

Correspondence Management Letter Building Tutorial (M2 release)

Author: [Stefan Cameron](#), Adobe CM Dev Team

About Correspondence Management

Correspondence Management (CM) is a new Adobe LiveCycle ES Solution Accelerator built with the combination of various out-of-the-box LiveCycle products (e.g. Designer, Workbench, Guide Builder, Workspace, Forms, etc.), a Flex-based API for creating XFA content directly in the LiveCycle Repository and a Flex application called "Content Creator" for creating paragraphs (XFA fragments containing rich text) that can be re-used in various letters.

The solution provides LiveCycle ES Update 1 customers with an SDK that includes all the tools they need to quickly get started in creating paragraph content and letters that use them. Paragraph content can be hooked-up to live data in the letter using XFA's floating field technology and letters can have data connections to schemas, databases or web services. Since data is bound to fields in the letter and not in the paragraph content, paragraphs can easily be re-used in multiple letters even if those letters use different data connections and data bindings.

In the M2 release, customers will be able to make use of a new CM-specific offering called "queries". A query is defined in a letter and is executed during the Letter Filling Experience (a Form Guide designed to fill the letter with content). The result is a dynamic list of paragraphs (fragments from the Repository) based on criteria such as a tag assigned to the content using Content Creator. The major advantage over traditional fragments is that the letter does not need to be modified in order to permit the user to include new paragraphs appropriately tagged or remove old ones that are no longer applicable.

Table of Contents

| | |
|---|-----------|
| About Correspondence Management | 1 |
| Table of Contents | 2 |
| Table of Figures | 4 |
| Software Requirements | 6 |
| Definitions | 6 |
| Getting Started | 6 |
| CM.M2 Kit | 6 |
| Installing and Running the Custom Communications Sample | 10 |
| CSR AIR App | 12 |
| Workspace | 14 |
| Building a CM Solution | 16 |
| The Premise | 16 |
| The Setup | 17 |
| Custom Tags..... | 18 |
| Creating the Content..... | 18 |
| Address Block..... | 19 |
| Salutation..... | 23 |
| Introduction | 23 |
| Next Statement..... | 23 |
| Donation | 24 |
| Promo | 24 |
| Sincerely..... | 24 |
| Yours Truly | 25 |
| Footer..... | 26 |
| Spot Check | 26 |
| Building the Letter Template | 27 |
| Broken Fragment Reference Bug Workaround..... | 27 |
| Understanding the Letter Template | 29 |
| Configuring the Master Pages..... | 30 |
| <i>Inserting/Configuring the Headers</i> | 31 |
| <i>Floating Field References</i> | 32 |

| | |
|--|-----------|
| <i>Inserting/Configuring the Footer</i> | 33 |
| CM Object Library | 34 |
| Adding Body Content | 34 |
| <i>CM XFOs</i> | 35 |
| <i>Steps to follow when using CM Content</i> | 36 |
| <i>Cleaning-Up cmBodyContent and cmDataFields</i> | 37 |
| <i>Salutation</i> | 38 |
| <i>Introduction</i> | 38 |
| <i>Next Statement</i> | 38 |
| <i>Donation</i> | 39 |
| <i>Promo</i> | 40 |
| <i>Conclusion</i> | 40 |
| <i>Spot Check: Finished Adding Content</i> | 41 |
| Adding Data Fields | 43 |
| <i>Example</i> | 44 |
| <i>Preconfigured Data Fields</i> | 44 |
| <i>Root Subform Binding</i> | 45 |
| Testing/Debugging the Letter Template | 45 |
| Further Testing/Debugging | 46 |
| <i>Exposing Debug Output</i> | 50 |
| Building the Letter Filling Experience | 51 |
| Form Guide Setup | 51 |
| Configuring the Guide Object | 52 |
| Handling the Optional/Editable Promo Paragraph | 53 |
| Configuring the Conclusion Query Panel | 54 |
| Running the Letter Filling Experience | 55 |
| Troubleshooting | 60 |
| Can't login to Content Creator? | 60 |
| Paragraphs aren't spaced correctly in the letter? | 60 |
| Red border around CM Content or CM Query object instance? | 60 |
| Getting a compilation error when previewing the LFE using Guide Builder? | 61 |

Table of Figures

| | |
|---|----|
| Figure 1: CM Import Tool | 8 |
| Figure 2: Content imported into Repository..... | 9 |
| Figure 3: Content Creator login screen | 10 |
| Figure 4: New Process dialog in Workbench | 11 |
| Figure 5: ArchiveLetter process imported and opened in Workbench | 11 |
| Figure 6: Incorrect "inDataDoc" property of "theGuidedForm" process variable..... | 12 |
| Figure 7: Fixed "inDataDoc" property of "theGuidedForm" process variable..... | 12 |
| Figure 8: Running the CSR AIR app for the Custom Communications sample | 13 |
| Figure 9: Remote invocation of ArchiveLetter process using CSR AIR app..... | 14 |
| Figure 10: Task in Administrator work list | 15 |
| Figure 11: Custom Communications "Letter Filling Experience" in Workspace | 15 |
| Figure 12: New "fcFollowUp" folder with "doc" and "res" subfolders and their contents | 17 |
| Figure 13: CC initial view | 19 |
| Figure 14: Floating field named "FirstName" to be inserted | 20 |
| Figure 15: Floating fields in Address Block paragraph..... | 20 |
| Figure 16: Save As dialog for new content | 21 |
| Figure 17: Looking at "//cm/samples/fcFollowUp" via the Save As dialog | 21 |
| Figure 18: Looking inside the new "frag" folder in "//cm/samples/fcFollowUp" | 22 |
| Figure 19: New "Address_Block.xdp" paragraph fragment in the Repository | 22 |
| Figure 20: New "Address_Block" paragraph in Search Result View | 23 |
| Figure 21: Tagging the "Sincerely" paragraph as a "Conclusion" | 25 |
| Figure 22: Search results based on the "Conclusion" tag | 26 |
| Figure 23: Paragraph content in the Repository after using Content Creator | 27 |
| Figure 24: Broken fragment references in the letter template | 28 |
| Figure 25: Broken reference to "BodyContentInstructions.xdp" | 28 |
| Figure 26: Fixed reference to "BodyContentInstructions.xdp" | 29 |
| Figure 27: Sections in "MasterPageForFirstPage" | 30 |
| Figure 28: Creating the Finance Corp Logo fragment..... | 31 |
| Figure 29: New Finance Corp Logo fragment subform..... | 31 |
| Figure 30: Logo.xdp fragment dropped in cmHeaders | 32 |
| Figure 31: cmHeaders and FirstPageContentArea resized according to header content | 32 |
| Figure 32: New footer content in resized and repositioned cmFooters..... | 34 |
| Figure 33: Palette menu button..... | 34 |
| Figure 34: CM Object Library panel in the Object Library palette | 34 |
| Figure 35: Conditional-optional content example | 37 |
| Figure 36: cmBodyContent and cmDataFields cleaned-up..... | 37 |
| Figure 37: Adding a new CM Content object into cmBodyContent..... | 38 |
| Figure 38: Adding the Salutation.xdp fragment as mandatory letter content | 38 |
| Figure 39: The conditional Donation paragraph content added to the letter template | 40 |
| Figure 40: Inserting a new CM Query object in cmBodyContent | 41 |

Figure 41: State of the Hierarchy after adding all the body content 42

Figure 42: State of the Canvas after adding all the body content (showing only the letter template, not the cmControlPage) 43

Figure 43: Preconfigured data fields inserted into cmDataFields subform 44

Figure 44: The root subform renamed to "fcFollowUp" to match the root node name in the XML data.. 45

Figure 45: Setting-up the preview data 45

Figure 46: Previewing/testing the letter template as PDF..... 46

Figure 47: Setting debug form variable values to 1 47

Figure 48: cmControlPage and control fields visible in PDF view after setting debug form variables 48

Figure 49: "Promo" paragraph appears after checking the box and refreshing the letter..... 49

Figure 50: Comparing design-time view of optional-editable "Promo" paragraph..... 49

Figure 51: Override to editable "Promo" paragraph in PDF view..... 49

Figure 52: "Add Custom Library" button in Guide Builder..... 51

Figure 53: New CM wrapper in Guide Layouts panel 52

Figure 54: LFE form guide setup with new name, guide layout and submit button 53

Figure 55: Promo paragraph panel defined 54

Figure 56: Setting Conclusion panel layout to "Cm Query Controller" 54

Figure 57: cmQuerySpec added to the Conclusion query panel..... 54

Figure 58: Guide Builder Preview options 55

Figure 59: LFE running as "Guide Preview" 56

Figure 60: Inserting the "Amount" field's value into the new promo paragraph..... 56

Figure 61: New promo paragraph in the letter..... 56

Figure 62: LiveCycle Login panel in the LFE..... 57

Figure 63: Conclusion Query results in the LFE..... 58

Figure 64: "Thank you" conclusion chosen for the letter 59

Figure 65: Query result editor in the LFE 59

Figure 66: "Mr. Smith" added to the end of the first sentence in the letter's conclusion paragraph..... 60

Software Requirements

To get started with Correspondence Management (CM), you need the following (or better):

- CM “M2” Kit released on August 20, 2008
- AIR 1.0
- Flash Player 9.0.124.0
- LiveCycle Workbench ES 8.2
- LiveCycle Designer ES 8.2
- Acrobat 9.0 Pro
- Access to a LiveCycle ES Update 1 server running Forms and Workspace, along with any other LC services you may need for your own CM solutions

Definitions

A definition of terms/acronyms used in this document.

- **CC:** Content Creator web application for Correspondence Management content creation
- **CM:** Correspondence Management
- **CM.M2:** “M2” release of Correspondence Management
- **CSR:** Customer Service Representative
- **LC:** LiveCycle ES Update 1
- **LFE:** Letter Filling Experience
- **LFEU:** Letter Filling Experience User (person that uses the LFE to build a letter)
- **XFO:** XML/XFA Form Object (piece of XFA that you obtain from the Object Library palette in Designer)

Getting Started

This section will help you install the CM.M2 kit in LC so that you can start creating content and building letters and then run through the included “Custom Communications” sample to give you a taste for what you can do with the kit.

CM.M2 Kit

You will have [downloaded](#) the kit in a ZIP file named, “sa_correspondence_management_pdk.zip”. The version of the kit that you should be using should be *at least* 8.2.1.080813.488451. This was the latest build at the time of writing. There has since been a formal release to the LiveCycle ES Solution Accelerator site on **August 20, 2008**, with a **version of 8.2.1.080815.489045**. These versions contain a lot of important bug fixes as well as a brand new sample solution.

In case you’re wondering what the “M2” stands for, it means “Milestone 2”. Solution Accelerator kits will be updated much more frequently than regular products because they aren’t products: They’re development kits that utilize LiveCycle ES features. The M2 release of CM simply identifies a particular feature set available in the kit and will transcend future releases until the next milestone.

The following steps assume that you have either never installed CM before or that you have completely removed any previous installations.

1. Unzip the kit to a local folder. You will have 1 file and 3 subfolders afterwards:
 - *cm_solution_guide.pdf*: Overview of the CM kit and architecture as well as information on how to use the kit to build your own solutions.
 - *cm_building_solutions.pdf*: The HTML documentation in PDF format (probably easier to read and search).
 - *documentation*: Contains technical documentation on how to build content and letters using the kit.
 - *Human Interactive Correspondence*: Contains source files, repository files, samples and tools for CM (Import Tool, Content Creator and more). This is the main folder.
 - *On Demand Correspondence*: Contains DDG samples and documentation. Note that CM.M2 does not make use of DDG. The SDK is simply provided in the kit in case you want to make use of it in your own non-interactive (e.g. batch) CM solutions.
2. Verify that you have the correct version of the kit by opening the “//Human Interactive Correspondence/version.txt” file and verifying that the version number specified is at least **8.2.1.080813.488451** (you’ll find it on the first line in the file).
3. Import the content inside “//Human Interactive Correspondence/repository/cm” into “//cm” in your Repository. This is done using the AIR-based CM Import tool located in “//Human Interactive Correspondence/tools/cmImport/dist”. You need to use this tool rather than simply manually dragging and dropping the contents into the Repository because the tool will automatically type the content that it imports so that it’s useable by the CC app (e.g. XDPs that contain what our API identifies as a “paragraph fragment” will be typed as such so that you can edit them in CC, tags will be automatically applied to content where appropriate – usually for the samples, etc.). Install the “cmImport.air” file and then launch the tool:

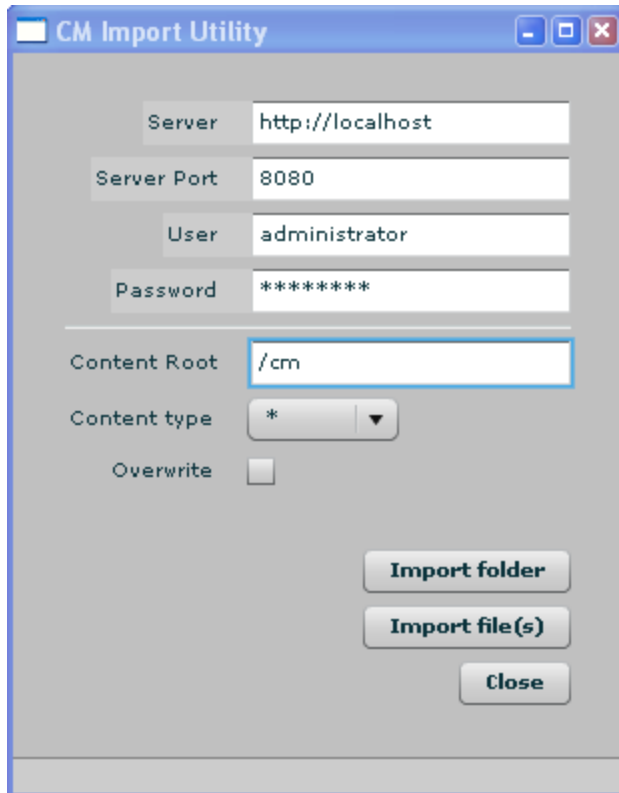


Figure 1: CM Import Tool

4. By default, the “Content Root” property is set to “/cm/content”. Change this to “/cm” (as in the picture above). Also, if your LC server is running elsewhere than “localhost”, change the “Server” property to reflect the proper URL.
5. Import the entire “//Human Interactive Correspondence/repository/cm” tree by clicking on the “Import folder” in the CM Import Tool and choosing it from the folder selector window that opens. You should see a counter appear above the “Import folder” button. The total count is approx. 88 files and takes a few minutes to complete.
6. Close the CM Import Tool and verify that the content was uploaded to your Repository using Workbench:

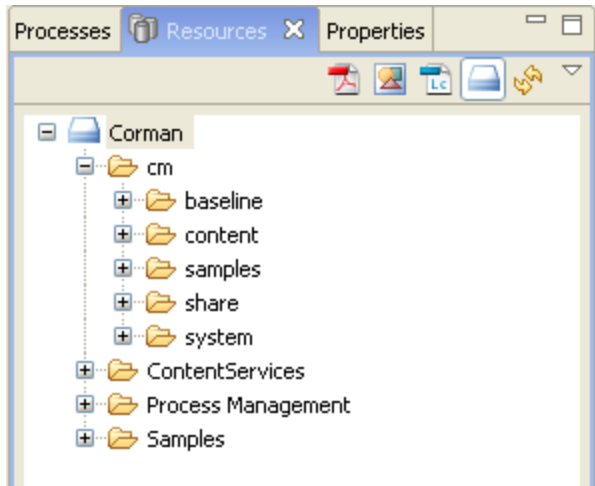


Figure 2: Content imported into Repository

Folder contents:

- *baseline*: The CM Base Template which you'll use later to create letters.
 - *content*: Empty folder where you can save paragraphs you create with CC.
 - *samples*: The CM sample solutions (3).
 - *share*: Common components required for building CM solutions and running the CM samples.
 - *system*: Configuration files used by CC.
7. Deploy CC to your LC server by dragging and dropping “//Human Interactive Correspondence/applications/ContentCreator/dist/cmContentCreator.war” into “//LiveCycle8.2/jboss/server/all/deploy”. You should now be able to access it by going to <http://localhost:8080/cmContentCreator> (where “localhost” is the name of the server on which LC is running):

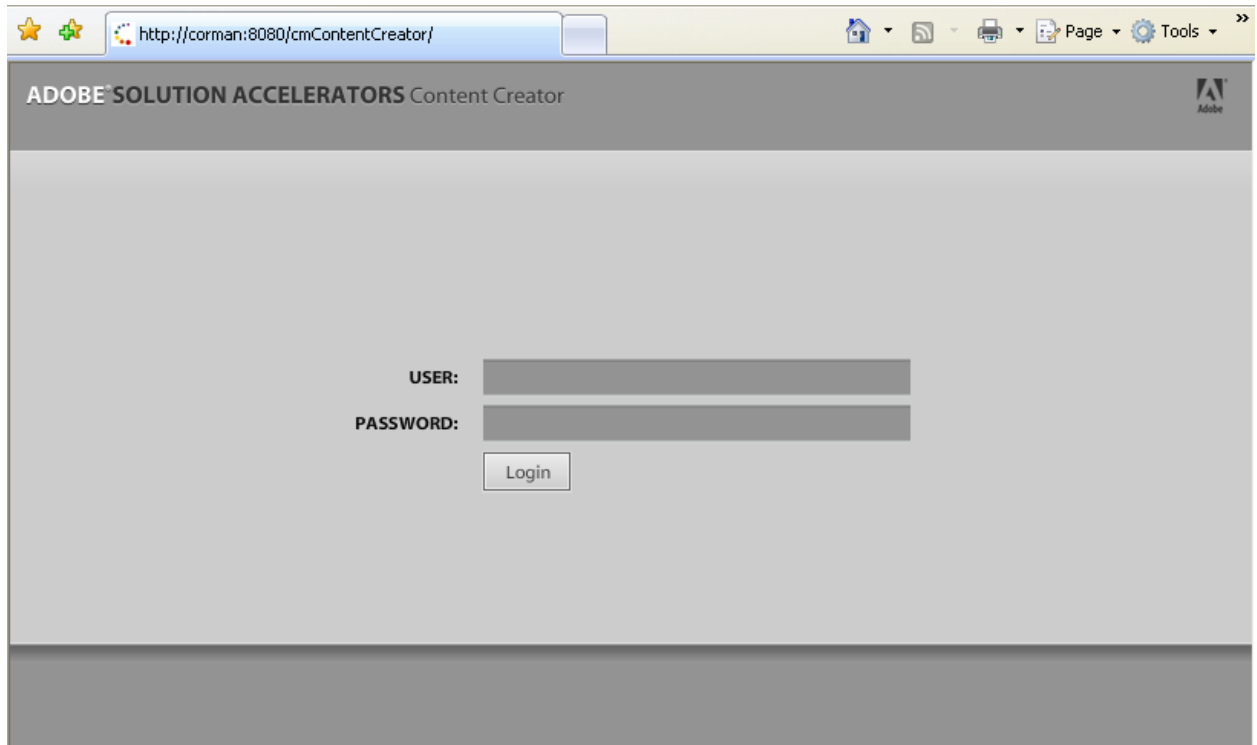


Figure 3: Content Creator login screen

At this point, you technically have everything you need to get started but you may want to install our “Custom Communications” sample which shows an end-to-end CM solution.

(Note that these installation steps are also covered in the “//cm_building_solutions.pdf” file found in your kit.)

Installing and Running the Custom Communications Sample

Pick-up where we just left-off installing the kit:

1. The sample uses a simple archival process that initiates a task in Workspace and, upon completion, generates the letter as a PDF/A file in “c:\” on the system where LC is installed. The process definition is located in “//Human Interactive Correspondence/repository/cm/share/res/ArchiveLetter.1.xml” in the kit. Switch to the Process view in Workbench and create a new process with right-clicking in the panel and choosing “New Process...” from the context menu.
2. Choose “Import a process from a file”:

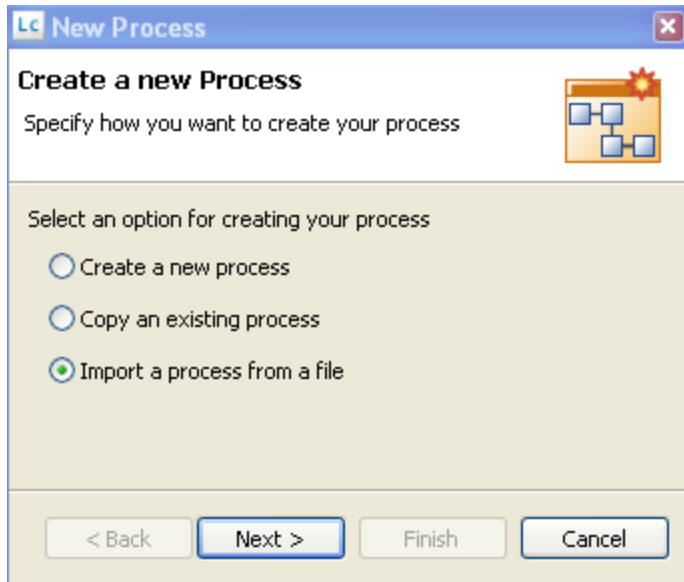


Figure 4: New Process dialog in Workbench

3. On the following screen, click on the “Browse” button and choose “//Human Interactive Correspondence/repository/cm/share/res/ArchiveLetter.1.xml” in your kit. Choose “yes” in the dialog that asks you if you want to set the name to the one specified in the process definition. Finish this step by creating a new “CM” category for the process and choosing “Finish”.
4. The “ArchiveLetter” process should now be opened in Workbench:

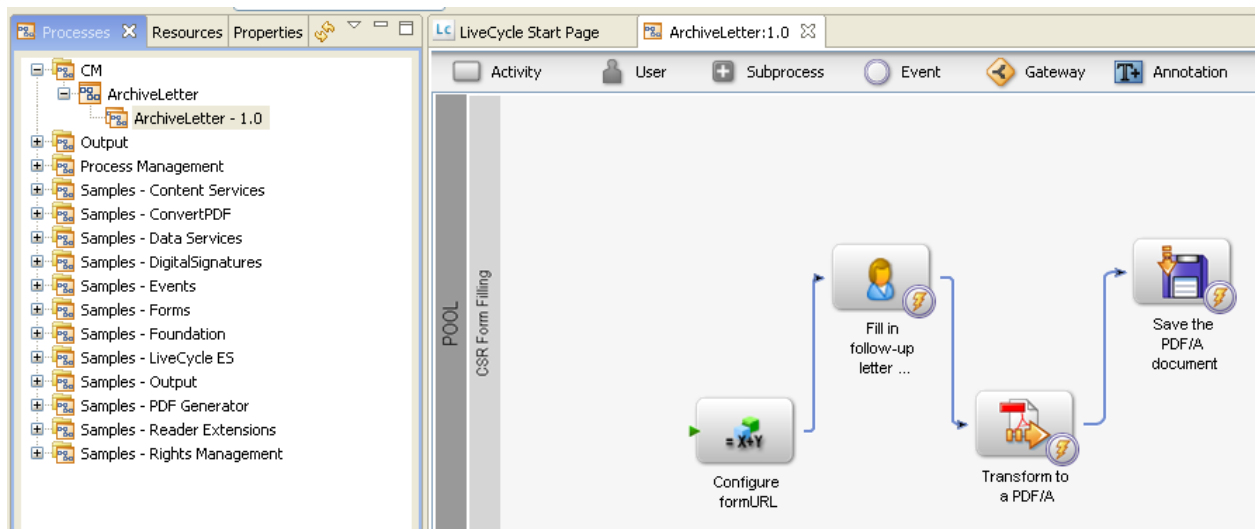


Figure 5: ArchiveLetter process imported and opened in Workbench

5. Verify that the “inXMLData” process variable is set as the “inDataDoc” property value for the “theGuidedForm” process variable. This binding sometimes gets removed when importing the process:
 - a. Edit the “theGuidedForm” process variable.
 - b. Click on the “Advanced Settings...” button.

- c. If the “inDataDoc” property looks like this (empty)

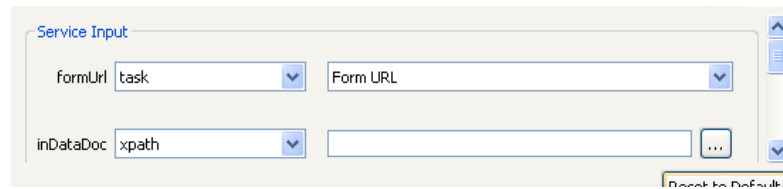


Figure 6: Incorrect "inDataDoc" property of "theGuidedForm" process variable

then **it needs to be fixed.**

- d. Set the type to “variable” and choose the “inXMLData” process variable:

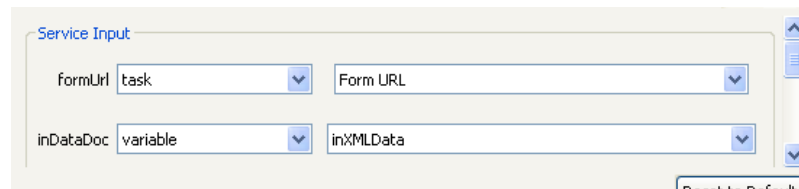


Figure 7: Fixed "inDataDoc" property of "theGuidedForm" process variable

- e. Click on OK, OK again and then **save** the process.
6. Activate the process by choosing “Activate” from the context menu on the “ActiveLetter – 1.0” item in the Process panel in Workbench.

The process has a variable named “theGuidedForm” which is set in the first step in the process, “Configure formURL”, to the URL set in the value of the <formURL> data node in the “inXMLData” process variable which will contain the XML data generated from the CSR AIR app that we will use to initiate this process (using Flex Remoting).

CSR AIR App

Now that the “ArchiveLetter” process is installed and activated, it can be invoked remotely using LC’s new Flex Remoting support. To demonstrate how this is done, the CM kit includes a CSR AIR app that simulates a data-capture application that a rep might use when talking to a client on the phone. You will use this app to initiate the “ArchiveLetter” process and go through the task of filling the “Custom Communications” letter (i.e. choosing which paragraphs to include in it). You will then complete the task which will result in the generation of the letter in PDF/A format.

1. Install the CSR AIR app located at “//Human Interactive Correspondence/applications/CustomCommunications/csrAirApp/dist/customerServiceRep.air” in your kit and run it:

Call Application

Customer Call Details

Customer Information

First Name

Last Name

Company

Street

City

State

ZIP

Conversation type

Budget

Product

Bonus

Account Executive

Policy

Server

Server Port

User

Password

Figure 8: Running the CSR AIR app for the Custom Communications sample

- Fill-in some data, **set the appropriate “Server” information**, and click on the “Init call” button at the bottom to invoke the process. You should see a dialog indicating that the process was successfully initiated:



Figure 9: Remote invocation of ArchiveLetter process using CSR AIR app

3. **Keep the CSR AIR app open during the next step so that you can compare with what you entered.**

Note that you have full access to the source for the CSR AIR app in the kit so you can check it out to see how it's done. You should also be able to modify the <formURL> in the generated data (that gets sent to the process during invocation) in order to invoke a different form with the "ArchiveLetter" process.

Workspace

To continue with the sample, load Workspace and login as Administrator by navigating to <http://localhost:8080/workspace> in your browser (where "localhost" is the name of your LC server).

1. You should have 1 task in your work list when you login:

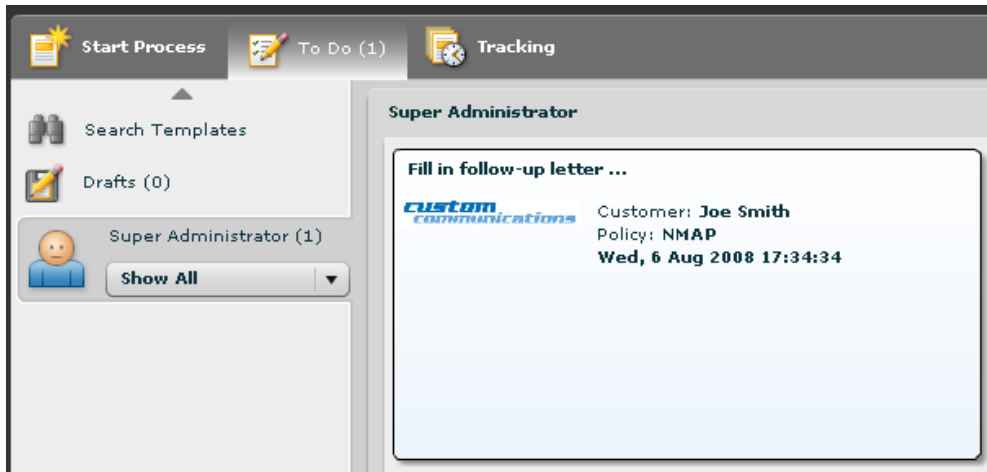


Figure 10: Task in Administrator work list

2. Click on the card to initiate the task. Depending on the performance of your server, it may take up to a few minutes before the “Letter Filling Experience” (the form guide) appears. You may want to maximize the view to see it better:

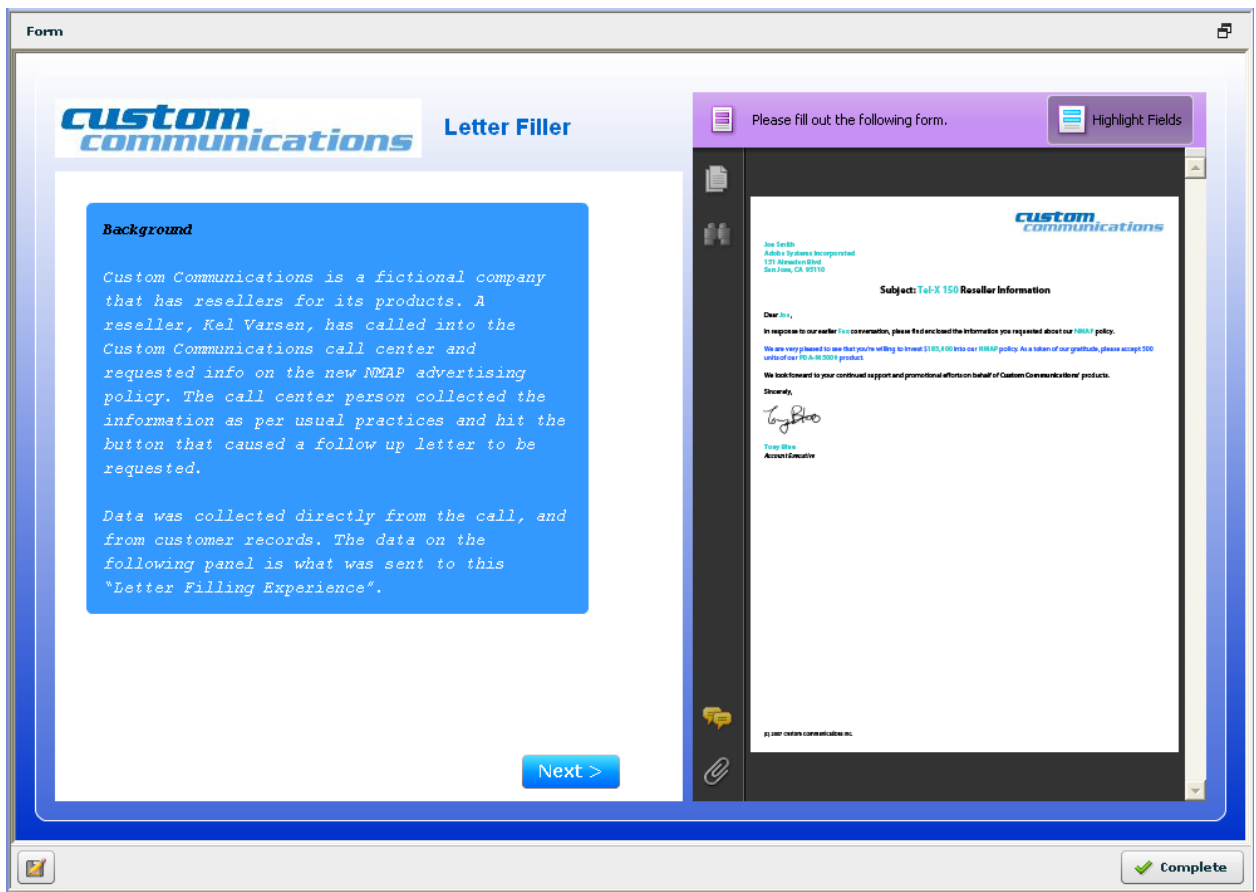


Figure 11: Custom Communications "Letter Filling Experience" in Workspace

The guide has a side-by-side view: On the left is the “Letter Filling Experience” where you choose the content to include in the letter. On the right, you see the letter being built. In this sample, we’ve highlighted all the data in the letter in light blue. If you compare this with what you entered in the CSR AIR app in the previous step, you should recognize the values. If you entered a budget over \$100,000, you’ll see the “conditional” bonus paragraph show-up in light blue in the middle of the letter.

3. Go through the “Letter Filling Experience” by clicking on “Next” until you get to the final step, choosing content as you see fit along the way.
4. When you get to the second last panel which has the latin “Lorem Ipsum” paragraphs, experiment with adding and removing various numbers of these paragraphs and observe what happens to the headers and footers in the letter when you include enough content to cause a second page to be created. This highlights some of the header and footer features of the CM Base Template which we’ll discuss in greater detail later on.
5. The last step in the form guide shows you the data as it will be submitted back to the “ArchiveLetter” process. You can see that data has been collected, in addition to what was submitted from the CSR AIR app, which identifies which paragraphs were chosen to be included in the letter.
6. Click on the “Complete” button at the bottom right hand side.

The task now completed, the process will finish by creating a PDF/A PDF file in your server’s “c:\” folder. This will be an exact replica of what you were seeing in the right hand panel in the “Letter Filling Experience” – WYSIWYG...

Building a CM Solution

We finally get down to business. The goal is to create a simple CM solution from scratch. Now that you’ve seen some of the capabilities of the CM kit, it’s time to show-off the most exciting new capabilities in CM.M2: Content Creator and queries. We’ll tackle these as we build the new solution.

The Premise

We want to build a follow-up letter from Finance Corp. that a rep would send to a customer in response to their initial investment into a new financial product (e.g. mutual fund) they’ve recently signed-up for.

- If the customer initially invests \$250,000 or more, Finance Corp will automatically donate 10% of the initial investment value in the customer’s name to a particular charity (as decided by Finance Corp).
- There is an optional promotional section that can be included in the follow-up letter. This section should also be editable so that the “Letter Filler” (person filling the letter via the “Letter Filling Experience”, that is, the form guide) can change specifics of the promo as they see fit.
- Finance Corp. has various types of conclusion paragraphs. Conclusions may be added or removed at any time so they shouldn’t be directly referenced by the letter. The Letter Filler should have the choice to include one amongst the various available conclusions. The letter

template itself should not need to be modified at a later date to account for new or removed Conclusions. Finally, the Letter Filler should have the ability to further customize the Conclusion they choose. (**Hint:** We will use a “query” to meet this requirement.)

The Setup

To keep things simple, we will use a sample XML file to simulate the data collection app (rather than a fancy AIR app like in the “Custom Communications” sample) and we’ll use Guide Builder’s Preview feature to simulate running the Letter Filling Experience in Workspace. Hence, we will not be building a process. If anything, a simple process would be very similar to the “ArchiveLetter” process so you can refer to it to see how it’s done.

1. Before we get started with creating the content, create a new folder named, “fcFollowUp”, under “//cm/samples” in your Repository.
2. Inside that folder, create two subfolders: “doc” and “res”.
3. In the “doc” subfolder, add the “testdata.xml” file attached to this PDF.
4. In the “res” subfolder, add the “finance-logo.png” file attached to this PDF.

Your Repository should now look like this:

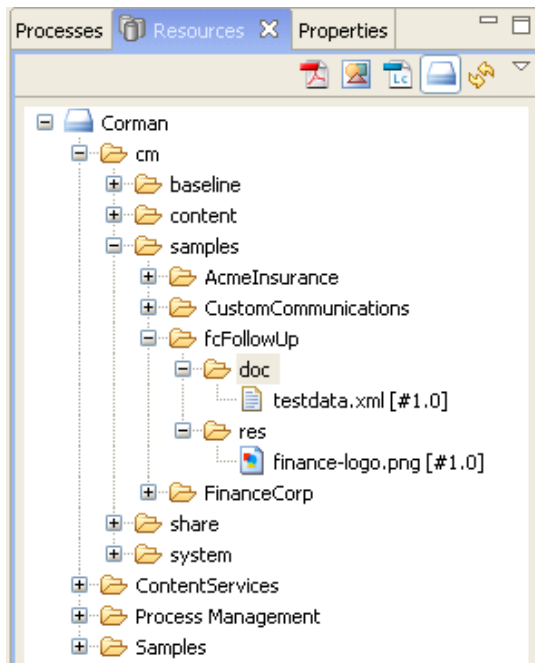


Figure 12: New "fcFollowUp" folder with "doc" and "res" subfolders and their contents

(The use of “doc” for XML data, “res” for resources such as images and, eventually, “frag” for fragments related to this solution, is based on the suggested hierarchy structure for CM as well as in the “Best practices for team-based development with Adobe LiveCycle Workbench ES” document located at http://www.adobe.com/devnet/livecycle/pdfs/lc_workbench_wp_ue2.pdf.)

Custom Tags

In order to use a query for selecting the Conclusion paragraph, we will have to tag certain paragraphs that we create as “conclusions”. We will do this in CC however CC needs to be configured to have the new “conclusion” tag so that we can choose it from the tag list. To do this, we’ll add a new tag category named “Finance Corp” and a new tag inside of it named “Conclusion”:

1. Using Workbench, edit the “//cm/system/cm_default_tags.xml” file.
2. Add the following category and tag definition just before the closing “</cm>” (note that the category and tag have unique IDs which should also be unique amongst all predefined tags):

```
<category id="fc" text="Finance Corp">  
    <tag id="fc-conclusion" text="Conclusion"/>  
</category>
```

3. Save and close the file.

Creating the Content

In CM.M2, paragraph content included in letters is created using **Content Creator (CC)**. This is a Flex application that uses our CM API (which you can find in “//Human Interactive Correspondence/libraries/cmModel” in the CM kit) to create new and edit existing paragraph content stored in the Repository. A “paragraph” is simply an XFA Fragment with a single rich text (draw) object inside of it. CC uses a special component, developed by the CM Team, to edit the rich text in a Flex environment. This component, the “XFA Rich Text Editor”, translates XFA rich text (XHTML) into Flex rich text and back.

You’ll recall that we deployed the CC app in the installation steps earlier. Access CC by navigating to this URL: <http://localhost:8080/cmContentCreator> (where “localhost” is the name of your LC server) and login as Administrator. You should see the following view:

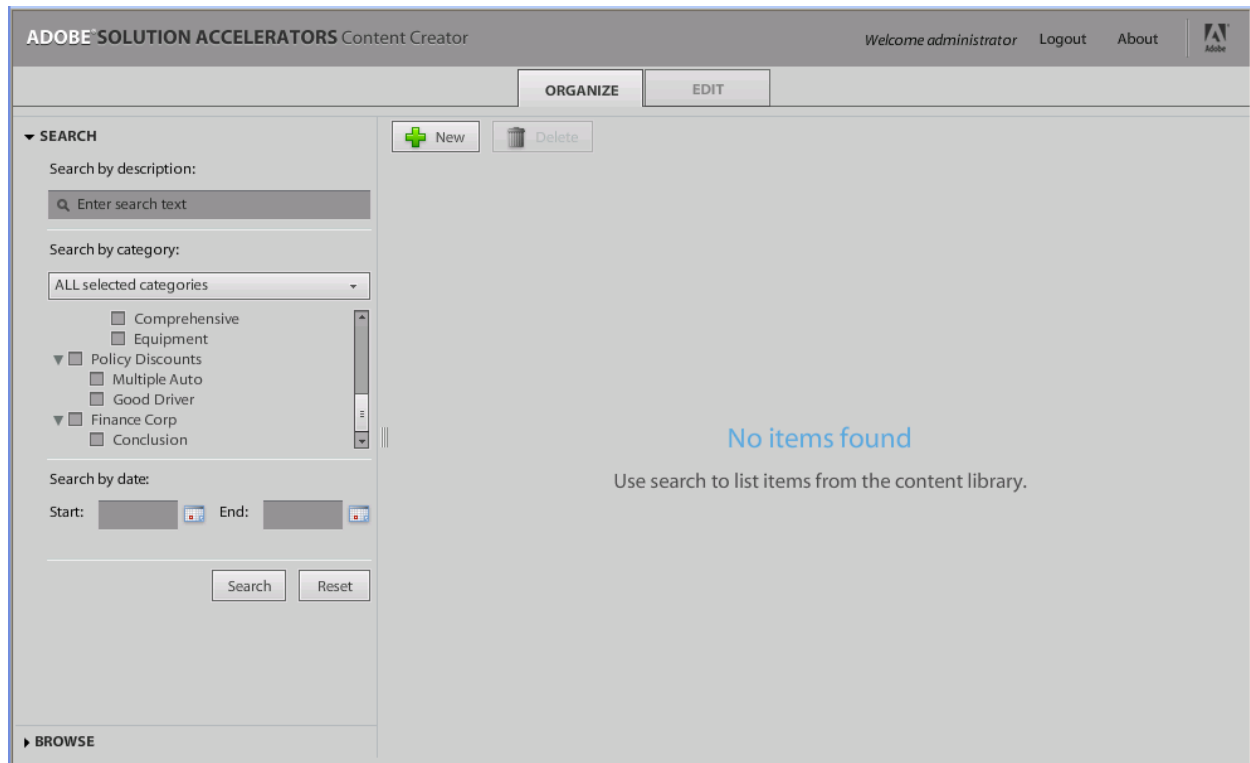


Figure 13: CC initial view

Make sure that you have the “Finance Corp” category and “Conclusion” child tag (as seen above) by scrolling the category list to the bottom.

What we need to do now is create various paragraphs for the follow-up letter that will make use of the data that we’ll obtain from the fake data capture app. In order to know what data we’re dealing with, have a look at the “//fcFollowUp/doc/testdata.xml” file in Workbench:

```
<fcFollowUp>
  <salutation>Mr.</salutation>
  <firstName>Joe</firstName>
  <lastName>Smith</lastName>
  <address1>151 Almaden Blvd</address1>
  <address2>San Jose, CA 95110</address2>
  <product>Mutual Fund</product>
  <initialInv>500000</initialInv>
  <statements>monthly</statements>
</fcFollowUp>
```

You can see that we’ll have salutation, first name, last name, two address lines, product, initial investment and statement information. The next steps will get you to create various paragraphs that make use of it (where “__FieldName__” is to be replaced by a floating field with the name “FieldName”):

Address Block

Content:

__FirstName__ __LastName__
 __AddressLine1__
 __AddressLine2__

1. Click on the “New” button. This will open the Editor View.
2. Set the “Title” property in the “Details” panel to “Address Block”.
3. In the editor, create the first floating field, named “FirstName”, by placing the I-beam in the white box and clicking the “Insert Field” button:

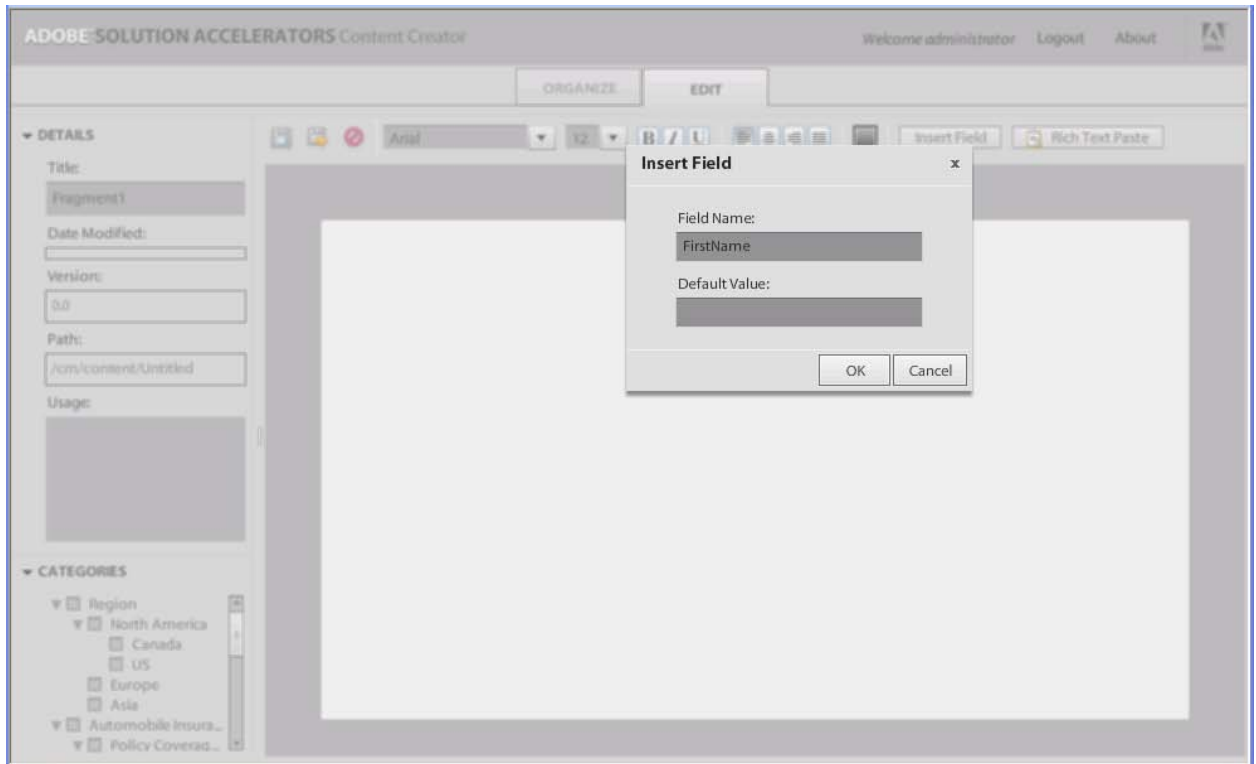


Figure 14: Floating field named "FirstName" to be inserted

4. Click on OK. Repeat this for “LastName”, “AddressLine1” and “AddressLine2”. This is creating floating field references to fields named “FirstName”, “LastName”, “AddressLine1” and “AddressLine2” which we will add to the letter template later on. When you’re done adding the floating fields, the content in the editor should look like this:

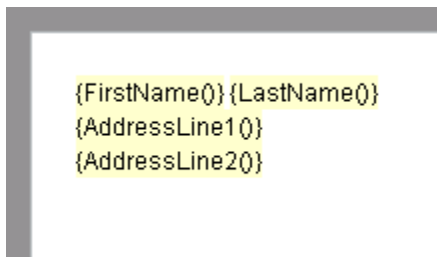


Figure 15: Floating fields in Address Block paragraph

5. Add some formatting by selecting all the contents and setting the font to “Times New Roman, 14pt, Italic”.


6. Click on the “Save” button (). Since this paragraph is new, you’ll get the “Save As” dialog:



Figure 16: Save As dialog for new content

This dialog is showing you live content right from the Repository. Notice that the file name defaults to the Title you gave to your paragraph. Using your mouse, click on the “..” folder at the top to go back to “//cm” (you can see the current path just below “Save Resource” in the dialog). Now go into “samples/fcFollowUp”:

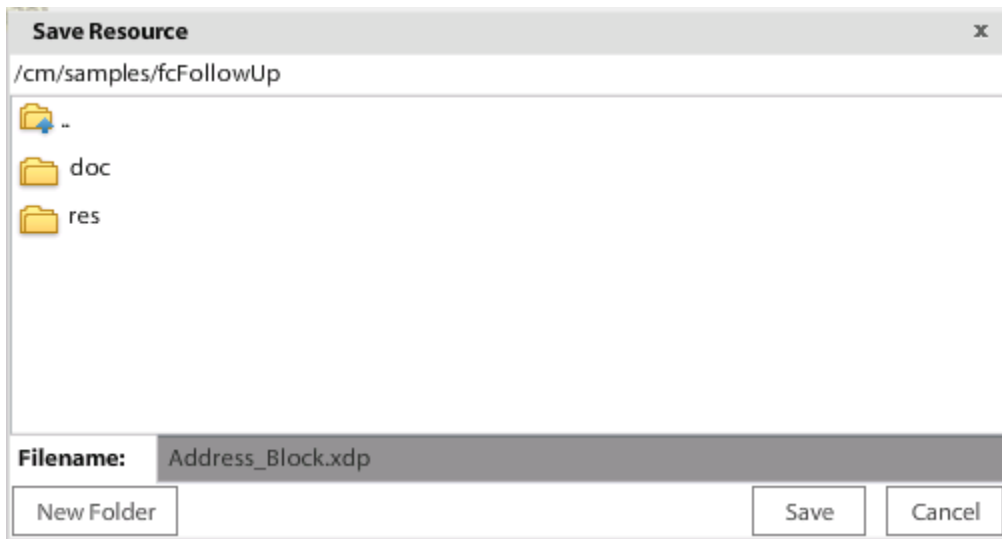


Figure 17: Looking at “//cm/samples/fcFollowUp” via the Save As dialog

7. Use the “New Folder” to create a new “frag” folder and double-click to go inside:

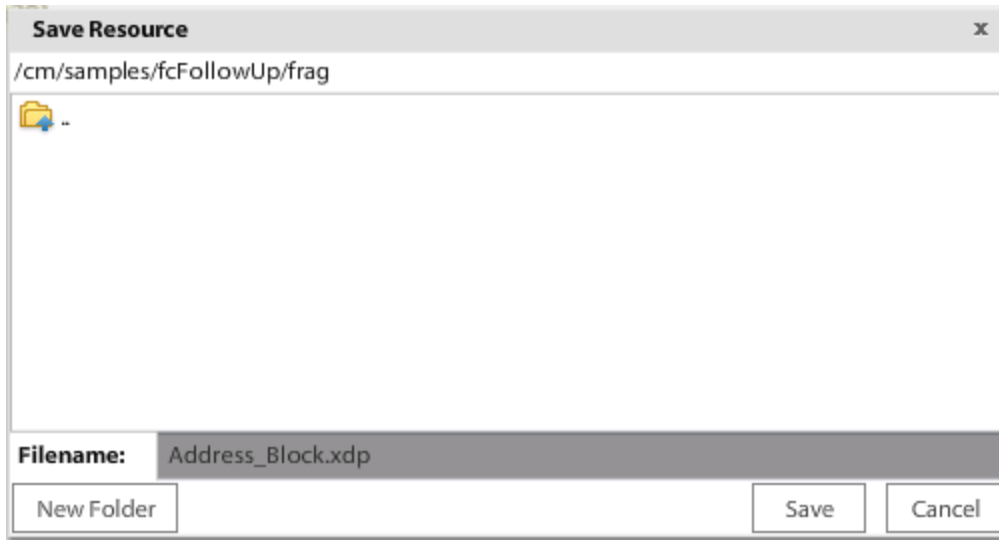


Figure 18: Looking inside the new "frag" folder in "//cm/samples/fcFollowUp"

8. Click on the "Save" button.
9. Notice how the "Path" property in the "Details" panel was updated to reflect the paragraph's new path in the Repository: "/cm/samples/fcFollowUp/frag/Address_Block.xdp". If you go back to Workbench, you should also see the new paragraph there, where you saved it:

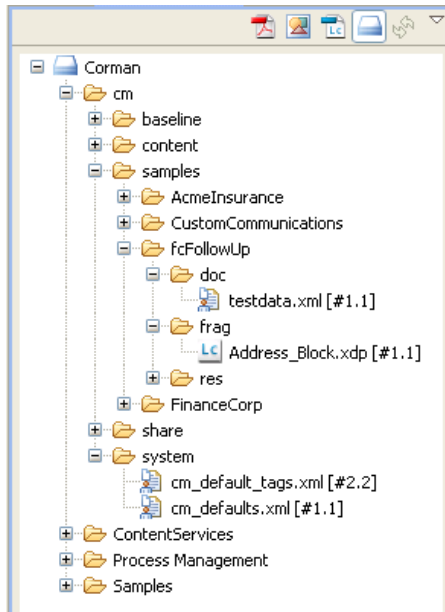


Figure 19: New "Address_Block.xdp" paragraph fragment in the Repository

10. Click on the "Organize" tab in CC to go back to the Organize View. New content is automatically displayed in the Search Result View:

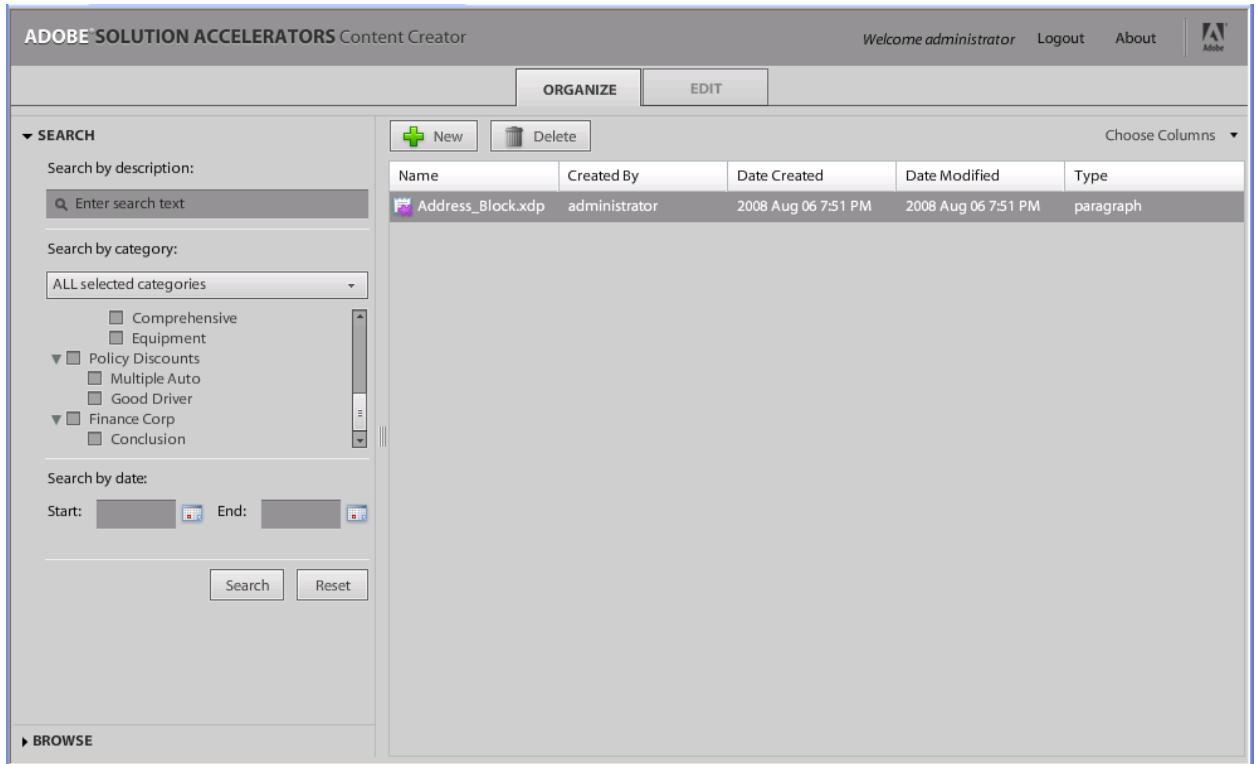


Figure 20: New "Address_Block" paragraph in Search Result View

For the following paragraphs, simply repeat the steps above (for the "Address Block" paragraph) using the specified content. Remember that "__FieldName__" represents a floating field with the name "FieldName".

Salutation

Content:

Dear __Salutation__ __LastName__,

Introduction

Content:

We are writing to confirm receipt of your initial investment of \$__Amount__ into your new Finance Corp __Product__ Portfolio.

Next Statement

Content:

In your Account Setup Interview, you indicated a preference for __StatementFreq__. As such, your next statement will be issued on __NextStatement__.

You may notice that there is no data in the XML data file (testdata.xml) for the "next statement" information. That's because it'll be a calculated value within the form itself. This not only demonstrates that floating field names don't have to match a data node name (as you've seen already with the

floating field named “Amount” which will hold the initial investment amount found in the <initialInv> data node), it shows that they don’t have to refer to a field that represents a data value either.

Donation

Content:

We are pleased to inform you that your initial investment into our __Product__ product qualifies for our current *Matching Investment Donation* program to help under-privileged children in Third World countries. A donation of \$__DonatedAmount__ will be made in the name of “__FirstName__ __LastName__” and you will receive a tax receipt at the end of the year.

As in the previous paragraph, there’s no data for the “donated amount”. This will also be a calculated value (10% of the <initialInv> data value).

Promo

Content:

We would also like to take this opportunity to tell you about our new ***FinanceCorp Investor's Counsel membership*** which you are eligible to receive for the low fee of only **\$69.99 per month** (that's **50% off the regular price**). The Investor's Counsel is a group of like-minded investors that meet every two weeks to discuss various investment strategies. Every meeting begins with a presentation by a special guest and is attended by one of our Investment Specialists so that all your questions can be answered.

Sincerely

The **Conclusion paragraphs** require an extra step **prior to saving**: Choose the “Conclusion” tag in the “Categories” panel (at the bottom left hand side):

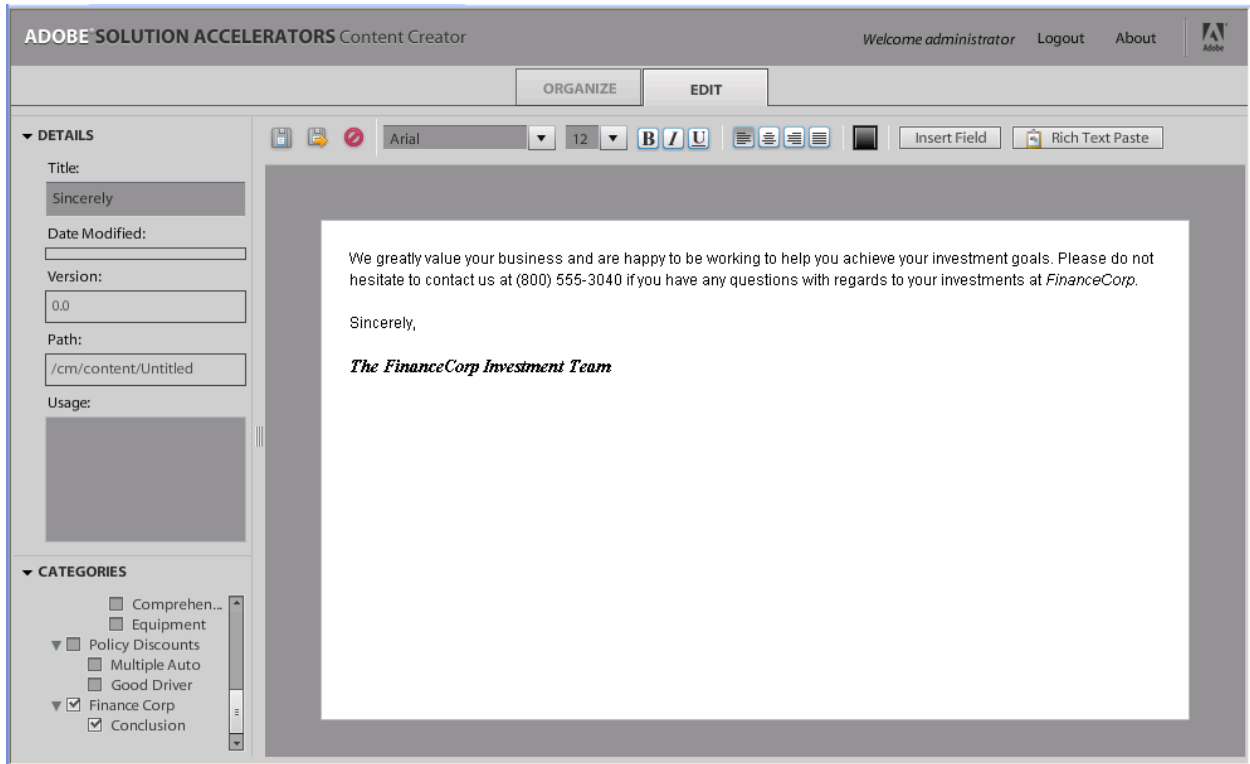


Figure 21: Tagging the "Sincerely" paragraph as a "Conclusion"

Content:

We greatly value your business and are happy to be working to help you achieve your investment goals. Please do not hesitate to contact us at (800) 555-3040 if you have any questions with regards to your investments at *FinanceCorp*.

Sincerely,

The FinanceCorp Investment Team

Yours Truly

This is the second **Conclusion** paragraph. Don't forget to **tag** it as a "Conclusion" paragraph.

Content:

Thank you for choosing *FinanceCorp* to help you achieve your investment goals. Please do not hesitate to contact us at (800) 555-3040 if you have any questions with regards to your investments at *FinanceCorp*.

Yours truly,

The FinanceCorp Investment Team

Footer

The letter footer paragraph (set the font to “Arial, 10pt” and right-align the paragraph).

Content:

(c) Finance Corp

Spot Check

At this point, we’ve created all the content. If you followed these instructions correctly, you should be able to search the Repository for content tagged as a “Conclusion” and get two results: “Sincerely.xdp” and “Yours_Truly.xdp”.

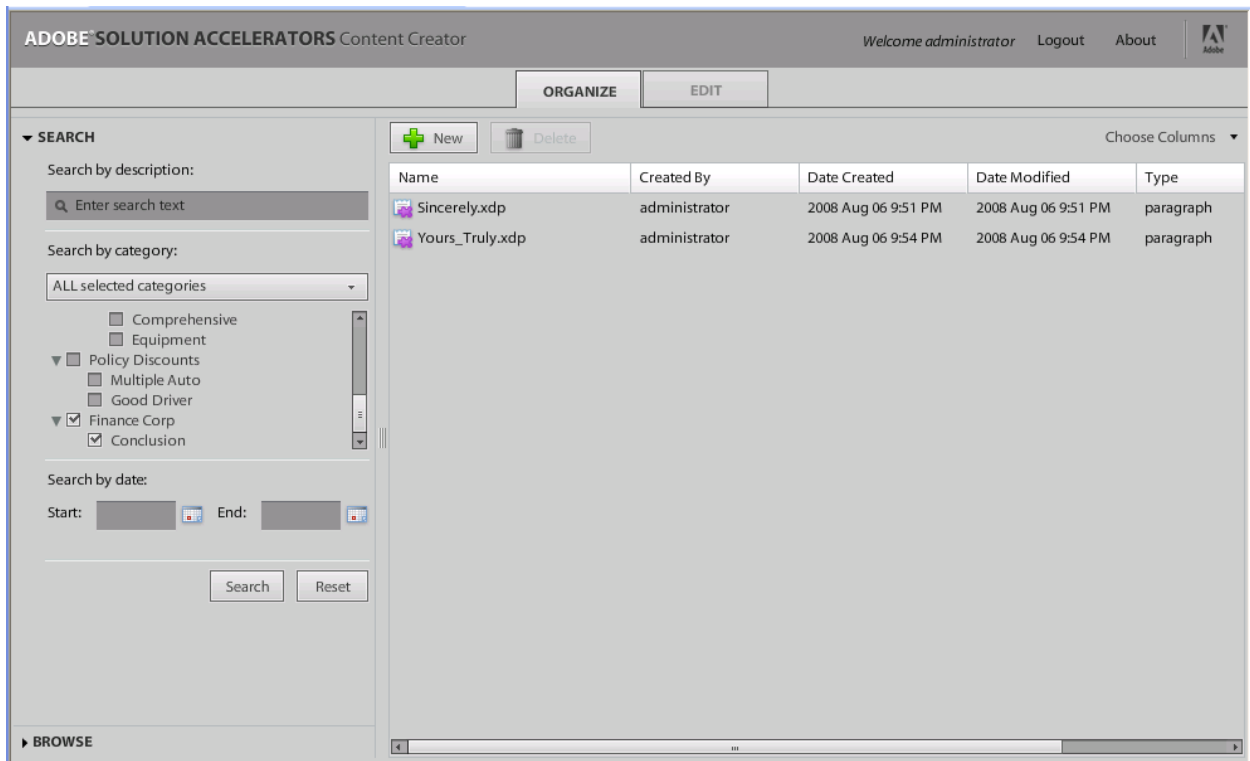


Figure 22: Search results based on the "Conclusion" tag

Finally, the “//cm/samples/fcFollowUp/frag” folder in your Repository should look like this (don’t worry about the version numbers):

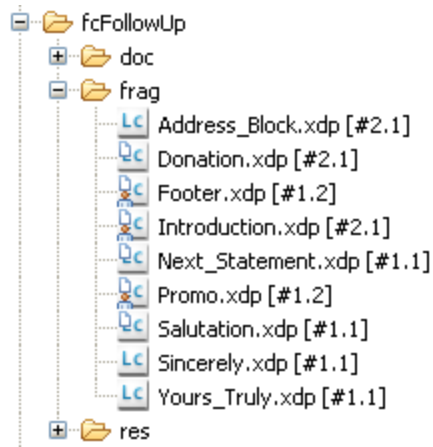


Figure 23: Paragraph content in the Repository after using Content Creator

Building the Letter Template

The next step is building the letter template – that is, the XDP form that will reference all those fragments and qualify them according to their inclusion rules (e.g. are they mandatory, optional, conditional, editable or a combination of these?).

1. To get started building a letter, open the “//cm/baseline/cmBaseTemplate.xdp” file in your Repository using Workbench. This will open the “CM Base Letter Template” in Designer. **Do not make modifications to this form!** Since Workbench doesn’t support creating new XDP documents based on real Designer Template Files (.TDS) (something you can do with standalone Designer), you must open the cmBaseTemplate.xdp file and **save it as** your new letter template to get started.
2. With the cmBaseTemplate.xdp file open in Workbench, save it as “//cm/samples/fcFollowUp/fcFollowUp.xdp”

Broken Fragment Reference Bug Workaround

At this point, you should have a new letter template named “fcFollowUp.xdp” open for edit in Workbench. Before we really get started on things, we need to **work around a bug** where the fragments referenced by the XDP are not properly resolved (or they aren’t brought down to the local file system) upon opening it. The cmBaseTemplate.xdp letter template contains some instructions and sample content which are fragment references. It also contains a reference to the “cmUtils.xdp” script object fragment which is essential to all CM Letter Templates. These references must be restored before we go any further:

1. In the Hierarchy palette, you will see 4 broken fragment references (look for the red broken puzzle pieces):

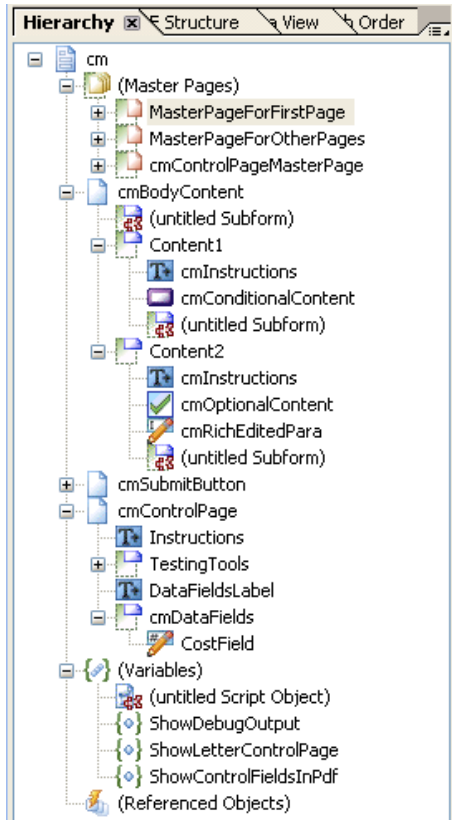


Figure 24: Broken fragment references in the letter template

2. Select the first one (located just under the “cmBodyContent” subform and choose the “//cm/baseline/BodyContentInstructions.xdp” fragment using the “Object palette > Subform tab”. The fragment will appear to already have been set however choosing it again will force Workbench to sync-up with the fragment’s content and properly establish the referencing relationship, thereby fixing the broken reference.

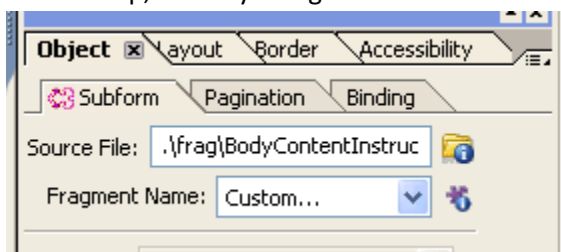


Figure 25: Broken reference to "BodyContentInstructions.xdp"

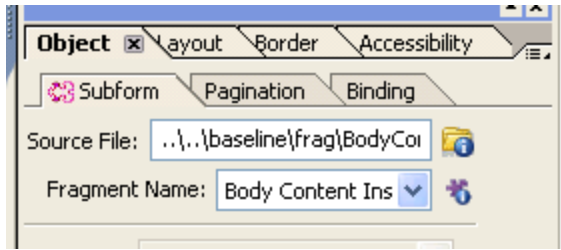


Figure 26: Fixed reference to "BodyContentInstructions.xdp"

(You may notice that the path is no longer correct. This is also related to the same bug. Since the reference is broken when you first open the cmBaseTemplate.xdp, Workbench doesn't update it when you save it as "//cm/samples/fcFollowUp/fcFollowUp.xdp".)

3. Select the second one, located just under the "cmConditionalContent" button, and set it to "//cm/baseline/frag/ConditionalParagraph.xdp".
4. Select the third one, located just under the "cmRichEditedPara" text field, and set it to "//cm/baseline/frag/OptEdtParagraph.xdp".
5. Select the fourth and final one, located just under the "(Variables)" node, and set it to "//cm/share/frag/cmUtils.xdp". **This is the most important one of them all.** It's the CM Script Object that makes the letters work!

Understanding the Letter Template

The letter template has a simple structure:

- Header and footer content goes in the master pages. There are master pages for the first (or only) page ("MasterPageForFirstPage") and for the rest of the pages ("MasterPageForOtherPages").
- The "cmControlPageMasterPage" should be left alone since it's meant to be used with the "cmControlPage" and won't affect your letter layout.
- Letter content goes in the "cmBodyContent" subform. This subform should have a set width which is the maximum width for content in the letter. You'll notice that paragraphs were created with a set width of 7 inches. By placing them inside the "cmBodyContent" subform, their widths will automatically be extended to the width of the "cmBodyContent" subform, thereby making them look like they were actually 8.5 inches wide rather than 7 inches wide.
- Since the results of the LFE (the form guide) are submitted back to LC, the letter template comes with pre-configured submit buttons in the "cmSubmitButton" subform. This subform is always hidden in the PDF view but not in the guide (LFE) view. Inside, you'll find a "cmGuideSubmitButton" which is a special button that will let you submit the LFE data from the LFE (form guide) itself rather than requiring the LFEU to click on a submit button in the PDF view. More on this later in the section on Building the LFE.
- The "cmControlPage" subform, always hidden in the PDF view, contains some tools for testing letter templates (mainly, a button to refresh the form with new data and an email submit button so that you can take a peek at what the data submitted back to LC will look like).

- The “cmDataFields” subform, contained within the “cmControlPage”, is where you must put all of the fields that are referenced as floating fields from paragraph content. When you created the paragraphs in the earlier steps and you inserted floating fields, those floating fields are references to fields with the same name located inside the “cmDataFields” subform. More on this later.
- The “cmUtils” script object fragment, located in “(Variables)”, is the brains of the CM letter template solution. It contains scripts that are referenced by the various objects in the CM Object Library.
- The “ShowDebugOutput” form variable, set using the “File > Form Properties > Variables tab”, when set to 1, will cause debug output to be displayed in Acrobat’s JavaScript Console. Very useful when previewing the PDF and getting errors.
- The “ShowLetterControlPage” form variable, when set to 1, will cause the “cmControlPage” subform to remain visible in the PDF view so that you can use the testing tools and manually set the values of the floating fields.
- The “ShowControlFieldsInPdf” form variable, when set to 1, will cause the control fields (which come from the CM Object Library – more on this later), to remain visible in the PDF view so that you can manually action them to test various scenarios (rather than always having to run the LFE form guide to do it).

For more information on the letter template, see the **CM Technical Documentation** included in the **cm_building_solutions.pdf** file in your CM.M2 kit.

Configuring the Master Pages

The letter template provides two master pages, one for the first page and another for all remaining pages, including the last page. We know our letter is short so we don’t need to bother with all the various combinations of headers and footers on the first, last and other pages. All we need is a single page and the “MasterPageForFirstPage” will suit us just fine.

1. Remove the “MasterPageForOtherPages” since we don’t need it. Just delete it from the Hierarchy palette.
2. In the remaining “MasterPageForFirstPage”, there are 3 sections: “cmHeaders”, “FirstPageContentArea” and “cmFooters”:

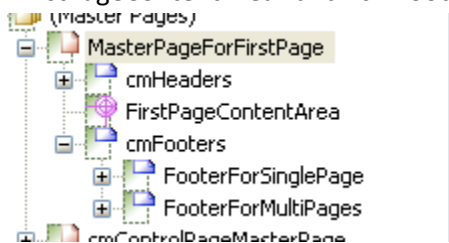


Figure 27: Sections in “MasterPageForFirstPage”

“cmFooters” contains two subforms, “FooterForSinglePage” and “FooterForMultiPages”. The idea is that you put the footer you want to have if the letter has a single page in the “FooterForSinglePage” subform and the one that should be used if the letter has multiple pages

(so the first page isn't the last page) in the "FooterForMultiPages" subform. Note that unlike the master pages, you can't simply remove the "FooterForMultiPages" subform even if you know you don't need it. If you do, you'll get an error in the cmUtils script.

Inserting/Configuring the Headers

1. In cmHeaders, replace the "FirstPageHeader" place-holder text object with the Finance Corp logo:
 - a. Create a new XDP form in Workbench in the "//cm/samples/fcFollowUp/frag" folder named "Logo.xdp". This will be a fragment for the logo.
 - b. Drag the "//cm/samples/fcFollowUp/res/finance-logo.png" image into the new logo.xdp form.
 - c. Size it the way you want (as you sit for a header). You can use "Scale Image Proportionally" for the "Object palette > Image tab > Sizing property" and set the width to 2.25in and height to 1in if you like.
 - d. Right-click on the image object and choose "Fragment > Create Fragment..." from the context menu.
 - e. Give it a title, "Finance Corp Logo", and set it to be created in the current file:

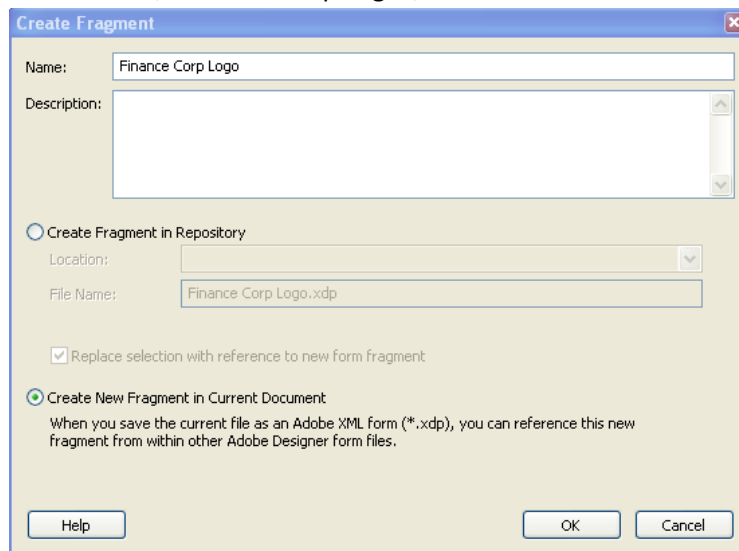


Figure 28: Creating the Finance Corp Logo fragment

- f. You should now have a new fragment subform named "Finance_Corp_Logo" that contains the logo image:

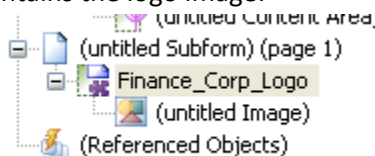


Figure 29: New Finance Corp Logo fragment subform

- g. Save and close the Logo.xdp form.

- h. Back in fcFollowUp.xdp (the letter template we’re building), switch to the Master Page view and delete the FirstPageHeader place-holder inside cmHeaders.
- i. Drag & drop the Logo.xdp fragment into cmHeaders:

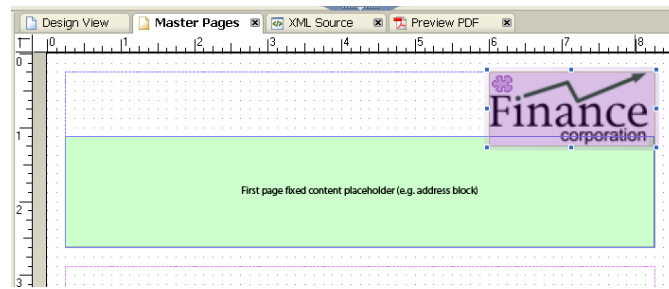


Figure 30: Logo.xdp fragment dropped in cmHeaders

- 2. Still in cmHeaders, replace the FirstPageFixedContent with a reference to the “Address Block” fragment by first deleting the FirstPageFixedContent place-holder and then dragging-in the Address_Block.xdp fragment. When you place the fragment reference into cmHeaders, make sure it’s positioned at x=0. This is because the **fragment will be resized**, at runtime, **to the width of the cmHeaders subform**. (As I mentioned before, paragraphs created with CC are set to 7in in width and depend on cmUtils scripts to be resized at runtime to fit nicely in their containers. 7in is an arbitrary width that we thought would generally be narrower than the average page width.)
- 3. Resize cmHeaders to fit nicely and resize FirstPageContentArea to regain some of the extra space:

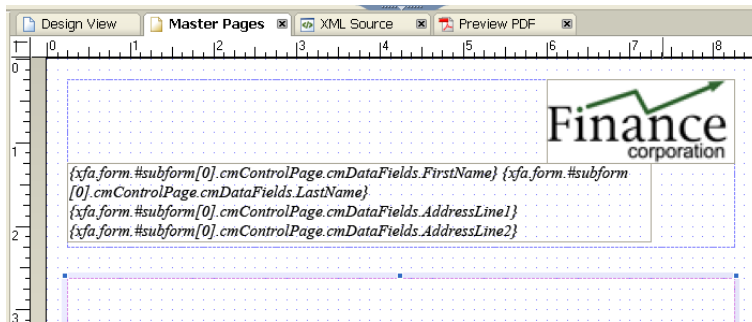


Figure 31: cmHeaders and FirstPageContentArea resized according to header content

Floating Field References

You’ll notice, at this point, that the content is showing some SOM expressions:

- `xfa.form.#subform[0].cmControlPage.cmDataFields.FirstName`
- `xfa.form.#subform[0].cmControlPage.cmDataFields.LastName`
- `xfa.form.#subform[0].cmControlPage.cmDataFields.AddressLine1`
- `xfa.form.#subform[0].cmControlPage.cmDataFields.AddressLine2`

These are the floating fields that you inserted into the paragraph using CC earlier. Floating fields aren’t typically inserted as SOM references to other fields. The only way Designer 8.2 works with floating fields

is by inserting a completely new sibling text field and referencing it by a unique ID. The disadvantage of this is that you would end-up with potentially hundreds of fields amongst all paragraph content that may all refer to the same data (form bloat) and you would also be forced to have a hard-set binding on those fields to get data into them – but what if your schema changes from letter to letter? By disassociating the floating fields from the content that references them, there is always only a single field for all paragraphs that want to display a common data value and each letter can have its own schema because the field bindings are specific to each letter – not each paragraph.

We'll get to what we do with these as-yet unresolved references later when we cover the cmDataFields subform in cmControlPage.

Inserting/Configuring the Footer

cmFooters is a little complicated (if not frustrating!) to work with because the FooterForSinglePage and FooterForMultiPages subforms are positioned one on top of another. This is because it's not possible to have them flowed, especially at the bottom of the page. If cmFooters were flowed, its contents would flow past the end of the page and you wouldn't find it any easier to work with (never mind the fact it really wouldn't look nice). Our work is a little easier, however, since we only need a "single page" footer.

1. Delete the MultiPagesFooterContent place-holder but **leave the FooterForMultiPages** subform.
2. To help with designing the single page footer, set the presence of the FooterForMultiPages subform to "hidden". This will ensure that the footer objects we create are placed in the FooterForSinglePage subform rather than the other one.
3. Delete the SinglePageFooterContent place-holder.
4. Drag-in the Footer.xdp paragraph (fragment) and place it at the top/left corner of the FooterForSinglePage subform (so that it's easier to resize the subform later). Note that any paragraph content in cmFooters will automatically be resized to fit the width of the footer subform it's place in (FooterForSinglePage in this case).
5. Resize the FooterForSinglePage subform to fit nicely around the footer content.
6. Make the FooterForMultiPages subform visible again and resize it the same way.
7. Resize cmFooters to fit nicely around its content.
8. Lower cmFooters and make FirstPageContentArea taller so that there isn't as big of a gap between the body content and the footer. **To move cmFooters** (this is difficult since Designer will think you're dragging the FooterForMultiPages subform since it's higher in z-order):
 - a. Select cmFooters in the Hierarchy palette.
 - b. Click on the **scroll bar** on the Canvas (not on the form, on the scroll bar). This sets focus to the Canvas without changing the selection.
 - c. Using the down arrow key, move cmFooters (and all its children) down about 0.5in.



Figure 32: New footer content in resized and repositioned cmFooters

CM Object Library

Before we get started with adding content to the letter (in cmBodyContent), we need to setup the Object Library palette with a shared library containing CM-specific objects that will let us qualify content (e.g. paragraphs and other types of fragments) as “optional”, “editable”, “conditional” or a combination thereof. This library also contains the tools needed to insert a query into a letter.

1. Choose “Shared Library Location...” from the Object Library palette menu located at the top right hand side of the palette:



Figure 33: Palette menu button

2. Choose the “//Human Interactive Correspondence/objectLibrary/cmObjectLibrary.xml” file from your kit (on your local file system – not in the Repository).

You should now have a “CM Object Library” panel in the Object Library palette:

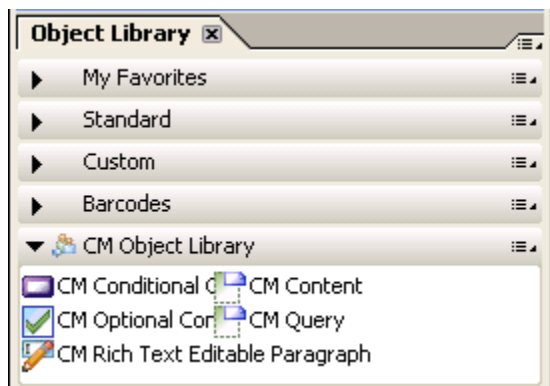


Figure 34: CM Object Library panel in the Object Library palette

Adding Body Content

We’re now at the stage where we’re ready to add the content that will flow into the body of the letter. This content, called the “body content” will be placed in cmBodyContent and will flow from page to page (our sample is a single page only so we won’t see this effect however you can see it in the Custom Communications sample which we reviewed earlier). As with cmHeaders and cmFooters, paragraph content (that is, a fragment reference containing a single text object) placed in cmBodyContent will automatically be resized to fit the width of cmBodyContent which is assumed to be the width of the letter (page).

CM XFOs

In the previous step, we added the shared CM Object Library to the Object Library palette. This library contains the tools that you will use to add and qualify body content in the letter template:

- *CM Content (subform)*: Every – yes, **every** – single piece of content (fragment paragraph or otherwise) must be added **to its own** CM Content XFO. This is a special subform that sets the scope for the “control/qualifier” XFOs you will add to it as well as identifies the piece of content on which the controls/qualifiers should act.
- *CM Optional Content (checkbox)*: Adding this object to a CM Content object will qualify its content as optional (the LFEU will have the option to include it in the letter).
- *CM Conditional Content (button)*: Adding this object to a CM Content object will qualify its content as conditional based on a comparison, which you specify, to a value (usually a merged-in data value).
- *CM Rich Text Editable Paragraph (rich text field)*: Adding this object to a CM Content object will qualify its content as being editable (the LFEU will have the option to edit the content to their liking). Note that this object only works with **rich text** paragraph content. (All paragraph content created by CC is in rich text.)
- *CM Query (complex subform)*: The **only** exception to the “CM Content rule” (where each piece of content must be wrapped in its own CM Content object) is when you want to add query result content to your letter template and you do it by using a CM Query subform. **Queries only support rich text paragraph content in their result set.** Any other type of content which satisfies the query spec will be completely ignored. The subform itself contains 3 fields and a repeatable subform:
 - *cmQuerySpec*: The query statement, set by editing the field’s Initialize event which sets the field’s default value.
 - *cmResultSelect*: The “selectability” of query results – set this by setting the drop down list’s default value using the Object palette:
 - *chooseAll*: All results are mandatory and will automatically be inserted into the letter when running the LFE.
 - *chooseAny*: Results are optional. The LFEU can choose which ones to include (note that more than one may be included).
 - *chooseOne*: Results are optional however only one may be included from the set.
 - *cmResultEdit*: The “editability” of query results – set this by setting the drop down list’s default value using the Object palette:
 - *readOnly*: Query results may not be modified by the LFEU.
 - *editable*: Query results may be modified (edited) by the LFEU.
 - *cmQueryResult (repeatable subform)*: Used to create new instances of the cmParagraph rich text field, one per chosen/mandatory query result.
 - *cmParagraph (rich text field)*: When a query result is selected/edited (whether automatically if results are “choose all” or manually by the LFEU), its XHTML content

There is one other type of content which is “mandatory” (content which is included in the letter and over which the LFEU has no control). Mandatory content is simply content which is either included in a CM Content object that doesn’t include any qualifiers or which is included as the result of a “choose all” query.

Of course, all of the visible aspects of these XFOs will automatically be hidden in the PDF view unless you set the “ShowControlFieldsInPdf” debug form variable to 1.

Steps to follow when using CM Content

Here are the steps to follow when you use the CM Content object to add content to a letter template:

1. Insert a new CM Content object in cmBodyContent.
2. Ensure the CM Content object has a unique binding. **By default, the CM Content object has no binding which will generate an error in the PDF view.** You may do this in one of two ways:
 - a. *Explicit Binding (recommended)*: Give the subform a unique, explicit binding reference using the “Object palette > Binding tab > Default Binding property”.
 - Though you don’t have to change the default name of the CM Content object, you would be **wise to do so** since it’ll be much easier to identify content once we get to building the LFE using Guide Builder.
 - Since the result will be that data required by the qualifier/control objects (optional, conditional, editable) will be saved in this data group node, I would recommend you use a common parent data group node such as “controls” when setting this binding. For example, set the first CM Content object’s binding to “controls.Para1” and the second to “controls.Para2”. The result will look like this in the XML data:


```
<root>
                <dataNode/>
                <controls>
                  <Para1>
                    ...
                  </Para1>
                  <Para2>
                    ...
                  </Para2>
                </controls>
              </root>
```
 - b. *Implicit Binding*: Give the subform a unique name using the “Object palette > Binding tab > Name property” (or rename it in the Hierarchy palette). This will ensure a unique binding for the subform in the XML data. Implicit bindings can lead to undesired/unpredictable behavior and therefore isn’t recommended.
3. Add the required controls/qualifiers **in the order they should be applied**. Each of the CM controls has script that makes them do what they’re supposed to do. These scripts will execute in top-down hierarchy order (i.e. “document order”). Therefore, if you want content to be conditional *then* optional (so the LFEU cannot choose to include it unless a condition has been

met), you would first add the CM Conditional Content control, then the CM Optional Content control to the CM Object subform. If you wanted the content to be optional *then* conditional (so the LFEU chooses the include it, at which point the condition is evaluated and the content may or may not be displayed), you would first insert the CM Optional Content control, then the CM Conditional Content control. **The CM Content object is a flowed subform with a blue cmInstructions “drop target” to make it easy to drop controls and the content into it.** The following is an example of conditional-optional content (notice the **order** of the controls):

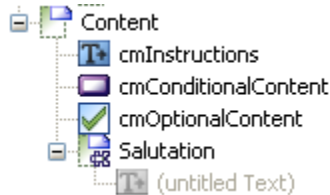


Figure 35: Conditional-optional content example

Note that mandatory content would simply not use any controls/qualifiers in the CM Content object.

4. Add the content that should be controlled by the CM Content object and the controls it contains.

Cleaning-Up cmBodyContent and cmDataFields

Before you start adding body content, you’ll need to remove the sample content included in the base letter template:

1. Delete the Content1 and Content2 subforms in cmBodyContent.
2. Delete the CostField field in cmDataFields (inside cmControlPage).

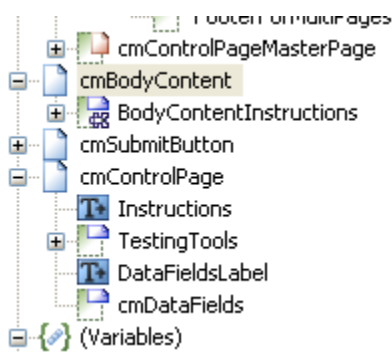


Figure 36: cmBodyContent and cmDataFields cleaned-up

Note that **you should leave** the BodyContentInstructions place-holder since it’ll make it much easier to drag and drop CM Content objects into cmBodyContent. cmBodyContent is a top-bottom flowed subform and without any content inside, it’ll have a height of zero, making it virtually impossible to drop anything into it!

Salutation

The first content we'll add is the **mandatory** Salutation paragraph.

1. Drag & drop a new CM Content object over the BodyContentInstructions subform. This will cause the CM Content object to be appended to the end of cmBodyContent's content (inside, after all existing children):

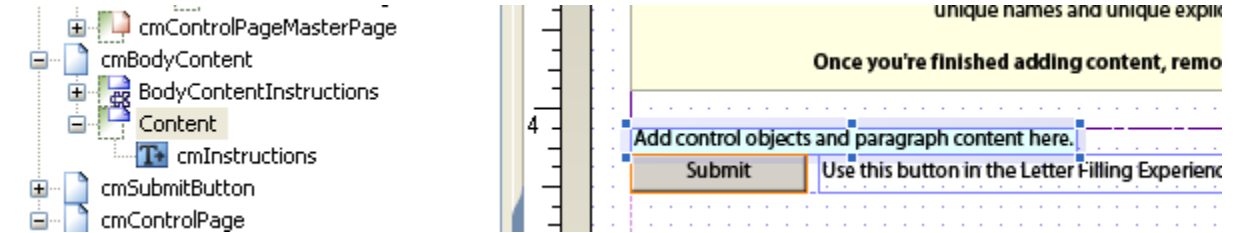


Figure 37: Adding a new CM Content object into cmBodyContent

2. Set the name of the CM Content object to "Cnt_Salutation".
3. Set the binding of Cnt_Salutation to "controls.salutation".
4. Drag & drop the Salutation.xdp content fragment **over the blue cmInstructions** inside

Cnt_Salutation:

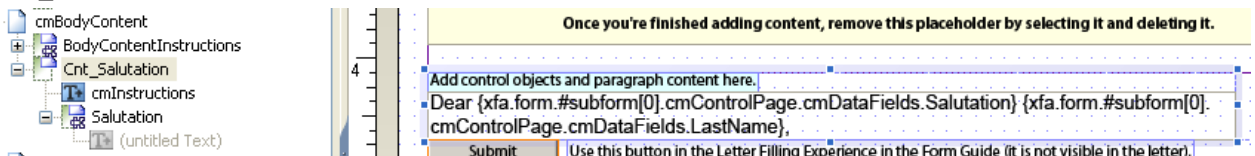


Figure 38: Adding the Salutation.xdp fragment as mandatory letter content

Don't worry about the floating field references just yet. We'll cover that soon when we get to the cmDataFields subform in the cmControlPage.

Introduction

Mandatory paragraph:

- Name: "Cnt_Introduction"
- Binding: "controls.introduction"
- Content: Introduction.xdp

Remember to start by dragging-in a new CM Content object from the CM Object Library and dropping it on top of the BodyContentInstructions place-holder.

Next Statement

Mandatory paragraph:

- Name: "Cnt_NextStatement"
- Binding: "controls.nextStatement"
- Content: Next_Statement.xdp

Donation

This is a conditional paragraph that should be included in the letter only if the initial investment amount is greater or equal to \$250,000.

1. Insert a new CM Content object and give it the following properties:
 - a. Name: "Cnt_Donation"
 - b. Binding: "controls.donation"
2. **Before adding the content** (though this is likely inconsequential but it's consistent and consistency is good!), add a new CM Conditional Content control to Cnt_Donation. If you already added the content, simply re-order the control to be prior to the content using the Hierarchy palette.
3. Set the condition by selecting the Cnt_Donation.cmConditionalContent button and activating the Script Editor palette:
 - a. Find the button's Initialize event (set the "Show" property to "Initialize").
 - b. Find the line that reads:


```
cmUtils.ConditionalContentInclusion(this, DataField, "operator", value);
```
 - c. Replace "DataField" with a reference to the field which will contain the data value for the initial investment. **Note that we haven't created this field in the cmDataFields subform yet** but you can still insert the proper expression:


```
xfa.form.resolveNode("#subform[0]").cmControlPage.cmDataFields.Amount
```

(The use of "#subform[0]" instead of "cm" keeps things generic. If the name of the root subform were to change, our script would still work – and it will, later on in the Data Fields section.)

How do I know where it'll be located and what its name will be? Well, I know that there's at least one paragraph (Introduction) that contains a floating field reference to a field named "Amount" in xfa.form.#subform[0].cmControlPage.cmDataFields. I also know that I intend to bind that field to the <initialInv> data value node so that's where I'll get the initial investment amount from in order to do the comparison.
 - d. Replace "operator" with ">=" since the initial investment must be "greater or equal" to \$250,000.
 - e. Replace "value" with 250000 (the minimum initial investment required for the donation plan):


```
cmUtils.ConditionalContentInclusion(this, xfa.form.resolveNode("#subform[0]").cmControlPage.cmDataFields.Amount, ">=", 250000);
```
4. Add the Donation.xdp content fragment to Cnt_Donation:

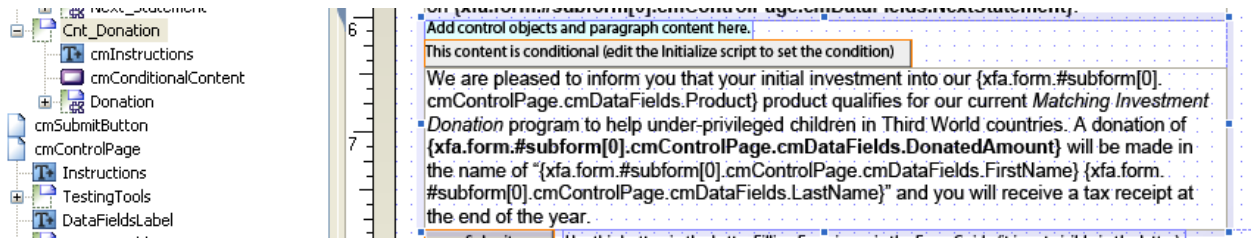


Figure 39: The conditional Donation paragraph content added to the letter template

Promo

This paragraph needs to be optional so that the LFEU can choose whether or not to offer it to the customer. Additionally, if included, it must be editable so that the LFEU can tweak it as necessary (perhaps they want to make the deal *even sweeter* for a particular customer).

1. Insert a new CM Content object and give it the following properties:
 - a. Name: "Cnt_Promo"
 - b. Binding: "controls.promo"
2. Insert a new CM Optional Content control into Cnt_Promo. We do this **first** because the content is "optional, then editable if included".
3. Insert a new CM Rich Text Editable Paragraph control into Cnt_Promo.
4. Add the Promo.xdp content fragment to Cnt_Promo.

Conclusion

The last piece is the conclusion. Remember that the LFEU must have the choice of one conclusion amongst any paragraph content in the Repository flagged as a "Finance Corp Conclusion" and that the LFEU must be able to edit the conclusion of their choice. Also remember that we tagged two paragraphs with the "Conclusion" tag in the "Finance Corp" category earlier on when we were creating the paragraphs with CC.

That's right, you guessed it: We'll use a **query** for this. Queries are a brand new technology developed by the CM Team which basically performs a live query against the Repository given specific parameters like a tag ID to search for or a creation date. Of course, you can build-up these queries by combining parameters such that, for example, it could search for all content tagged as "Conclusion" that was created on or after a particular date. Keep in mind that you can also create fragments out of the CM Query objects so that you can easily re-use queries in multiple letters.

1. This time, since we're dealing with a query, **start with a CM Query object**, not a CM Content object:

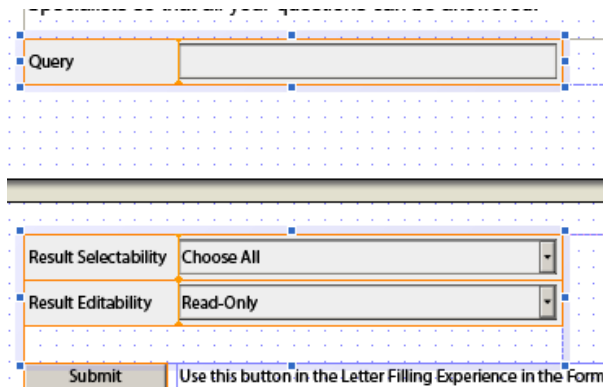


Figure 40: Inserting a new CM Query object in cmBodyContent

At this point, the CM Query object will likely break across the page. This is expected since the controls are visible at design time and therefore cause the cmBodyContent to be longer than a single page can hold. This problem will be rectified once we remove the BodyContentInstructions place-holder and even further in the PDF view when the controls will no longer be included in the layout (because they'll automatically be hidden).

2. Rename "cmQuery" to "Cnt_Conclusion" and set the binding to "controls.conclusion".
3. Select the cmQuerySpec field and look at its Initialize event using the Script Editor palette. The default Initialize event script sets the field's default value to a query specification which searches for content with a specific **tag ID**. You must replace "PUT_YOUR_TAG_HERE" with the **ID** of the tag you're searching for. Recall that we added a category and a tag to the "//cm/system/cm_default_tags.xml" file so that CC would get the new category and tag. At that time, we specified that the ID for the "Conclusion" tag was "**fc-conclusion**".
4. Select the cmResultSelect drop down list and set its "Object palette > Value tab > Default Value property" to "chooseOne" since the LFEU can only choose one of the conclusions.
5. Select the cmResultEdit drop down list and set its "Object palette > Value tab > Default Value property" to "editable" since the LFEU must be able to edit the conclusion they choose.

Setting Query Result Paragraph Spacing: By default, query results (paragraphs) will have a 12pt "below spacing". You can easily change this by setting the "Paragraph palette > Spacing section > Below property" on the cmParagraph text field, inside the cmQueryResult subform.

Spot Check: Finished Adding Content

Now that we're done with the content, we can delete the BodyContentInstructions place-holder that we were using as a drop target.

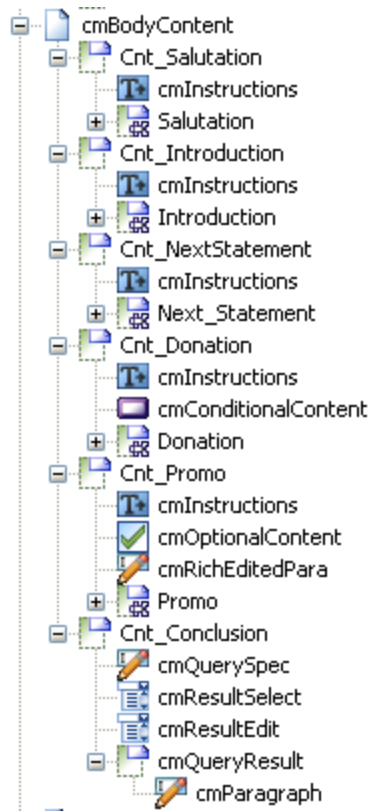


Figure 41: State of the Hierarchy after adding all the body content

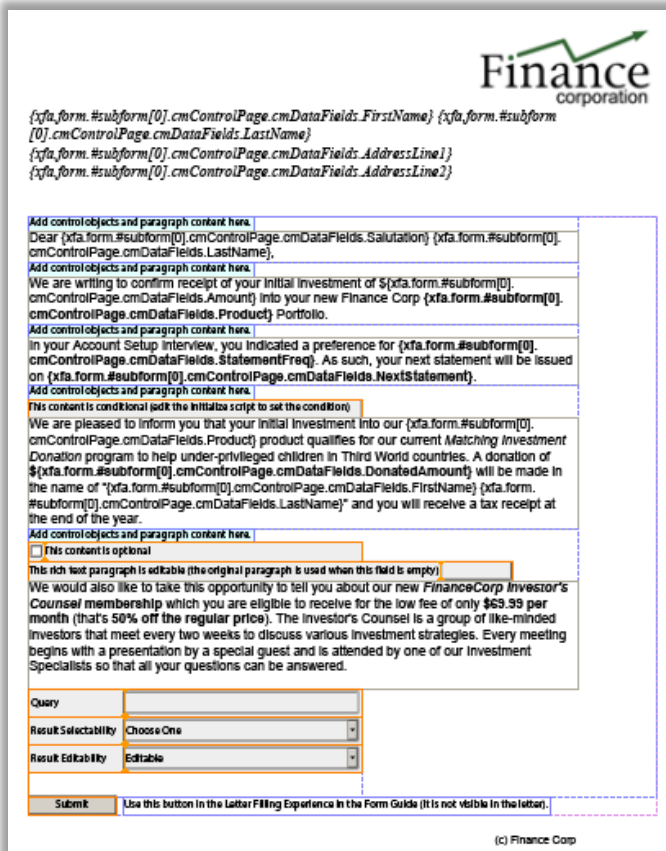


Figure 42: State of the Canvas after adding all the body content (showing only the letter template, not the cmControlPage)

Adding Data Fields

Now that the content has been added, the last step to complete the letter template is to add the data fields (i.e. the fields referenced as floating fields within the various content paragraphs). Note that we could've added these prior to adding the content – I just chose to do it this way around since it's easier to tell which fields are required by looking at the SOM expressions in the paragraphs, but maybe that's just me...

We need to add a data field for each floating field reference found in the included content into the cmDataFields subform. We must put them there specifically because when a field is added to paragraph content in CC, the reference is always

`xfa.form.#subform[0].cmControlPage.cmDataFields.FIELD_NAME`

The only part that ever changes is "FIELD_NAME" which becomes whatever name you gave to the field.

(Note that even query results may have floating field references which will need fields so make sure they're all covered. In our case, none of them do so we don't have to worry about it.)

For each data field that you add, make sure that you give it the **same name you gave to the floating field** when you created the paragraph content earlier in CC.

Finally, you will want to bind those fields to data. In our case, that'll be data coming from our "testdata.xml" file.

Example

To save time, we'll add preconfigured data fields from the attached "dataFieldsXfa.txt" file but I wanted to walk through the steps of adding one so that you understand what needs to be done.

To add a field for the floating field reference to the "first name" data:

1. Recall that we named the floating field "FirstName" when we created the various paragraphs that refer to it in CC (Address_Block.xdp, Donation.xdp).
2. Add a new text field to cmDataFields.
3. Give it the name, "FirstName".
4. This field will be bound to the <firstName> data value node so give it the explicit binding reference of "firstName".

Preconfigured Data Fields

For the purposes of speeding-up this sample, I've included the definition for all required data fields and their bindings in the attached file named "dataFieldsXfa.txt":

1. Open the file in Notepad.
2. Select all its contents and copy it.
3. Switch to Workbench.
4. Select the cmDataFields subform.
5. Place your cursor **inside** the select subform.
6. Paste using "Ctrl + V".

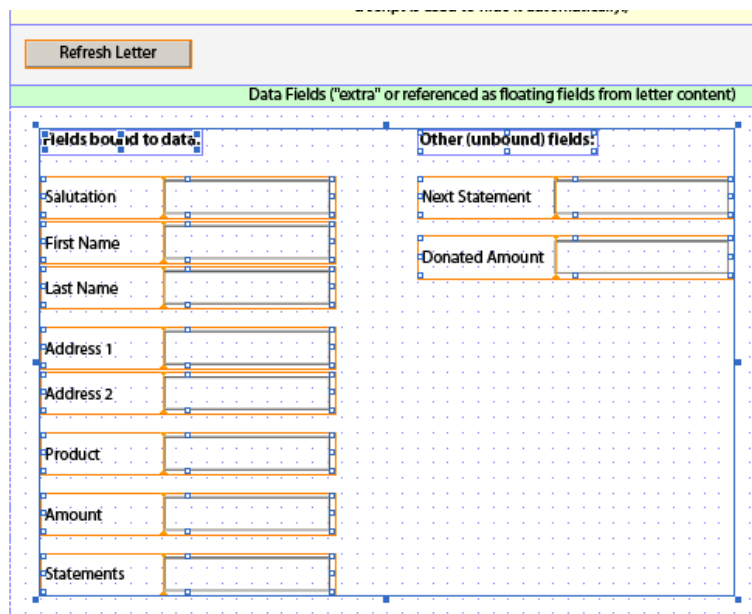


Figure 43: Preconfigured data fields inserted into cmDataFields subform

It's worth noting here that there are two data fields, "NextStatement" and "DonatedAmount" which are not bound to data. These are calculated values from Initialize scripts on each field based on values set in other (bound) data fields.

If you inspect each field, you'll see that they're all named according to the floating field references and all the fields in the left column are bound to data value nodes from "testdata.xml".

Root Subform Binding

The last binding we need to specify is on the root subform, named "cm" by default in the cmBaseTemplate.xdp. Designer doesn't let you edit much other than its name but it's important to do so since its name, via implicit binding, will be the name of the root data node in the XML data submitted back to a process, should you eventually create one for this – or another – letter.

In our case, we want our data wrapped in a <fcFollowUp> root data node so we need to set the root subform's name to "fcFollowUp" by renaming it using the Hierarchy palette:



Figure 44: The root subform renamed to "fcFollowUp" to match the root node name in the XML data

Testing/Debugging the Letter Template

Our goal is to simulate data from a data-collection application (e.g. the Custom Communications CSR AIR app) being merged into our letter template in order to start the LFE. To do this, we're using an XML data file, "testdata.xml", which we've previously uploaded to "//cm/samples/fcFollowUp/doc/testdata.xml" in the Repository.

To test our letter in PDF Preview (and eventually to test the LFE), we simply need to set that XML data file as the preview data file for our letter template:

1. Go to "Edit menu > Form Properties > Preview tab" (in Workbench).
2. Choose the "testdata.xml" as the Data File.
3. Verify that the "Preview Type" property is set to "Interactive Form" and that the "Preview Adobe XML Form As" property is set to "Dynamic XML Form":

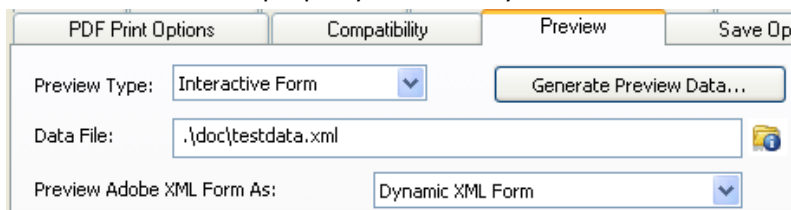


Figure 45: Setting-up the preview data

4. Click OK and then save your form.

You can now preview the letter template using the **Preview PDF** tab:

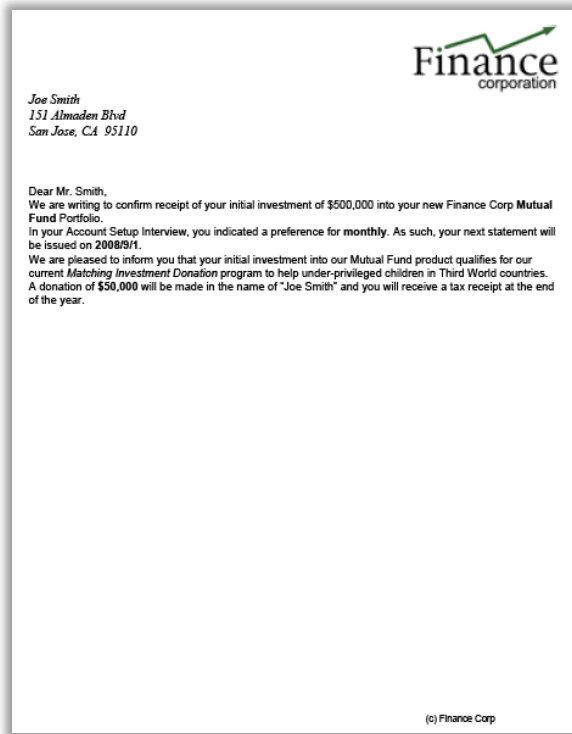


Figure 46: Previewing/testing the letter template as PDF

You should be able to see that:

- the data from the XML data file is coming through in the floating field references in the various paragraphs by way of the fields in cmDataFields;
- the conditional "Donation" paragraph is automatically included since the initial investment amount is greater than \$250,000;
- the optional "Promo" paragraph and query-selected "Conclusion" paragraphs are not included in the letter (this is because the "Promo" paragraph hasn't been included and the query hasn't been executed – the LFE is required for executing queries).

Further Testing/Debugging

We can take this a little further in the PDF view by setting the *ShowLetterControlPage* and *ShowControlFieldsInPdf* debug form variables to prevent the cmControlPage and all the CM Controls (optional, conditional, editable, query) from being hidden when the PDF is initialized:

1. Go to "Edit > Form Properties > Variables" (in Workbench).
2. Set the value of both "ShowLetterControlPage" and "ShowControlFieldsInPdf" form variables to 1.

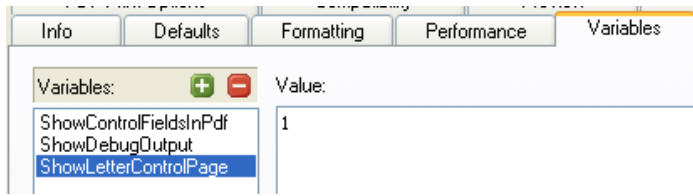


Figure 47: Setting debug form variable values to 1

3. Click OK.
4. Preview the letter template using the Preview PDF tab:

The screenshot shows a web form for Finance Corporation. At the top right is the Finance Corporation logo. Below it is the contact information for Joe Smith: 151 Almaden Blvd, San Jose, CA 95110. The main content is a letter preview. The letter starts with 'Dear Mr. Smith,' and mentions a \$500,000 investment. There is a section for a 'Promo' paragraph with a checkbox and a 'Submit' button. Below the letter preview are several control fields: 'Query' (a text box with a value), 'Result Selectability' (a dropdown menu), 'Result Editability' (a dropdown menu), and 'Submit' (a button). At the bottom of the form is a 'Data Fields' section with two columns: 'Field to bound to data:' and 'Other (not used) fields:'. The 'Field to bound to data:' column contains fields for Salutation (Mr.), First Name (Joe), Last Name (Smith), Address 1 (151 Almaden Blvd), Address 2 (San Jose, CA 95110), Product (Mutual Fund), Amount (500,000), and Statements (monthly). The 'Other (not used) fields:' column contains fields for Next Statement (jose@f1) and DonorAAmount (50,000). At the bottom of the form are two buttons: 'Refresh Letter' and 'Test Submission'.

Figure 48: cmControlPage and control fields visible in PDF view after setting debug form variables

Now you can test different scenarios:

- Check the box for the optional “Promo” paragraph in the letter, then click the “Refresh Letter” button: The “Promo” paragraph appears – just like if it was included via the LFE.

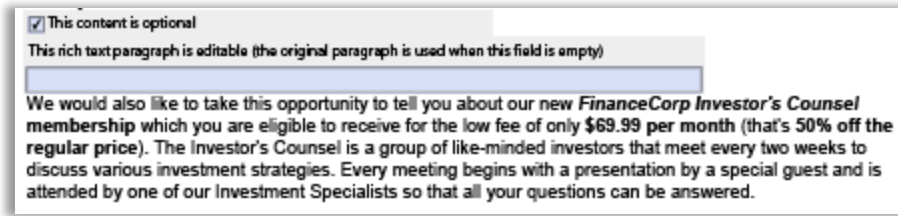


Figure 49: "Promo" paragraph appears after checking the box and refreshing the letter

Also notice that the "editable paragraph" control is now displayed with the caption at the top and has a larger input area than at design-time (you might also notice the difference in width of the "Promo" paragraph itself...). This is done automatically when the control fields are visible so that it's easier to enter an override for an editable paragraph:

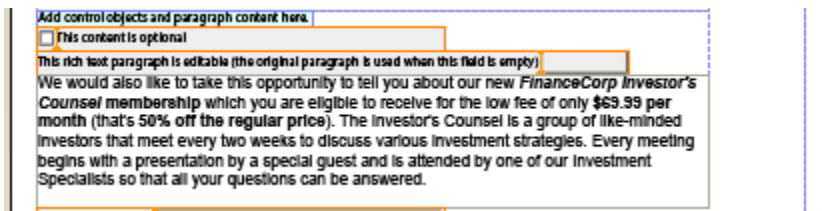


Figure 50: Comparing design-time view of optional-editable "Promo" paragraph

- If you then enter text in the field and click on the "Refresh Letter" button, you'll see that what you entered is now set as the "Promo" paragraph, overriding what was there before. To insert rich text, press "Ctrl + E" to get Acrobat's Formatting toolbar:

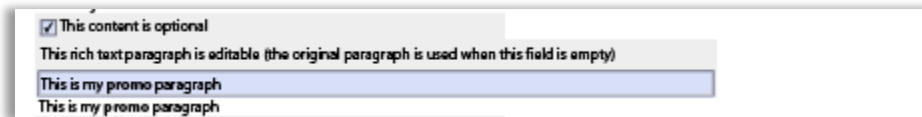


Figure 51: Override to editable "Promo" paragraph in PDF view

- You can also modify the initial investment data to see what happens if it's below \$250,000 by changing the value directly in the "Amount" field in the cmDataFields subform on the cmControlPage to, say, 100000, and refreshing the letter: You guessed it – the conditional "Donation" paragraph disappears from the letter!
- Finally, you can see what would be submitted from the LFE by clicking on the "Test Submission" button. This is a simple email submit button that will cause an email with an XML data file attached to it to be created. You can then inspect the XML data file to make sure your bindings are set correctly.

What you can't do here (at least not easily) is experiment with the query results. That's something we hope to address in our next release.

Exposing Debug Output

Another debugging feature in cmBaseTemplate-based solutions, is the ability to see debug output generated by all the various CM-related components in your letter template. This is done by setting the *ShowDebugOutput* debug form variable to 1 using the “Form Properties” dialog.

Once this is set, preview the letter template as PDF and press “Ctrl + J” to view the JavaScript Console. You should see something like this:

```
{cm} Info > --- xfa[0].form[0].fcFollowUp[0].cmBodyContent[0].Cnt_Salutation[0]:
{cm} Info > --- xfa[0].form[0].fcFollowUp[0].cmBodyContent[0].Cnt_Introduction[0]:
{cm} Info > --- xfa[0].form[0].fcFollowUp[0].cmBodyContent[0].Cnt_NextStatement[0]:
{cm} Info > ---
xfa[0].form[0].fcFollowUp[0].cmBodyContent[0].Cnt_Donation[0].cmConditionalContent[0]:
{cm} Info > --- xfa[0].form[0].fcFollowUp[0].cmBodyContent[0].Cnt_Donation[0]:
{cm} Info > ---
xfa[0].form[0].fcFollowUp[0].cmBodyContent[0].Cnt_Promo[0].cmOptionalContent[0]:
{cm} Info > ---
xfa[0].form[0].fcFollowUp[0].cmBodyContent[0].Cnt_Promo[0].cmRichEditedPara[0]:
{cm} Warning > cmUtils.ControlParagraphEditability -- Invalid content subform object.
Either it has already been removed by a previous control field or it's not in the form.
{cm} Info > --- xfa[0].form[0].fcFollowUp[0].cmBodyContent[0].Cnt_Promo[0]:
{cm} Info > --- xfa[0].form[0].fcFollowUp[0].cmBodyContent[0].Cnt_Conclusion[0]:
{cm} Info > ---
xfa[0].form[0].fcFollowUp[0].#pageSet[0].MasterPageForFirstPage[0].cmHeaders[0]:
{cm} Info > ---
xfa[0].form[0].fcFollowUp[0].#pageSet[0].MasterPageForFirstPage[0].cmFooters[0]:
{cm} Info > ---
xfa[0].form[0].fcFollowUp[0].#pageSet[0].MasterPageForFirstPage[0].cmHeaders[0]:
{cm} Info > ---
xfa[0].form[0].fcFollowUp[0].#pageSet[0].MasterPageForFirstPage[0].cmFooters[0]:
```

You want to be certain that all you see are informational (“{cm} Info”) messages. You should be weary if you see warnings or even errors. If you do see them, they should be accompanied with some helpful text to explain what went wrong and how you might fix it.

For example, there’s one warning in the output above which is indicating that an editable paragraph (based on the function “ControlParagraphEditability”) couldn’t be located in the form. What’s happening is that due to the **order of execution**, the CM Optional Content control for the “Promo” paragraph is executing first and, since it’s not checked, it’s removing the “Promo” paragraph from the letter. Then the CM Rich Text Editable Paragraph control executes but can’t find the “Promo” paragraph so it issues this warning. It doesn’t treat this as an error because the script can’t know whether it was removed by a

previous control or if you actually did forget to include it in the CM Content object that contains the control.

See the Troubleshooting section for another example on using the debug output to resolve a common problem.

Building the Letter Filling Experience

Now that the letter template is built, we need to build the LFE. This part of the CM solution is provided via a Form Guide so we'll use Guide Builder as our design tool:

1. Reset the 3 debug form variables to 0 if you had enabled them (set them to 1) while going through the previous section on Testing/Debugging the Letter Template.
2. Save the fcFollowUp.xdp letter template if you haven't done so yet (I hope you have – at least once!).
3. Launch Guide Builder by choosing the "Tools > Create or Edit Form Guide".

Our LFE will have two panels: One for including the optional "Promo" paragraph and editing its content and another for the Conclusion query. The rest of the paragraphs are either conditional or mandatory, neither of which require UI in the LFE.

Form Guide Setup

Part of what you imported into the Repository earlier on when you were installing the CM.M2 kit were two SWC libraries: `"/cm/shared/res/cmModel3.swc"` and `"cmGuideLayouts3.swc"`. These libraries are essential to building LFE form guides. Just like the `"cmUtils.xdp"` script object fragment must be present in all letter templates, these SWCs must be included in all LFE form guides. They include the CM Wrapper and various panel layouts and custom controls (components) that you'll need to build your LFE.

1. In Guide Builder, click the "Add Custom Library" button:

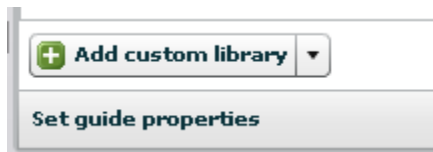


Figure 52: "Add Custom Library" button in Guide Builder

2. Choose the `"/cm/share/res/cmModel3.swc"`.
3. Repeat this for the `"/cm/share/res/cmGuideLayouts3.swc"`.

With both libraries included, you'll now notice a new wrapper in the "Guide Layouts" panel on the right: "Cm Correspondence Management":

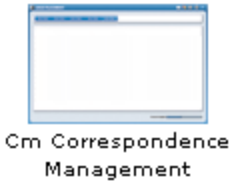


Figure 53: New CM wrapper in Guide Layouts panel

Configuring the Guide Object

Set the Guide Layout for your LFE to the CM-specific guide layout:

1. With the “New Guide” object selected on the left, choose the “Cm Correspondence Management” Guide Layout on the right.
2. While we’re editing the guide object, give it a nicer name: “FC Follow-Up Letter Builder”

This layout gives the CM “side-by-side” view where you see the LFE on the left and a preview of the letter in PDF on the right. It also provides various support functions for some of the CM-specific panel layouts and controls.

Now we need to tell the LFE form guide which button to use to submit the built letter’s (result of the LFE) XML data definition (the initiating data from the data collection app combined with the data collected by the various CM controls while the LFEU was building the letter using the LFE). As we saw earlier, there’s a “cmSubmitButton” subform in our letter template which includes two buttons:

- *cmPdfSubmitButton*: This is the actual submit button. You can configure it to submit any type of data that you like (XML, XDP, PDF, etc.) via the “Object palette > Submit tab”.
- *cmGuideSubmitButton*: This button is meant to be used in the LFE as the submit button and it’s only reason for existence is to programmatically call the *cmPdfSubmitButton*’s submit action. This indirection is necessary since XDP and PDF data cannot be submitted directly from a form guide. By executing the *cmPdfSubmitButton*’s submit action, the submission actually occurs from the (possibly hidden) PDF running in Acrobat which is capable of submitting much more than plain XML data should your CM solution need to do so.

In our case, we just need to submit XML data so the default settings on these buttons are fine. We just need to set the *cmGuideSubmitButton* as the submit button that our LFE should use:

1. Activate the “Set guide properties” panel on the right.
2. Set the “Submit from” property to “Guide”.
3. Set the “Submit Button” property to “cmGuideSubmitButton”.

So we’re effectively *tricking* the LFE form guide into thinking it’s doing the submission however the submission is still occurring from the PDF (Acrobat).

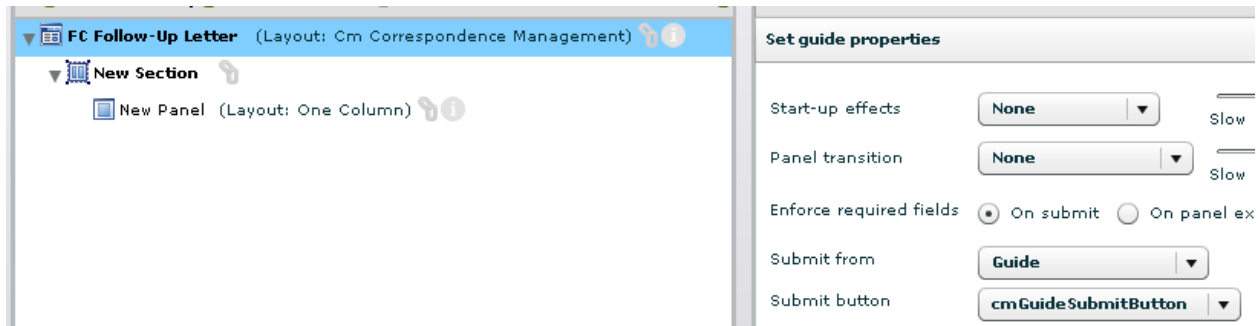


Figure 54: LFE form guide setup with new name, guide layout and submit button

Handling the Optional/Editable Promo Paragraph

For this paragraph, we'll use a simple panel that has a checkbox for including it and a special edit box for editing it. We could make it more complex, perhaps putting the edit box on a different panel that only appears if the checkbox is checked, but that's not CM-specific and therefore beyond the scope of this document.

1. Rename the existing panel to "Promo".
2. Click on "Add or Bind Fields" and add a guide text object with the following content: "Please choose whether to include the promotional paragraph in the letter:". Set the text to "Verdana, 12pt, Bold".
3. Back in "Add or Bind Fields", find the "Cnt_Promo.cmOptionalContent" checkbox (Aren't you glad you gave unique names to your CM Content objects earlier? You would be having a hard time finding this if you hadn't...) and drag it into the panel after the guide text object you just added.
4. Locate the "#draw" object inside the "Promo" subform (which was the "Promo" paragraph fragment reference) and link it to the cmOptionalContent checkbox's caption.
5. With the cmOptionalContent checkbox selected, switch to "Set Properties" and set its "Display field as" property to "Cm Optional Paragraph".
6. Back in "Add or Bind Fields", find the "Cnt_Promo.cmRichEditedPara" text field and drag it into the panel after the cmOptionalContent checkbox.
7. Override the cmRichEditedPara's caption with the following content: "Please provide any edits you find necessary:". Set the text to "Verdana, 12pt, Bold".
8. With the cmRichEditedPara text field still selected, set the "Display field as" property to "Cm Xfa Rich Text Editor". This is a special control that translates Flex Rich Text into XFA Rich Text so that you can edit the paragraph in the LFE (Flex Rich Text) and see your changes appear in the letter (XFA Rich Text).
9. Set the "Caption position" property to "top" to give the rich text editor as much room as possible in the panel.
10. Set the "Field height" property to "100%" (otherwise the rich text editor will be too small to work with in the LFE).

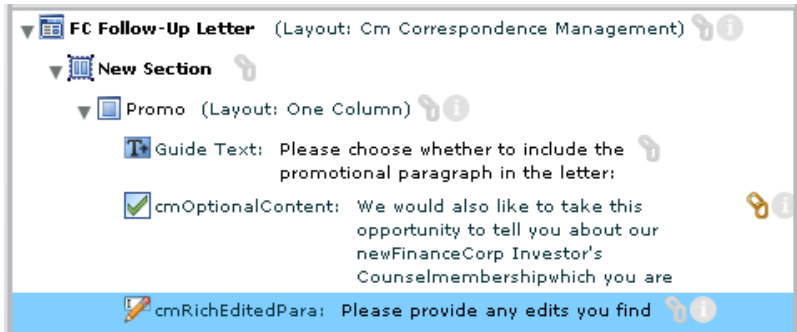


Figure 55: Promo paragraph panel defined

Configuring the Conclusion Query Panel

The second (and last) panel in the LFE will be a panel that will present the Conclusion query results and let the LFEU choose which Conclusion to include in the letter. To do this, we'll use a special panel layout called a "Cm Query Controller":

1. Add a new panel. Give it the title, "Conclusion".
2. Set the panel layout to "Cm Query Controller".

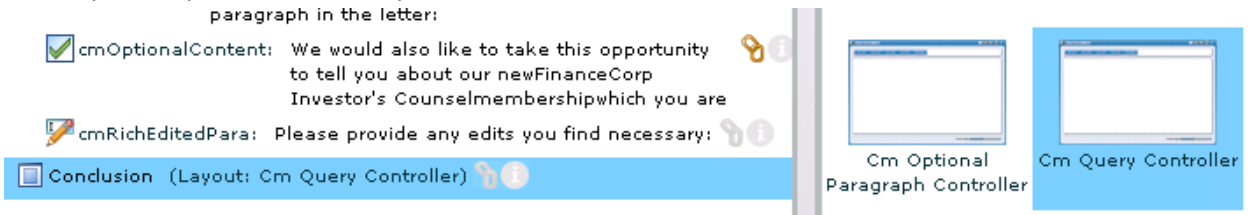


Figure 56: Setting Conclusion panel layout to "Cm Query Controller"

The Cm Query Controller layout has special functions that will execute the query specification you specify simply by including the "cmQuerySpec" field that contains the query you want executed in the panel. It uses our CM.M2 Model API (cmModel3.swc) to execute the query against the Repository and retrieve the paragraph content.

1. Find the "Cnt_Conclusion.cmQuerySpec" field in the "Add or Bind Fields" panel and drag it into the Conclusion panel:



Figure 57: cmQuerySpec added to the Conclusion query panel

That's it! Query panels are relatively easy to setup.

1. **Click "Apply"** in Guide Builder and then **save the form** in Workbench.

Note that we haven't done any customization to the appearance of the LFE (the form guide). Customization is done in the same way you would normally customize a form guide and is beyond the scope of this document.

Also note that the CM.M2 kit includes the **full source** for the wrapper, layouts and controls included in the "cmGuideLayouts3.swc" library for further customization using FlexBuilder. You can find it in the "//Human Interactive Correspondence/libraries/cmLayouts" folder in the CM kit.

Running the Letter Filling Experience

With the "testdata.xml" file set as the Preview Data File in the letter template, running the LFE in Guide Builder Preview mode will cause the test data to be loaded into the LFE, simulating the data having been merged into the form via the process.

1. Click on the Preview tab in Guide Builder.
2. Check the "Include PDF Preview" option so that the PDF version of the letter is included in the preview. The Cm Correspondence Management guide layout has a side-by-side view that shows the LFE (form guide) on the left and the letter on the right (as a WYSIWYG preview of the letter being built).
3. **Uncheck** the "Quick preview" option otherwise you'll get compilation errors when Guide Builder attempts to build the SWC for the LFE.



Figure 58: Guide Builder Preview options

4. Click the "Preview" button to run the LFE.

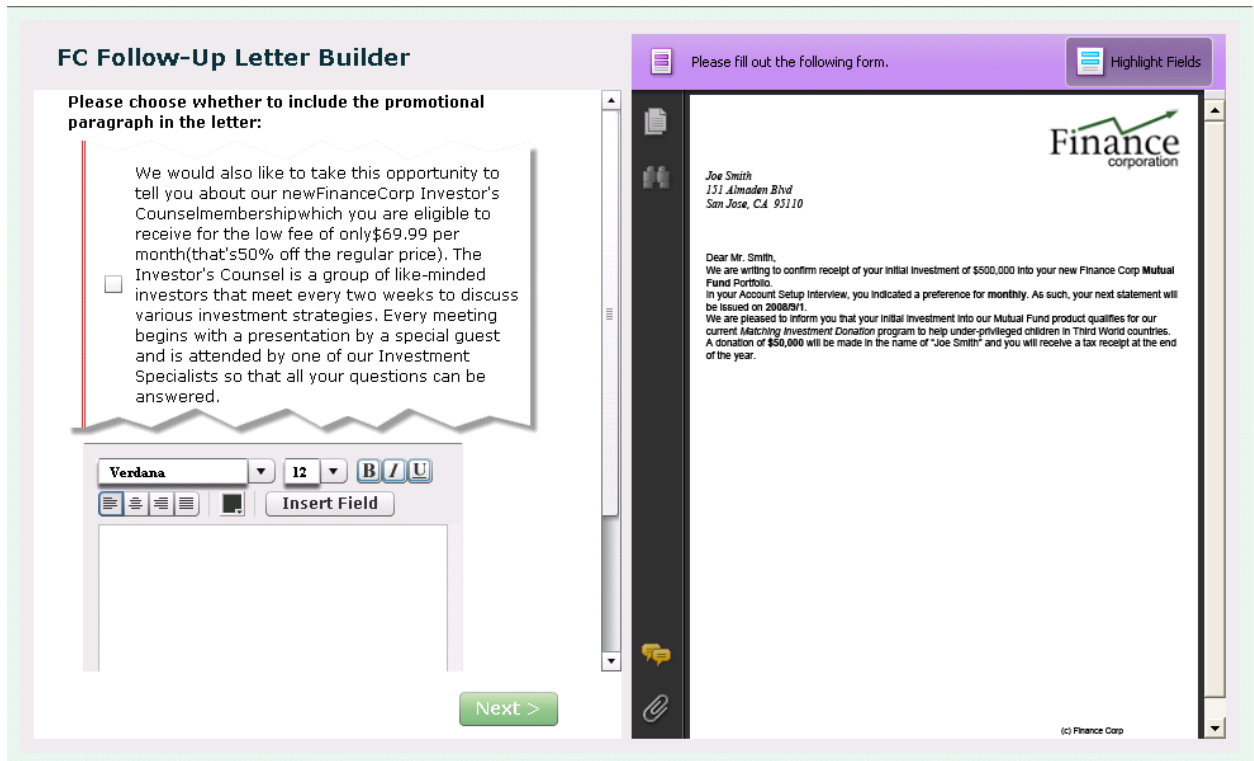


Figure 59: LFE running as "Guide Preview"

5. Choose to include the "Promo" paragraph by checking the box.
6. Change the promotion to "Given your initial investment of \$__Amount__, you will receive a new iPod Touch." To insert the value of the "Amount" field, simply use the "Insert Field" button as you would have if you were doing this in Content Creator – notice how the UI has changed and now suggests existing field names (those found directly inside the cmDataFields subform) rather than letting you enter a new (floating) field name:

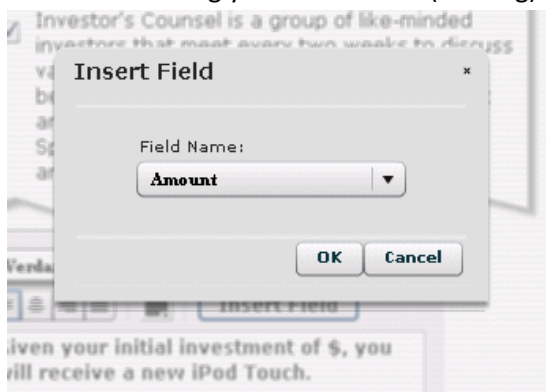


Figure 60: Inserting the "Amount" field's value into the new promo paragraph

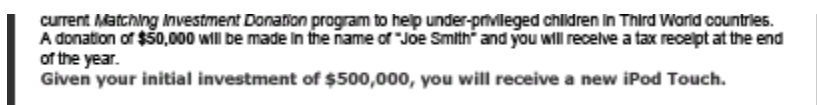
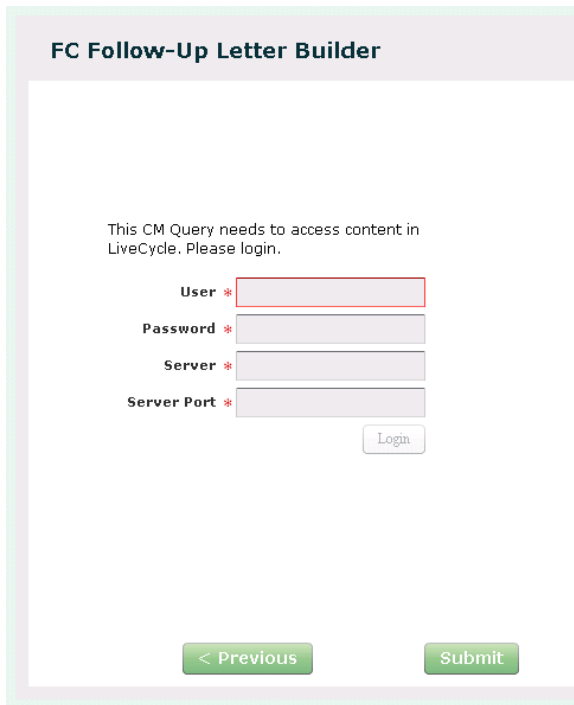


Figure 61: New promo paragraph in the letter

7. Click “Next” to choose the conclusion. You will be presented with a LiveCycle Login panel to establish a connection to the Repository. If you were running this LFE within Workspace, the LFE would automatically pick-up the current session and bypass the login panel:



The screenshot shows a web interface titled "FC Follow-Up Letter Builder". Below the title, there is a message: "This CM Query needs to access content in LiveCycle. Please login." Below this message are four input fields, each with a red asterisk indicating a required field: "User", "Password", "Server", and "Server Port". To the right of the "Server Port" field is a "Login" button. At the bottom of the form are two green buttons: "< Previous" and "Submit".

Figure 62: LiveCycle Login panel in the LFE

8. Login to your LiveCycle server. Once you login, the query will execute immediately and you will be presented with the results (2 paragraphs since two were tagged as “Conclusion” earlier using Content Creator):

FC Follow-Up Letter Builder

Paragraph Content

We greatly value your business and are happy to be working to help you achieve your investment goals. Please do not hesitate to contact us at (800) 555-3040 if you have any questions with regards to your investments at *FinanceCorp*.

Sincerely,

The FinanceCorp Investment Team

Thank you for choosing *FinanceCorp* to help you achieve your investment goals. Please do not hesitate to contact us at (800) 555-3040 if you have any questions with regards to your investments at *FinanceCorp*.

Yours truly,

The FinanceCorp Investment Team

< Previous
Submit

Figure 63: Conclusion Query results in the LFE

9. Notice that radio buttons are provided to make the selection because the “result selectability” was set to “chooseOne”. Choose the result which begins with “Thank you for choosing...” (there is no control over the order in which results are presented so make sure you choose the right one). The paragraph is immediately inserted into the letter:

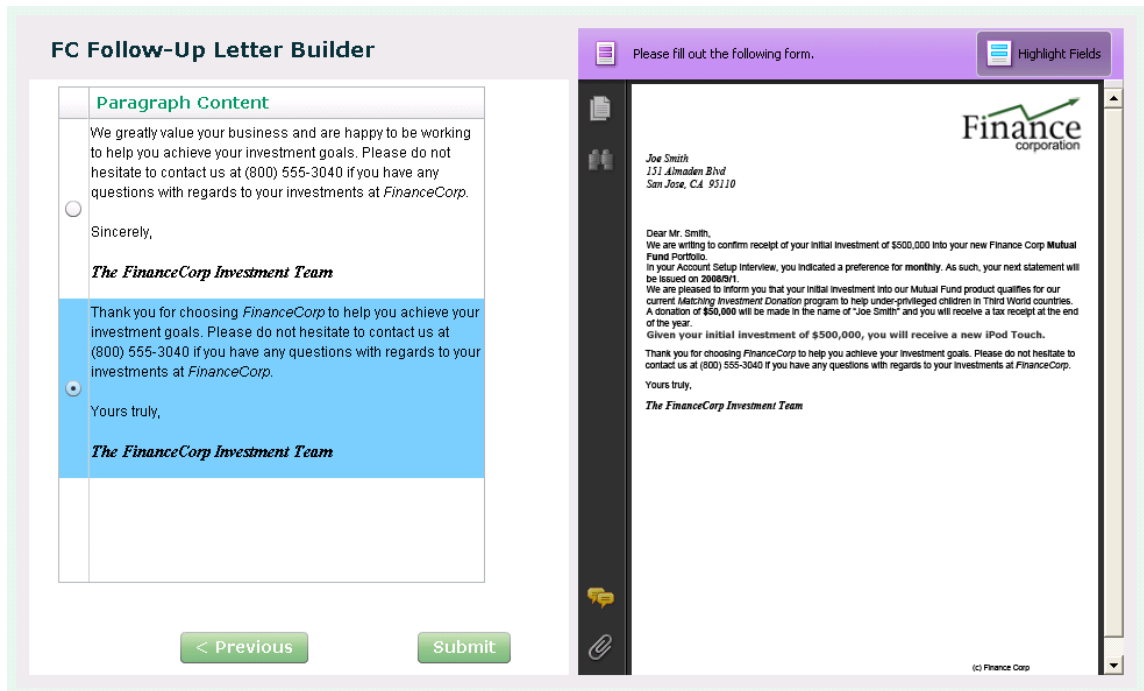


Figure 64: "Thank you" conclusion chosen for the letter

10. Mouse over the selected conclusion on the left. An overlay with an "edit" icon will appear. Click the icon to open the editor (since we set the query's "result editability" to "editable"):

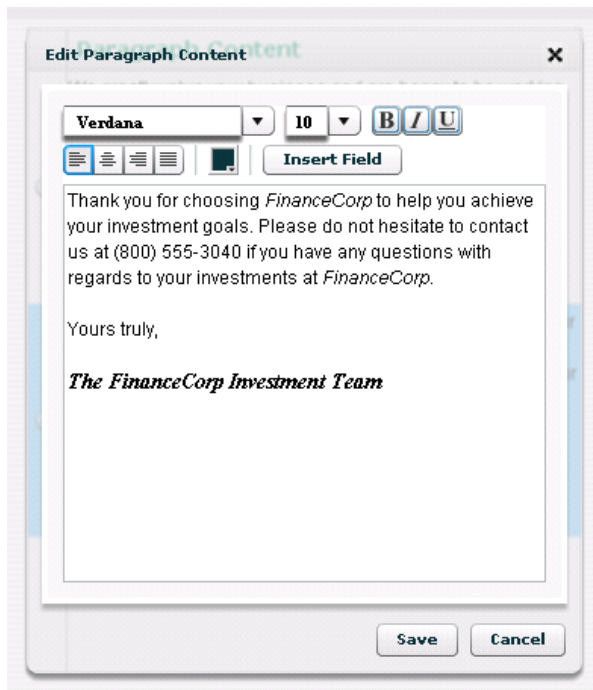


Figure 65: Query result editor in the LFE

11. Insert the text “, __Salutation__ __LastName__” at the end of the first sentence in the same way you edited the promo paragraph earlier and click on the Save button. The modification to the paragraph (query result) are automatically made to the letter as well:

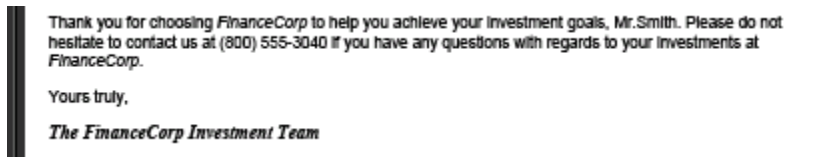


Figure 66: "Mr. Smith" added to the end of the first sentence in the letter's conclusion paragraph

The last step to the LFE would be to use the Submit button however we haven't set that up to do anything special in this example. Upon clicking the Submit button in a live solution, the data would be submitted back to the LiveCycle process that produced the LFE and the process would continue (perhaps it would merge the data back into the letter, produce a PDF/A document from it and email it to the client).

You may have noticed a paragraph spacing problem in the letter (with the paragraphs preceding the conclusion). This is caused by a bug currently under investigation whereby the default paragraph "below spacing" for content authored in Content Creator isn't respected by the XFA Text Engine. To learn how to resolve this problem and other typical issues that may arise, please see the Troubleshooting section.

Troubleshooting

In case things go wrong, this section contains a series of typical issues and ways in which they can be resolved.

Can't login to Content Creator?

Try the debug player – there is a possible bug related to security changes in Flash Player 9.0.124.0 and the debug player seems to be more lenient (so far).

Paragraphs aren't spaced correctly in the letter?

There is a bug, currently under investigation, where the XFA Text Engine is not respecting the default paragraph "below spacing" of 12pt automatically set on all paragraphs authored in Content Creator.

The first solution is to edit each fragment (paragraph) using Designer and reset the "below spacing" (**even if** it looks like it's already set to "12pt" as it should – just re-enter "12pt" in the property box and hit the Enter key).

The second solution is to use Content Creator to add a blank line at the end of all paragraphs.

We hope to have this issue resolved in the next version of the kit.

Red border around CM Content or CM Query object instance?

You may have forgotten to set a unique binding reference on the CM Content or CM Query subform. Preview the letter with the "ShowDebutOutput" debug form variable enabled (set to "1") in Designer

and check the debug output in the JavaScript Console to see what the error is and which subform it relates to.

Getting a compilation error when previewing the LFE using Guide Builder?

This is usually related to using the “quick preview” option which isn’t recommended.