################################

' Objects
Private mDocument As finDocumentRO

Public Function Main(source As Object,
                     eventId As String,
                     eventArgs As ISKeyValueList,
                     ByRef handled As Boolean,
                     parameters As Object,
                     ByRef returnValues As ISKeyValueList,
                     ByRef text As String) As Boolean

  Dim LogPublishBatchItem As finLogPublishBatchItem

  ' Assume Success
  Main = True

  ' Initialise
  mDocument = finBL.Documents(Mid(ScriptInfo.ScriptId, 10))

  ' Handle Events
  Select Case eventId
    Case "Publish"
      ' Get Publish Batch Item
      LogPublishBatchItem = DirectCast(source, finLogPublishBatch)(0)

      ' Publish
      Main = Publish_Log(LogPublishBatchItem)
  End Select

End Function

'#################################################################
' Publish via Log
'#################################################################
Private Function Publish_Log(logPublishBatchItem As finLogPublishBatchItem) As Boolean

  Dim Client As finClient
  Dim ClientLog As finClientLog
  Dim DocumentManagerFileName As String
  Dim DocumentManagerFileNameRelative As String
  Dim Success As Boolean
  Dim WordDocument As ISWordDocument
  Dim WordDocumentBookmarks As ISWordDocumentBookmarks

  ' Assume Success
  Success = True

  ' Load Client Log and Client
  Success = logPublishBatchItem.GetLog(ClientLog, Client)

  ' Get Word Template
  If Success Then
    Success = mDocument.CreateTemplateWordDocument(WordDocument)
  End If

  ' Create Word Document
  If Success Then
    ' Get Bookmarks
    WordDocumentBookmarks = WordDocument.GetBookmarks()

    ' Set Bookmarks
    With WordDocumentBookmarks
      ' Branch
      .SetContentString("BranchAddress", Client.Branch.GetAddressForLetters())
      .SetContentString("BranchName", Client.Branch.Name)
      .SetContentString("BranchPhone", Client.Branch.GetPhoneNumberForLetters())

      ' Client
      .SetContentString("ClientAddress", Client.ContactAddresses.GetAddressForLetters())
      .SetContentString("ClientSalutation", Client.SalutationResolved)

      ' General
      .SetContentDate("Date", ClientLog.LogDate, True)
      .SetContentString("DocumentId", mDocument.DocumentId)

      ' Parameters
      .SetContentString("LetterContent", ClientLog.Parameters.GetString("LetterContent"))
    End With

    ' Update Bookmarks
    WordDocument.UpdateBookmarks(WordDocumentBookmarks)
  End If

  ' Save to Document Manager as PDF
  If Success Then
    Success =
finBL.DocumentManagerFunctions.SaveWordDocumentToDocumentManagerPdfClient(Client.ClientId,
WordDocument, mDocument.FileName, DocumentManagerFileName, DocumentManagerFileNameRelative)
  End If

  ' Update Log Publish details
  If Success Then
    With logPublishBatchItem
      .PublishSuccess = True
      .PublishDocumentFileName = DocumentManagerFileNameRelative
      .PrintStatus = isefinLogPrintStatus.ToPrint
    End With
  End If

  Return Success

End Function
```

NOTE: The above sample does not use the UpdateStandardBookmarks method to update standard bookmarks; it sets all bookmarks manually.

Emailing the Document

Adding the following code before the 'Update Log Publish details' block will send the Document as a PDF Email attachment.

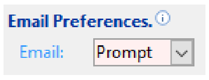
```
' Email Document?
If Success AndAlso ClientLog.DocumentSendEmailResolved Then
  Dim DocumentData As Byte()
  Dim EmailAddress As String
  Dim Request As ISMessageProviderRequest

  ' Get Email Address
  EmailAddress = Client.ContactMethods.GetCurrentEmail().Value

  If Len(EmailAddress) = 0 Then
    ' Client has no Email Address
  Else
    ' Convert Word Document to binary PDF data
    Success = WordDocument.SavePdfToByteArray(DocumentData)

    ' Create Email Request
    ' NOTE: Also resolve tags in Tempalate Email
    If Success Then
      Request = finBL.CreateMessageProviderRequest()
      With Request
        If mDocument.EmailHtml Then
```

```
          .MessageFormat = iseMessageFormat.Html
        Else
          .MessageFormat = iseMessageFormat.Text
        End If
        .MessageTarget = iseMessageTarget.Send
        .RecipientsTo.Add(EmailAddress)
        .Subject =
finBL.DocumentFunctions.ResolveTaggedMessageFromObjects(mDocument.TemplateSubject,
                                                                           ClientLog.LogDate,
                                                                           Nothing,
                                                                           Client)
        .Body = finBL.DocumentFunctions.ResolveTaggedMessageFromObjects(mDocument.TemplateMessage,
                                                                        ClientLog.LogDate,
                                                                        Nothing,
                                                                        Client)
        .Attachments.AddDataBinary("Letter.pdf", DocumentData)
      End With
    End If

    ' Send Email
    If Success Then
      Success = finBL.SendEmailByRequest(Request)
    End If

    ' Update Log to record Email Details
    ' NOTE: Ignore errors since Email has already been sent
    If Success Then
      ClientLog.ExtendedDataSetEmail(Request)
      ClientLog.Save()
    End If
  End If
End If
```

NOTE: The variable declarations are inside the `if` block. This is for clarity and typically they would appear at the beginning of the function.

Also, an `ISMessageRequest` object is used to send the Email since this allows most flexibility and does not 'hide' any functionality behind helper functions.

To give the User the option to Email or not, set the Email preference on the Documents form, Options page to either 'Yes' or 'Prompt':



Email Preferences.

Email: Prompt

Create Single Document but Email to All Account Clients

NOTE: The full Document Script code for this example is available in Appendix A.

Typically, when an Account Document is published, the following would occur:

- A list of recipients would be built using parameters defined on the Options page of the Document and on the Account Client (e.g., whether the Client is to receive Mail or not).
- A Word Document is generated for each item in this list and then emailed if required.
    - If multiple recipients are being emailed, multiple, child Account Logs may be created to record this.
        - Also, a bulk Document containing all child Account Log Documents is created and attached to the main Log.

However, if you simply want to create a single Document (that maybe lists all Clients) and sends an email to all of the Clients and also the Account's Dealer, this can be achieved quite simply as follows in this updated version of the standard `Publish_Log` method (the `CreateAccountDocument` method is removed completely):

Load the Account Log and Account

```
' Load Account Log
If Success Then
  AccountLog = finBL.CreateAccountLog()
  Success = AccountLog.Load(logPublishBatchItem.LogPk)
End If

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

Get the Main Account Client

Or, if this Account Log has been created from a Workflow Item targeting a particular Client, get this Account Client instead:

```
' Get 'Main' Account Client or Account Client specified in Log's linked Workflow Item
If Success Then
  If Account.Clients.GetItemFromWorfklowItemPk(AccountLog.WorkflowPk, AccountLog.WorkflowItemPk,
AccountClient) Then
    If AccountClient Is Nothing Then
      ' Get 'Main' Account Client
      AccountClient = Account.Clients(0)
    Else
      ' Account Client was retrieved from Workflow Item
    End If
  Else
    Success = False
  End If
End If
```

Create a Recipient Item for the Account Client

The Recipient Item contains a template Word Document and functionality to Email the Document.

```
' Create a Recipient Item for Account Client
If Success Then
  di = mDocument.GetAccountClientDocumentRecipientDetailsItem(AccountClient, AccountLog)

  ' Validate that an Email can be sent (i.e., this Account Client has an Email Address)
  If Not di.EmailDetails.CanSendEmail(strTemp) Then
    Success = False
    finBL.Error.ErrorBegin(strTemp)
  End If
End If
```

Append Further Email Addresses

Loop through all Account Clients and add to the correct To or CC address lists.

Also, append the Account Manager's Email Address.

```
' Append further Email Addresses
If Success Then
   ' For other Account Clients
  For Each AccountClient2 In Account.Clients
    If AccountClient2.SendDocuments AndAlso accountclient2 IsNot AccountClient Then
      ' Get Email Address
      strTemp = AccountClient2.Client.ContactMethods.GetCurrentEmail().Value

      If Len(strTemp) <> 0 Then
        Select Case AccountClient2.AccountRole.RoleType
          Case isefinAccountRoleType.Owner
            ' Owners
            di.EmailDetails.EmailAddresses.Add(strTemp)

          Case isefinAccountRoleType.Guarantor
            ' CC Guarantors
            di.EmailDetails.EmailAddressesCc.Add(strTemp)
        End Select
      End If
    End If
  Next

  ' BCC Account Manager
  strTemp = finBL.Users.ItemByPkSafe(Account.AccountManagerUserPk).Email
  If Len(strTemp) <> 0 Then di.EmailDetails.EmailAddressesBcc.Add(strTemp)
End If
```

Create the Document

Use the standard, unmodified `CreateDocument` method to create the Word Document.

```
' Create Document
If Success Then
  Success = CreateDocument(di)
End If
```

The rest of this method deals with saving the generated Document if necessary and, if Emailing failed, deleting the Document that was created from the Document Manager folder since this sample will fail the entire Publish process if an Email was unable to be sent.

Formatting Emails

Email content can be specified on the Document page of the Documents form. Alternatively, it can be dynamically created from within the Document Script.

NOTE: Unlike Email type Documents and when sending Emails from the "Send Email Message" wizard, no signature/ disclaimer is automatically added to the Email body and no special HTML formatting is added.

General

Smart Tags

When defining a template Email Subject and Message on the Document page of the Documents form, special tags can be inserted.

Available tags will depend on the type of Document (e.g., Account or Client) but there are certain tags that are available to ALL types of Document:

- `[Signature]`
  - Inserts the signature defined under User Preferences AND the disclaimer defined under Global Settings.
- `[SignatureUser]`
  - Inserts the signature defined under User Preferences.
- `[Disclaimer]`
  - Inserts the disclaimer defined under Global Settings or the Entity.

When generating Email content from code, the `finBL.SettingsUser.GetEmailSignature` method can be used to access the signature and disclaimer.

Plain Text

Plain Text Emails do not contain any special formatting and therefore, whatever text the Email Message is set to will be what the recipient sees.

NOTE: Different Email clients, e.g., MS Outlook vs Gmail may vary the font in which plain text Emails are displayed. This is often down to the recipient's preferences.

HTML

When HTML Emails are generated from the template Email Message defined on the Document, no additional HTML is added, i.e., no surrounding tags or font styles are added.

NOTE: Different Email clients, e.g., MS Outlook vs Gmail may vary the font in which HTML Emails are displayed.

Internally, the font that finPOWER Connect uses to display/ preview HTML Emails may not match what the recipient will actually receive.

Styles

WARNING: HTML support across various Email clients varies immensely, therefore it is important to test HTML Emails can be viewed correctly in as many Email clients as possible.

As a minimum, you probably want to set the font for HTML Emails. Email clients such as MS Outlook may use a default font such as "Time New Roman" which may not be what you expect.

CSS is one way to set the default font, e.g.:

```
<div style='font-family:Arial,Helvertica; font-size:10pt; color:navy'>
<p>Please review the attached letter.</p>

<p>[Signature]</p>

</div>
```

Also, always test the Global (or Entity) Email Disclaimer and User Signatures format correctly in HTML Emails (the plain text versions simply strip out HTML tags).

Images

'TODO:

Appendix A – Full Document Script Samples

All samples in this section contain full Document Script code which can be pasted into a Document.

Create Single Document but Email to All Account Clients

This code sample assumes that the supplied Account_Letter.docx document is embedded in the Document record.

Customised Script section is highlighted.

This sample demonstrates the following:

- Only ever creates a Document for the 'Main' Account Client.
  - Unless this Log has been created for a specific Account Client via a Workflow.
- Always Emails.
  - Ignores the setting on the Options page of Documents form.
- Respects many other Document options, e.g.:
  - Save the Document to the Document manager.
  - Embed the Document in the Account Log.

```
Option Explicit On
Option Strict On

' #################################################################
' Custom Account Letter to Email
'
' Version: 1.00 (05/05/2016)
'
' Usage:
' Account Document can only be published via a Log.
' #################################################################

' Objects
Private mDocument As finDocumentRO
Private mWordDocuments As List(Of ISWordDocument)
Private mWordDocumentsAccount As List(Of ISWordDocument)

Public Function Main(source As Object, eventId As String, eventArgs As ISKeyValueList, ByRef
handled As Boolean, parameters As Object, ByRef returnValues As ISKeyValueList, ByRef text As
String) As Boolean

  Dim Success As Boolean

  ' Assume Success
  Success = True

  ' Initialise
  mDocument = finBL.Documents(Mid(ScriptInfo.ScriptId, 10))
  mWordDocuments = New List(Of ISWordDocument)
  mWordDocumentsAccount = New List(Of ISWordDocument)

  ' Get Constants

  ' Handle Events
  Select Case eventId
    Case "Publish"
      ' Publish
      If TypeOf source Is finLogPublishBatch Then
        ' Log Publish
        Success = Publish_Log(DirectCast(source, finLogPublishBatch)(0))
      Else
        ' Standalone (Ad-Hoc)
        Success = False
        finBL.Error.ErrorBegin("This Document does not support Standalone publishing and must be
published from a Log.")
      End If
  End Select

  Return Success

End Function

'#################################################################
' Publish via Log
'#################################################################
Private Function Publish_Log(logPublishBatchItem As finLogPublishBatchItem) As Boolean
```

```
  Dim Account As finAccount
  Dim AccountClient As finAccountClient
  Dim AccountClient2 As finAccountClient
  Dim AccountLog As finAccountLog
  Dim di As finDocumentRecipientDetailsItem
  Dim DocumentBinaryData As Byte()
  Dim DocumentManagerFileNameRelative As String
  Dim DocumentManagerFileName As String
  Dim strTemp As String
  Dim Success As Boolean

  ' Assume Success
  Success = True

  ' Load Account Log
  If Success Then
    AccountLog = finBL.CreateAccountLog()
    Success = AccountLog.Load(logPublishBatchItem.LogPk)
  End If

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

  ' Get 'Main' Account Client or Account Client specified in Log's linked Workflow Item
  If Success Then
    If Account.Clients.GetItemFromWorfklowItemPk(AccountLog.WorkflowPk, AccountLog.WorkflowItemPk,
AccountClient) Then
      If AccountClient Is Nothing Then
        ' Get 'Main' Account Client
        AccountClient = Account.Clients(0)
      Else
        ' Account Client was retrieved from Workflow Item
      End If
    Else
      Success = False
    End If
  End If

  ' Create a Recipient Item for Account Client
  If Success Then
    di = mDocument.GetAccountClientDocumentRecipientDetailsItem(AccountClient, AccountLog)

    ' Validate that an Email can be sent (i.e., this Account Client has an Email Address)
    If Not di.EmailDetails.CanSendEmail(strTemp) Then
      Success = False
      finBL.Error.ErrorBegin(strTemp)
    End If
  End If

  ' Append further Email Addresses
  If Success Then
     ' For other Account Clients
    For Each AccountClient2 In Account.Clients
      If AccountClient2.SendDocuments AndAlso accountclient2 IsNot AccountClient Then
        ' Get Email Address
        strTemp = AccountClient2.Client.ContactMethods.GetCurrentEmail().Value

        If Len(strTemp) <> 0 Then
          Select Case AccountClient2.AccountRole.RoleType
            Case isefinAccountRoleType.Owner
              ' Owners
              di.EmailDetails.EmailAddresses.Add(strTemp)

            Case isefinAccountRoleType.Guarantor
              ' CC Guarantors
              di.EmailDetails.EmailAddressesCc.Add(strTemp)
          End Select
        End If
      End If
    Next

    ' BCC Account Manager
    strTemp = finBL.Users.ItemByPkSafe(Account.AccountManagerUserPk).Email
    If Len(strTemp) <> 0 Then di.EmailDetails.EmailAddressesBcc.Add(strTemp)
  End If
  ' Create Document
  If Success Then
    Success = CreateDocument(di)
  End If

  ' Save Document in Document manager
  If Success AndAlso mDocument.AutoSaveDocument Then
    Success =
finBL.DocumentManagerFunctions.SaveWordDocumentsToDocumentManagerPdfAccount(di.AccountId,
mWordDocumentsAccount, mDocument.FileName, DocumentManagerFileName,
DocumentManagerFileNameRelative)
  End If

  ' Embed Document in Log?
  If Success AndAlso mDocument.EmbedDocumentInLog Then
    Success = finBL.PdfUtilities.CombineWordDocumentsToPdfByteArray(mWordDocumentsAccount,
DocumentBinaryData)
  End If

  ' Send Email
  If Success Then
    Success = di.EmailDetails.SendEmailPdf()
  End If

  ' Remove Saved Document if failed (since this Document is designed only to be Emailed)
  If Not Success AndAlso Len(DocumentManagerFileName) <> 0 Then
    finBL.Runtime.FileUtilities.DeleteFile(DocumentManagerFileName)
  End If

  ' Update Log Publish details
  If Success AndAlso logPublishBatchItem IsNot Nothing Then
    With logPublishBatchItem
      ' Success
      .PublishSuccess = True
      .PublishDocumentFileName = DocumentManagerFileNameRelative
      .PublishDocumentBinaryDataFileType = isefinLogEmbeddedFileType.Pdf
      .PublishDocumentBinaryData = DocumentBinaryData
      .PrintStatus = di.PrintStatus
    End With
  End If

  ' Record Email details on Account Log
  If Success AndAlso di.EmailDetails.EmailSent Then
    If Not di.EmailDetails.AccountLogUpdateForEmailSend(logPublishBatchItem.LogPk) Then
      ' Ignore Errors
    End If
  End If

  Return Success

End Function

'#################################################################
' Create Document
'#################################################################
Private Function CreateDocument(di As finDocumentRecipientDetailsItem) As Boolean

  Dim Account As finAccount
  Dim AccountLog As finAccountLog
  Dim Reference As String
  Dim Success As Boolean
  Dim WordDocumentBookmarks As ISWordDocumentBookmarks

  ' Assume Success
  Success = True

  ' Initialise
  Account = di.Account
  AccountLog = DirectCast(di.Log, finAccountLog)

  ' Create Word Document for this Account Client and get Bookmarks
  WordDocumentBookmarks = di.WordDocument.GetBookmarks()

  ' Reference
  With finBL.AccountRoles(di.AccountClient.AccountRoleId)
    Select Case .RoleType
      Case isefinAccountRoleType.Other
      Case Else
        Reference = "As " & .DescriptionShortResolved & " for "
    End Select
    Reference &= "Account " & di.AccountId
    If finBL.Settings.AddOns.ExternalParty AndAlso di.Account.DealerExternalParty IsNot Nothing
Then
      Reference &= ", with " & di.Account.DealerExternalParty.Name & "."
    End If
  End With

  ' Set Bookmarks
  With WordDocumentBookmarks
    ' Branch
    .SetContentString("BranchAddress", di.BranchAddressForLetters)
    .SetContentString("BranchName", di.BranchName)
    .SetContentString("BranchPhone", di.BranchPhoneNumber)

    ' Client
    .SetContentString("ClientAddress",
di.PostalDetails.AddressForAccountLettersIncludingAddressee)
    .SetContentString("ClientSalutation", di.SalutationAccount)

    ' General
    .SetContentDate("Date", di.DocumentDate, True)
    .SetContentString("DocumentId", mDocument.DocumentId)
    .SetContentString("Reference", Reference)
  End With

  ' Update Bookmarks
  di.WordDocument.UpdateBookmarks(WordDocumentBookmarks)

  ' Record Word Document
  mWordDocuments.Add(di.WordDocument)
  mWordDocumentsAccount.Add(di.WordDocument)

  Return Success

End Function
```

Create Client Excel Workbook

This sample demonstrates the following:

- Create an Excel Workbook for a Client.

```
Option Explicit On
Option Strict On

' ###################################################################
' Client Workbook
'
' Version: 1.00 (23/05/2016)
'
' Usage: Sample
' ###################################################################

' Constants

' Objects
Private mDocument As finDocumentRO
Private mExcelWorkbooks As List(Of ISExcelWorkbook)

' Other
Private mBulkDocumentFileName As String

Public Function Main(source As Object, eventId As String, eventArgs As ISKeyValueList, ByRef
handled As Boolean, parameters As Object, ByRef returnValues As ISKeyValueList, ByRef text As
String) As Boolean

  Dim Success As Boolean

  ' Assume Success
  Success = True

  ' Initialise
  mDocument = finBL.Documents(Mid(ScriptInfo.ScriptId, 10))
  mExcelWorkbooks = New List(Of ISExcelWorkbook)

  ' Get Constants
  With ScriptInfo.Constants
  End With

  ' Handle Events
  Select Case eventId
    Case "Publish"
      ' Publish
      If TypeOf source Is finLogPublishBatch Then
        ' Log Publish
        Success = Publish_Log(DirectCast(source, finLogPublishBatch)(0))
      Else
        ' Standalone (Ad-Hoc)
        Success = False
        finBL.Error.ErrorBegin("Standalone not supported.")
      End If
  End Select

  Return Success

End Function

'###################################################################
' Publish via Log
'###################################################################
Private Function Publish_Log(logPublishBatchItem As finLogPublishBatchItem) As Boolean

  Dim di As finDocumentRecipientDetailsItem
  Dim DocumentBinaryData As Byte()
  Dim DocumentManagerFileName As String
  Dim DocumentManagerFileNameRelative As String
  Dim Success As Boolean

  ' Assume Success
  Success = True

  ' Validate

  ' Get Document Recipient Details for Client
  If Success Then
```

```
    Success =
mDocument.GetClientDocumentRecipientDetailsItemFromLogPublishBatchItem(logPublishBatchItem, di)
  End If

  ' Create Document
  If Success Then
    Success = CreateDocument(di)
  End If

  ' Save Document to Document Manager?
  If Success AndAlso mDocument.AutoSaveDocument Then
    Success =
finBL.DocumentManagerFunctions.SaveExcelWorkbookToDocumentManagerXlsxClient(di.ClientId,
di.ExcelWorkbook, mDocument.FileName, DocumentManagerFileName, DocumentManagerFileNameRelative)
  End If

  ' Embed in Log?
  If Success AndAlso mDocument.EmbedDocumentInLog Then
    Success = di.ExcelWorkbook.SaveXlsxToByteArray(DocumentBinaryData)
  End If

  ' Update Log Publish details
  If Success Then
    With logPublishBatchItem
      .PublishSuccess = True
      .PublishDocumentFileName = DocumentManagerFileNameRelative
      .PublishDocumentBinaryDataFileType = isefinLogEmbeddedFileType.Xlsx
      .PublishDocumentBinaryData = DocumentBinaryData
      .PrintStatus = di.PrintStatus
    End With
  End If

  ' Record Email details
  If Success AndAlso di.EmailDetails.EmailSent Then
    If Not di.EmailDetails.ClientLogUpdateForEmailSend(logPublishBatchItem.LogPk) Then
      ' Ignore Errors
    End If
  End If

  Return Success

End Function

'##################################################################
' Create Document
'##################################################################
Private Function CreateDocument(di As finDocumentRecipientDetailsItem) As Boolean

  Dim Success As Boolean

  ' Assume Success
  Success = True

  ' Record Excel Workbook
  If Success Then
    mExcelWorkbooks.Add(di.ExcelWorkbook)
  End If

  ' Update di.EmailDetails if required
  ' NOTE: By default, these are set to the Email template details defined on this Document

  ' Email Document?
  If di.EmailDetails.PublishEmail Then di.EmailDetails.SendEmailXlsx()

  Return Success

End Function
```

Appendix B – Standard Bookmarks

These sections list all standard bookmarks that are replaced by the `finDocumentRecipientDetailsItem.UpdateStandardBookmarks` method.

NOTE: This list will likely grow over time so it is important that any custom, document-specific bookmarks are always replaced AFTER standard bookmarks to avoid any naming conflicts.

Although bookmark names cannot contain dots (e.g., AccountName, BranchAddress), typically, the content of the bookmark should be enclosed in square brackets with a dot in between the object and property name, e.g.:

```
[Account.Name]

[Branch.Address]
```

Document

These bookmarks are always prefixed "Document".

Name	Version	Description
CurrentDate	3.00.04	The current database Date.
CurrentDateTime	3.00.04	The current database Date and Time
Date	3.00.00	
DocumentId	3.00.00	
SignatureBlock	3.00.04	A standard Signature Block, including Witness.
SignatureBlock1 to SignatureBlock.9	3.00.04	One or more standard Signature Blocks.
SignatureBlockSBS	3.00.08	A side-by-side Signature Block.
SignatureBlockSBS1 to SignatureBlockSBS.9	3.00.08	One or more side-by-side Signature Blocks.

Account

These bookmarks are always prefixed "Account".

Name	Version	Description
AccountId	3.00.00	
AccountType	3.00.00	The Account Type's description.
AccountTypeId	3.00.00	
AmountFinanced	3.01.08	Net Advanced + Costs.
BorrowerNames	3.00.00	A list of borrower (including co-borrower) names.
BusinessDevelopmentManager.* or BDM.*	3.02.03	User information for the Business Development Manager. For example, "Account.BDM.Name".
ClientBlock	3.00.03	A table of Clients, including address details.
ClientBlockWithGuarantors	3.00.03	As per Client Block, but lists Guarantors.

CurrentInterestRate	3.01.08	The Interest Rate as at the document date.
CreditLimit	3.00.05	The Credit Limit
CreditLimit1	3.00.05	Same as CreditLimit
CreditLimit2	3.00.05	Credit Limit 2
CreditLimit3	3.00.05	Credit Limit 3
DateFinalPayment	3.00.04	The date of a Loan Account's final payment.
DateMaturity	3.00.00	
DateOpened	3.00.00	
Description	3.00.00	The Account's description.
Domain Or Dm	3.00.02	The text "Account" or "Application".
ESignatureBlock	3.00.02	Signature Block for Electronic Signatures.
FinalPaymentDate	4.00.00	Alias of DateFinalPayment
FirstRegularPaymentDate	4.00.00	The date of the first regular using the current Loan schedule
FirstRegularPaymentValue	4.00.00	The value of the first regular using the current Loan schedule
FinalRegularPaymentDate	4.00.00	The date of the final regular using the current Loan schedule
FinalRegularPaymentValue	4.00.00	The value of the final regular using the current Loan schedule
InvestmentValue	3.00.08	
LastPaymentDate	3.00.07	
LastPaymentValue	3.00.07	
LastPaymentDueDate	3.03.05	
LastPaymentDueValue	3.03.05	
Manager.*	3.02.03	User information for the Manager. For example, "Account.Manager.Name".
MaturityInstruction	3.00.08	
Name	3.00.02	
NetAdvance	3.01.08	Total Advances + Refinances – Deposits.
NextPaymentDate	3.00.07	
NextPaymentValue	3.00.07	
OriginalFirstRegularPaymentDate	4.00.00	The date of the first regular using the original Loan schedule
OriginalFirstRegularPaymentValue	4.00.00	The value of the first regular using the original Loan schedule
OriginalFinalRegularPaymentDate	4.00.00	The date of the final regular using the original Loan schedule

OriginalFinalRegularPaymentValue	4.00.00	The value of the final regular using the original Loan schedule
PaymentBalloon	3.02.00	
PaymentCycle	3.01.08	Description of the current Payment Cycle.
PayIDContactEmail	4.00.03	The Zepto PayID Contact Email.
PayIDContactName	4.00.03	The Zepto PayID Contact Name.
PayIDEmail	4.00.03	The Zepto PayID Email.
PaymentRegular	3.02.00	
Reference	3.00.00	Standard reference text, e.g., "John Smith As Borrower For Account L10000, with The Dealer"
ResidualValue	3.02.00	
RoleId	3.00.00	The Id of the Account Client's Role. See also AccountClient.RoleId.
RoleDescription	3.00.00	The Description of the Account Client's Role. See also AccountClient.RoleDescription.
Salutation	3.00.00	Salutation for letters. This may be the Client's salutation or a combined salutation for Joint Clients.
Security	3.00.03	A table of Security Items for the Account. Excludes defunct items.
SecurityPhysical	3.00.03	A table of Security Items for the Account. Only includes non-Defunct, Physical items.
SecurityPhysicalWithLocation	3.00.05	A table of Security Items, including Location information, for the Account. Only includes non-Defunct, Physical items.
SecurityTradeIn	3.00.03	A table of Security Items for the Account. Only includes non-Defunct, Trade-In items.
SignatureBlock	3.00.02	Signature Block for Clients.
StatementCycle	3.00.04	The current Statement Frequency.
Term	3.00.08	
UserData:[Key]	3.04.02	User Data. The Key defines the item to use. For example: [Account.UserData:ExtraDataKey]

Account Client

These bookmarks are always prefixed "AccountClient".

Name	Version	Description
ClientId	3.00.03	
Name	3.00.03	
RoleId	3.00.03	
RoleDescription	3.00.03	

Account Log

These bookmarks are always prefixed "AccountLog".

Name	Version	Description
Balance	3.00.05	The Account's Balance as recorded on the Log.
BalanceOverdue	3.00.00	The Account's Overdue Balance as recorded on the Log.
BalanceOverdueContractual or BalanceOverdueC	3.01.00	The Account's Contractual Overdue Balance as recorded on the Log.
ExternalParty.*	3.00.05	The External Party assigned to the Log. For example, "AccountLog.ExternalParty.Name".
Fee	3.00.00	
FeeResolved	3.03.01	This is the Fee reduced to the remaining Cost of Borrowing or Default Fee balance under NZ High-Cost Loan regulations.
OverdueBlock	3.00.03	Adds a table of Overdue values.
OverdueContractualBlock or BalanceCBlock	3.01.00	Adds a table of Contractual Overdue values.
PayIDContactEmail	4.00.03	The Zepto PayID Contact Email.
PayIDContactName	4.00.03	The Zepto PayID Contact Name.
PayIDEmail	4.00.03	The Zepto PayID Email.
UserData:[Key]	3.04.02	User Data. The Key defines the item to use. For example: [AccountLog.UserData:ExtraDataKey]

Account Application

These bookmarks are always prefixed "AccountApp".

NOTE: Both Account and Client standard bookmarks are also available to Account Applications since Account Applications have the ability to create temporary (unsaved) Account and Client objects.

Name	Version	Description
AccountAppId	3.00.00	
Description	3.00.00	The Account Application's description.
UserData:[Key]	3.04.02	User Data. The Key defines the item to use. For example: [AccountApp.UserData:ExtraDataKey]

Account Type

These bookmarks are always prefixed "AccountType".

Name	Version	Description
AccountTypeId	3.00.04	The Account Type's Id. This is the same as Account.AccountTypeId.
Description	3.00.04	The Account Type's Description.
EarlySettlementFee	3.00.04	The Fee charged on an Early Settlement.
UserData:[Key]	3.04.02	User Data. The Key defines the item to use. For example: [AccountType.UserData:ExtraDataKey]

Branch

These bookmarks are always prefixed "Branch".

Name	Version	Description
Address	3.00.00	
BranchId	3.00.00	
Email	3.00.00	
Facsimile or Fax	3.00.00	
Logo	3.00.03	The Branch Logo Image or URL. Use a colon to specify the height in points, e.g.: [Branch.Logo:40]
Logo2	3.04.00	The Branch Logo 2 Image or URL. Use a colon to specify the height in points, e.g.: [Branch.Logo2:40]
LegalAndTradingName	3.00.04	The Legal Name and Trading Name if defined.
LegalAndTradingNameAndAddress	3.00.04	The Legal Name, Trading Name if defined, and Address.
LegaName	3.00.04	The Legal Name (or Name if Legal Name undefined).
LegalNameAndAddress	3.00.04	The Legal Name (or Name if Legal Name undefined) and Address.
Name	3.00.00	
NameAndAddress	3.00.00	
Phone or PhoneNumber	3.00.00	
PhysicalAddress	3.00.03	
PostalAddress	3.00.03	
PostalNameAndAddress	3.00.03	
ResourceLogo	3.04.00	The Branch Resource Logo as a PNG. Use a colon to specify the height in points, e.g.: [Branch.ResourceLogo:40]
ResourceOther	3.04.00	The Branch Resource 'Other' as a PNG. Will only display if the Resource is an image type.

		Use a colon to specify the height in points, e.g.: [Branch.ResourceOther:40]
UserData:[Key]	3.04.02	User Data. The Key defines the item to use. For example: [Branch.UserData:ExtraDataKey]

Broker, Dealer, Insurer

These bookmarks are always prefixed by the External Party Type, e.g. "Broker", "Dealer" or "Insurer".

Name	Version	Description
BusinessNumber or ABN, NZBN	3.00.00	
CompanyNumber or ACN	3.00.00	
Description	3.00.00	
Email	3.00.00	
Facsimile or Fax	3.00.00	
LegalName	3.00.00	
Mobile	3.00.00	
Name	3.00.00	
Phone or PhoneNumber	3.00.00	
PhysicalAddress	3.00.00	
PhysicalNameAndAddress	3.00.00	
PostalAddress	3.00.00	
PostalNameAndAddress	3.00.00	
Salutation	3.00.00	
TaxNumber	3.00.00	
UserData:[Key]	3.04.02	User Data. The Key defines the item to use. For example: [Dealer.UserData:ExtraDataKey]

Client

These bookmarks are always prefixed "Client".

Name	Version	Description
Address	3.00.00	
BusinessNumber or ABN, NZBN	3.00.00	
ClientId	3.00.00	
CompanyNumber or ACN	3.00.00	
DateOfBirth	3.00.08	
Email	3.00.00	

Facsimile or Fax	3.00.00	
FirstName	3.00.06	
LastName	3.00.06	
LegalName	3.00.00	
MiddleNames	3.00.06	
Mobile	3.00.00	
Name	3.00.00	The Client's Name, e.g., "Smith, John"
NameAndAddress	3.00.00	
Phone or PhoneNumber	3.00.00	
PhoneHome	4.00.01	
PhoneSms	4.00.01	The first Phone Number found that supports text messaging
PhoneWork	4.00.01	
PhysicalAddress	3.00.00	
PhysicalNameAndAddress	3.00.00	
PostalAddress	3.00.00	
PostalNameAndAddress	3.00.00	
PreferredName	3.03.00	
Salutation	3.00.00	
TaxNumber	3.00.00	
Title	3.00.06	
UserData:[Key]	3.04.02	User Data. The Key defines the item to use. For example: [Client.UserData:ExtraDataKey]

CurrentUser

The Currently Logged-In User.

These bookmarks are always prefixed "CurrentUser".

Name	Version	Description
Email	3.00.02	
Facsimile or Fax	3.00.02	
Mobile	3.00.02	
Name	3.00.02	The Client's Name, e.g., "Smith, John"
Phone or PhoneNumber	3.00.02	
UserData:[Key]	3.04.02	User Data. The Key defines the item to use. For example: [CurrentUser.UserData:ExtraDataKey]
UserId	3.00.02	

Entity

These bookmarks are always prefixed "Entity".

Name	Version	Description
BusinessNumber or ABN, NZBN	3.00.00	
CompanyNumber or CAN	3.00.00	
DisputeResolutionAddress	3.00.03	
DisputeResolutionEmail	3.00.03	
DisputeResolutionFacsimile or DisputeResolutionFax	3.00.03	
DisputeResolutionName	3.00.03	
DisputeResolutionPhone DisputeResolutionPhoneNumber	3.00.03	
DisputeResolutionWebsite	3.00.03	
DocumentContent1	3.04.02	Document Content 1 as defined on the Entity. Will be inserted as HTML.
DocumentContent2	3.04.02	Document Content 2 as defined on the Entity. Will be inserted as HTML.
EntityId	3.00.00	
LicenceNumber	3.00.00	
LicenceName	3.00.00	
Logo	3.04.00	The Entity Logo Image or URL. Use a colon to specify the height in points, e.g.: [Entity.Logo:40]
Logo2	3.04.00	The Entity Logo 2 Image or URL. Use a colon to specify the height in points, e.g.: [Entity.Logo2:40]
Name	3.00.00	
ResourceLogo	3.04.00	The Entity Resource Logo as a PNG. Use a colon to specify the height in points, e.g.: [Entity.ResourceLogo:40]
ResourceOther	3.04.00	The Entity Resource ‘Other’ as a PNG. Will only display if the Resource is an image type. Use a colon to specify the height in points, e.g.: [Entity.ResourceOther:40]
TaxNumberName	3.00.08	The localised name used in the database for the Tax Number. E.g. in NZ this is IRD Number.
UserData:[Key]	3.04.02	User Data. The Key defines the item to use. For example: [Entity.UserData:ExtraDataKey]

Statement

Loan or Deposit Statement details.

These bookmarks are always prefixed "Statement".

Name	Version	Description
Balance	3.00.03	The Closing Balance of the Statement.

BalanceOverdue or OD	3.00.03	The Overdue Balance as at the Statement date.
BalanceOverdueLast or ODLast	3.00.03	The Overdue Balance as at the Last Statement date.
BalanceOverdueOrPrepaid or ODP	3.03.03	The Overdue (positive) or Prepaid (negative) Balance as at the Statement date.
BalanceOverdueThis or ODThis	3.00.03	The Overdue Balance change for this Statement, i.e. BalanceOverdue less BalanceOverdueLast.
CreditLimit	3.00.03	The Credit Limit as at the Statement date.
Credits	3.00.03	Total Credit Transactions.
DateFrom	3.00.03	
DateTo	3.00.03	
Debits	3.00.03	Total Debit Transactions.
DirectDebitPaymentDate or DDDate	3.00.03	Direct Debit payment date.
DirectDebitPaymentValue or DDValue	3.00.03	Direct Debit payment value.
DueDate or Due	3.00.03	The Date the next Payment/ Minimum Payment is due to be paid.
DueNow	3.00.03	The Value that is currently overdue and therefore due immediately. This is the same as BalanceOverdue.
DueValue	3.00.03	The Value to pay when next due.
FinalNotice	3.00.03	A simple notice if this is the Final Statement.
MinimumPaymentDate or MinDate	3.00.03	Minimum Payment date due.
MinimumPaymentValue or MinValue	3.00.03	Minimum Payment value.
NextPaymentDate or NextDate	3.00.03	Next Payment date due.
NextPaymentValue or NextValue	3.00.03	Next Payment value.
Opening	3.00.03	The Opening Balance of the Statement.
Transactions	3.00.03	A table of Transactions, including columns Date, Reference, Description, Debit, Credit and Balance.
TransactionsWithGrossWTaxNet	3.00.03	A table of Transactions suitable for Deposits. Columns include Date, Reference, Description, Gross, Withholding Tax, Net and Balance.
UserData:[Key]	3.04.02	User Data. The Key defines the item to use. For example: [Statement.UserData:ExtraDataKey]

TaxCertificate

Tax Certificate details, from a Client Log.

These bookmarks are always prefixed "TaxCertificate" or "TC".

Name	Version	Description
AverageWTaxRate	3.00.08	The Average Withholding Tax Rate.
CurrentWTaxRate	3.00.08	The Current Withholding Tax Rate (as at Date).
Date	3.00.08	The Date of the Tax Certificate. This is the same as TaxYearEnded.
Branch.*	3.00.08	The Tax Certificate's Branch. For example, "TaxCertificate.Branch.LegalName".
ClientId	3.00.08	The Id of the Client.
DateOfBirth	3.00.08	The Date of Birth of the Client from the Tax Certificate.
Entity.*	3.00.08	The Tax Certificate's Entity. For example, "TaxCertificate.Entity.TaxNumber".
Investments	3.00.08	A table of Deposit Accounts.
Name	3.00.08	The Name of the Client from the Tax Certificate.
Rates	3.01.09	A table of Rates used during the year.
TaxNumber	3.00.08	
TaxYearEnded	3.00.08	The Year End the Tax Certificate relates to.
Title	3.00.08	The Title of the Tax Certificate.
TotalGross	3.00.08	
TotalNet	3.00.08	
TotalWithHoldingTax	3.00.08	

User

User information (for the Log, Account Manager etc).

These bookmarks are always prefixed "User".

Name	Version	Description
Email	3.00.00	
Facsimile or Fax	3.00.00	
JobTitle	3.02.03	
Mobile	3.00.00	
Name	3.00.00	
Phone or PhoneNumber	3.00.00	
Salutation	3.00.00	
UserData:[Key]	3.04.02	User Data. The Key defines the item to use. For example: [User.UserData:ExtraDataKey]
UserId	3.00.00	

Appendix C – Converting an Existing MS Word VBA Template

When converting an existing MS Word VBA Template (.dot file) for use as a template for a "Word Document", the following steps should be taken:

- Open the VBA template (.dot file).
- Remove all VBA code modules.
  - NOTE: Ensure you save the original VBA code (including code in the `ThisDocument` module) for reference when creating the Document Script.
- Tidy up bookmarks, e.g., remove Document Start indicators and rename bookmarks where applicable as per the Bookmark Naming section.
- Tidy up tables as per the Tables section.
- Save the file as an MS Word Document (.docx file).

The MS Word Document is now ready to be used as a template for a finPOWER Connect Word Document.

WARNING: If the existing MS Word document has complicated formatting, it might not be possible for this formatting to be replicated by the GemBox component used by finPOWER Connect.

Creating a new .docx file and pasting in portions of the original document may fix or improve formatting issues.

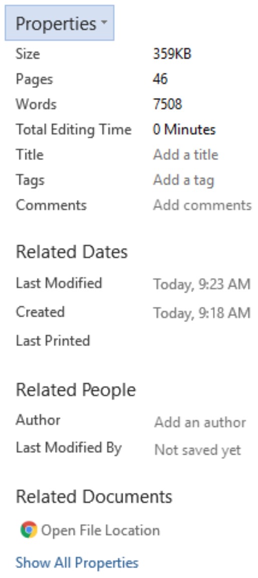
Appendix D – Removing MS Word Personalisation Details

Word holds meta data with each document. This includes information such as Author and often a history or changes made to the document.

It is important to remove or tidy up this information before finalising a Word document.

The steps below show how to achieve this in Word 2013:

1. Open the Word document.
2. From 'File', select 'Info':



Properties

Size	359KB
Pages	46
Words	7508
Total Editing Time	0 Minutes
Title	Add a title
Tags	Add a tag
Comments	Add comments

Related Dates

Last Modified	Today, 9:23 AM
Created	Today, 9:18 AM
Last Printed	

Related People

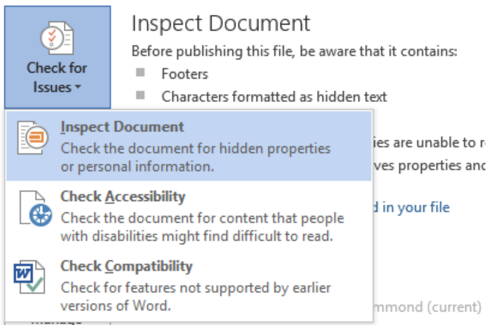
Author	Add an author
Last Modified By	Not saved yet

Related Documents

Open File Location

Show All Properties

3. Click the "Show All Properties" link.
4. The above screen shot shows a clean document but often, this list of properties will contain information not related to the document which should be cleaned up.
5. The clear this information, click 'Check for Issues' and then 'Inspect Document':



Check for Issues

Inspect Document

Before publishing this file, be aware that it contains:

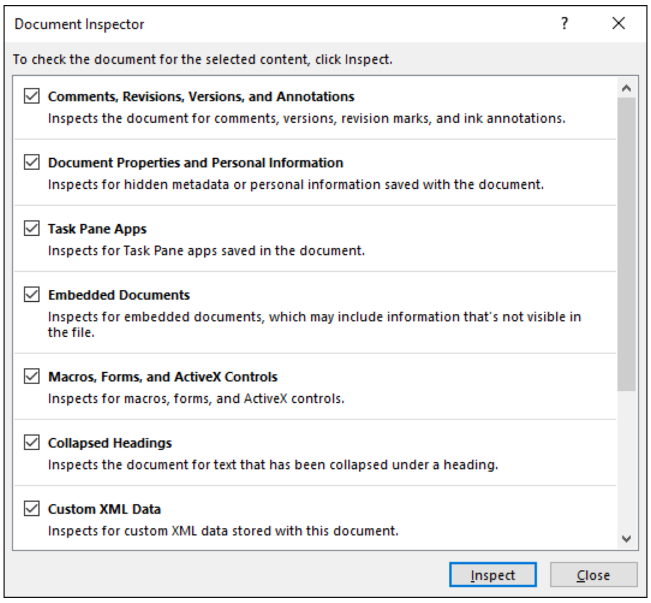
- Footers
- Characters formatted as hidden text

Inspect Document
Check the document for hidden properties or personal information.

Check Accessibility
Check the document for content that people with disabilities might find difficult to read.

Check Compatibility
Check for features not supported by earlier versions of Word.

6. This displays the 'Document Inspector' form:


Document Inspector

To check the document for the selected content, click Inspect.

☑ **Comments, Revisions, Versions, and Annotations**
Inspects the document for comments, versions, revision marks, and ink annotations.

☑ **Document Properties and Personal Information**
Inspects for hidden metadata or personal information saved with the document.

☑ **Task Pane Apps**
Inspects for Task Pane apps saved in the document.

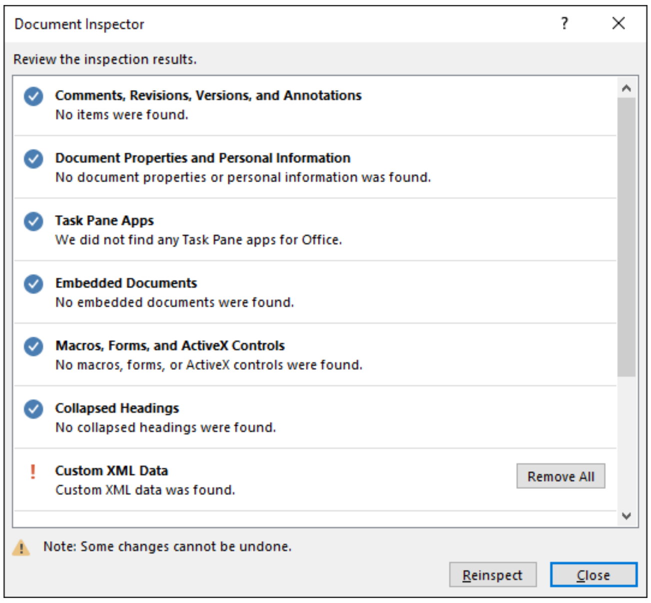
☑ **Embedded Documents**
Inspects for embedded documents, which may include information that's not visible in the file.

☑ **Macros, Forms, and ActiveX Controls**
Inspects for macros, forms, and ActiveX controls.

☑ **Collapsed Headings**
Inspects the document for text that has been collapsed under a heading.

☑ **Custom XML Data**
Inspects for custom XML data stored with this document.

7. Click 'Inspect'.
8. This then shows a list of information types contained in the document and has a 'Remove All' button against each option. This can be used to quickly clear out information such as Revisions and document properties.


Document Inspector

Review the inspection results.

Comments, Revisions, Versions, and Annotations
No items were found.

Document Properties and Personal Information
No document properties or personal information was found.

Task Pane Apps
We did not find any Task Pane apps for Office.

Embedded Documents
No embedded documents were found.

Macros, Forms, and ActiveX Controls
No macros, forms, or ActiveX controls were found.

Collapsed Headings
No collapsed headings were found.

! **Custom XML Data**
Custom XML data was found.

Note: Some changes cannot be undone.

Appendix E – Troubleshooting

Word Documents

- Bookmarks:
    - Make sure bookmark content and naming conventions are used.
    - Make sure all bookmark content that is meant to be replaced or deleted is fully inside the bookmark.
    - If Bookmarks duplicate content.
        - Ensure actual bookmarks are not doubled-up and/ or on top of each other.
        - Using consistent naming conventions helps.
        - Check Bookmarks by opening the Bookmarks dialog and "Go To" each Bookmark.
    - Using in Headers and Footers with Section Breaks.
        - If bookmarks in section 2 and later do not work, try unchecking "Link to Previous" in Header/ Footer properties.
    - Using in Tables.
        - Avoid using a Bookmark covering the entire first or last Column in a Table.
            - This type of Bookmark is not currently supported.
            - Change to a Bookmark that just covers the Cell's content.
- Tables:
    - Tidy up tables as per the Tables section.
    - Under Tables Properties.
        - Set "Text Wrapping" as "None" – do not use "Around".
    - Combine Tables by moving together.
    - Split using Table Layout menu.
    - If tables don't line up.
        - Check cell margins are all zero and use Paragraph formatting to add spacing before and after cell text.
    - Heading Rows for each page do not work check:
        - Make sure Table Text Wrapping is set to "None".
        - Make sure there are no Page Breaks in the Table.
        - Make sure you are not using nested tables.
- Shapes:
    - Grouped Shapes are not supported.
    - Shapes anchored inside a table are not supported.
- Pagination.
    - Paragraph Pagination option "Keep with next" may work better than "Keep lines together", especially where in a table.
- Fonts
    - Fonts can be an issue when printing and/ or viewing documents outside a simple Windows environment.
    - Font used is not as expected.
        - It may look ok in Word, but when saved to PDF the font is incorrect.
        - If in an inserted HTML block, try specifically setting the font in the block.
    - If possible, stick to very basic fonts.

        - E.g. Arial/ Helvetica, Times New Roman, Courier New/ Courier, Verdana.
        - Web Safe Fonts.
        - See http://www.w3schools.com/cssref/css_websafe_fonts.asp.
        - Palatino, Garamond, Bookman, Avant Garde.
        - Windows/ Mac but not Unix include Verdana, Georgia, Comic Sans, Trebuchet, Arial Black and Impact.
    - PDF has very few "built-in" fonts.
        - Courier, Helvetica, Time-Roman and Symbol.
        - PDF/A specification states all fonts must be embedded.
    - If you use any non-standard fonts make sure you test them on target environment.
        - Both the environment where the file is created and where is will be viewed.
        - E.g. can you install a font on a web server?
    - Embedded Fonts.
        - Fonts may be embedded in Word Documents. However, this is not currently supported, but will preserve the font if the document if saved as a Word Document.
        - In a PDF the embedded font is lost.
    - Private Fonts.
        - Use where fonts cannot be "installed" on a computer.
        - The `ISWordDocument` object includes a `FontsFolder` property.
            - Set this to the folder where private font files are stored.
        - PDFs will embed special fonts used and found in this folder.
    - Issues where Fonts are not being correctly embedded.
        - Occasionally a font may not be embedded in a Word or PDF document.
        - This may occur because there is a difference in the font name verses the font file name.
        - Use the `UpdateFont` method in `ISWordDocument` to quickly rename fonts.
            - View installed fonts to review what a font is really called and use this function to update before creating the PDF file.

```
WordDocument.FontsFolder = "c:\fonts"

' Update Fonts
WordDocument.UpdateFont("Almonte Snow","Almonte Snow Regular")
```

Tables (updated from a Summary Table)

- Table Width.
    - Use "%" to set width.
- Column Widths.
    - Use "%" or "px" units for columns widths.
    - One column should be "auto".
- Use SetContentSummaryTable rather than SetContentHtmlSummaryTable.
    - Use Table, Row and Cell WordDocumentOptions to tweak formatting.