Table of Contents

The ADPDemo Sample Application 1

Scan (VScan or NScan) 5

Processing 6

SendPageToADP action 10

Guidance for SendPageToADP 13

Document classification considerations 13

Keeping ADP fields distinct from your own Datacap fields 14

Which fields should I use? 15

Configuring for throughput 16

Handling sensitive data 18

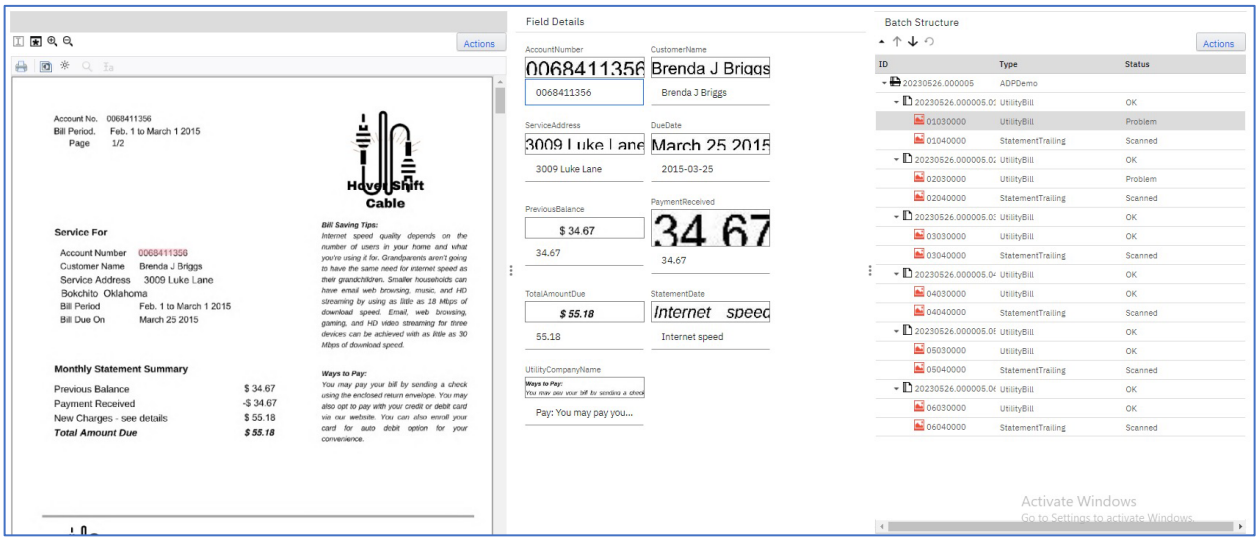
PostProcessing 18

2023-10-06 update: Deeper discussion of the parameters to SendPageToADP and SpreadTheADPResults, plus table of contents
2023-08-24 update: Now supports single and multi-page tiff files as input, not just PDFs.

The ADPDemo Sample Application

This sample application demonstrates how the ADP Connector can be used to send documents from Datacap to ADP, get the results back, and use those results in Datacap.

A view of the results of this application in Datacap Navigator is shown below – note that the document classification and all of the fields are from ADP but are available for processing in Datacap:



The ADPDemo sample application contains relatively bare-bones processing, but can be used as the basis for a more robust production application utilizing the strengths of both Datacap and ADP.

This sample application is written for multi-page utility bills, but can easily be changed to handle other types of documents.

The Datacap workflow consists of the following task profiles:

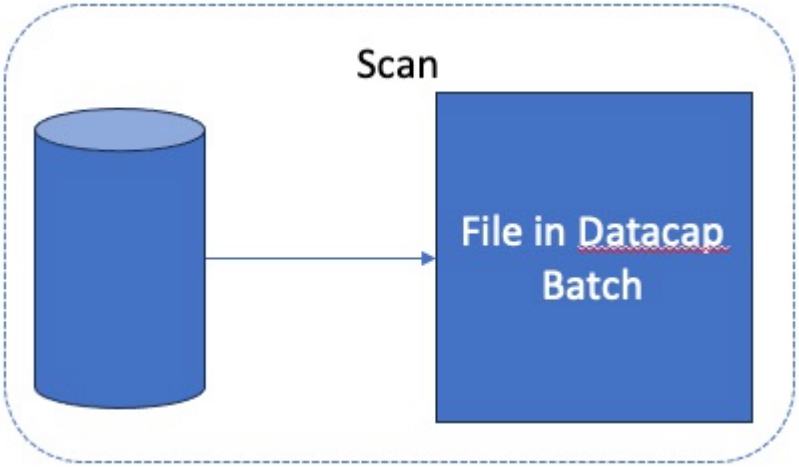
- Scan
- Processing
- Post-Processing
- Verify
- Export

Scan, Processing and Post-Processing are described in more detail in the following sections, but it will be helpful to have an overview of the processing before getting into the details.

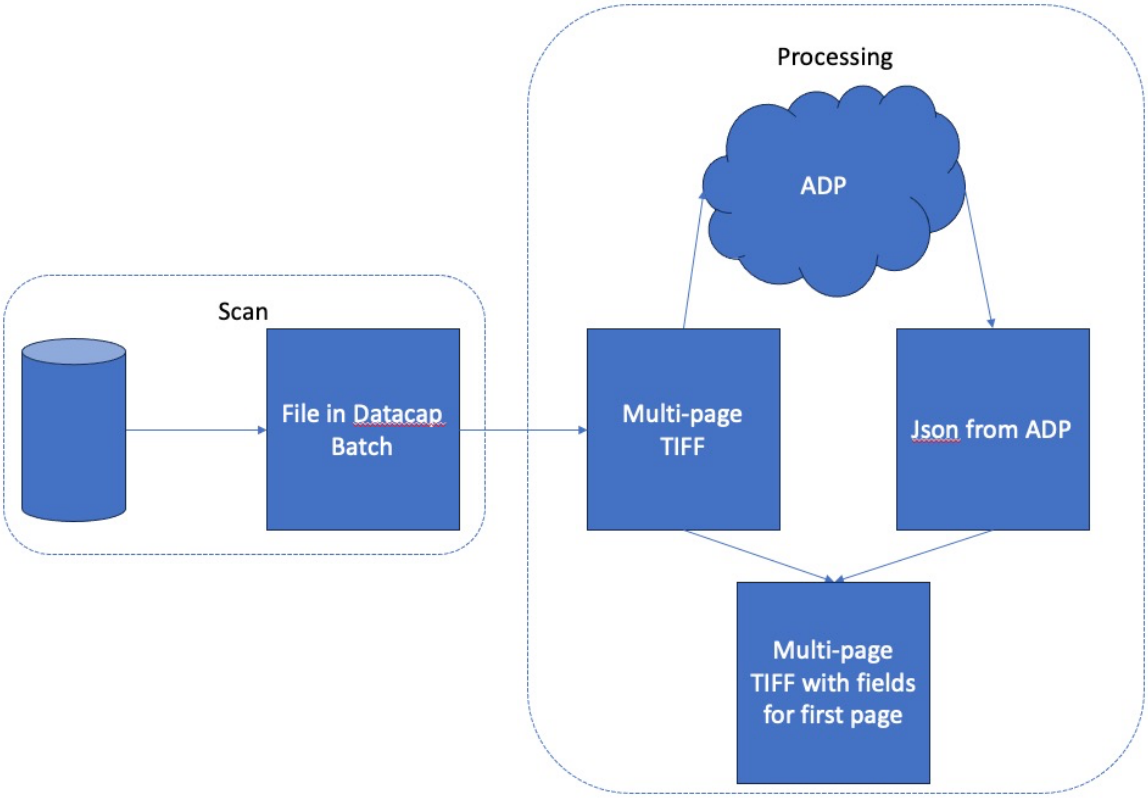
Documents being processed by Datacap and ADP can either be single-page or multi-page, and fields to be extracted from them can exist on any page. Since ADP treats each file it processes as a single document, it's important to ensure that the files sent to ADP contain all the pages associated with the document. Furthermore, when the results are finally displayed to the user for verification, it is helpful to have all of the fields shown together, rather than shown page by page.

Because of these considerations, the high-level steps (task profiles) in the ADPDemo sample application perform the following:

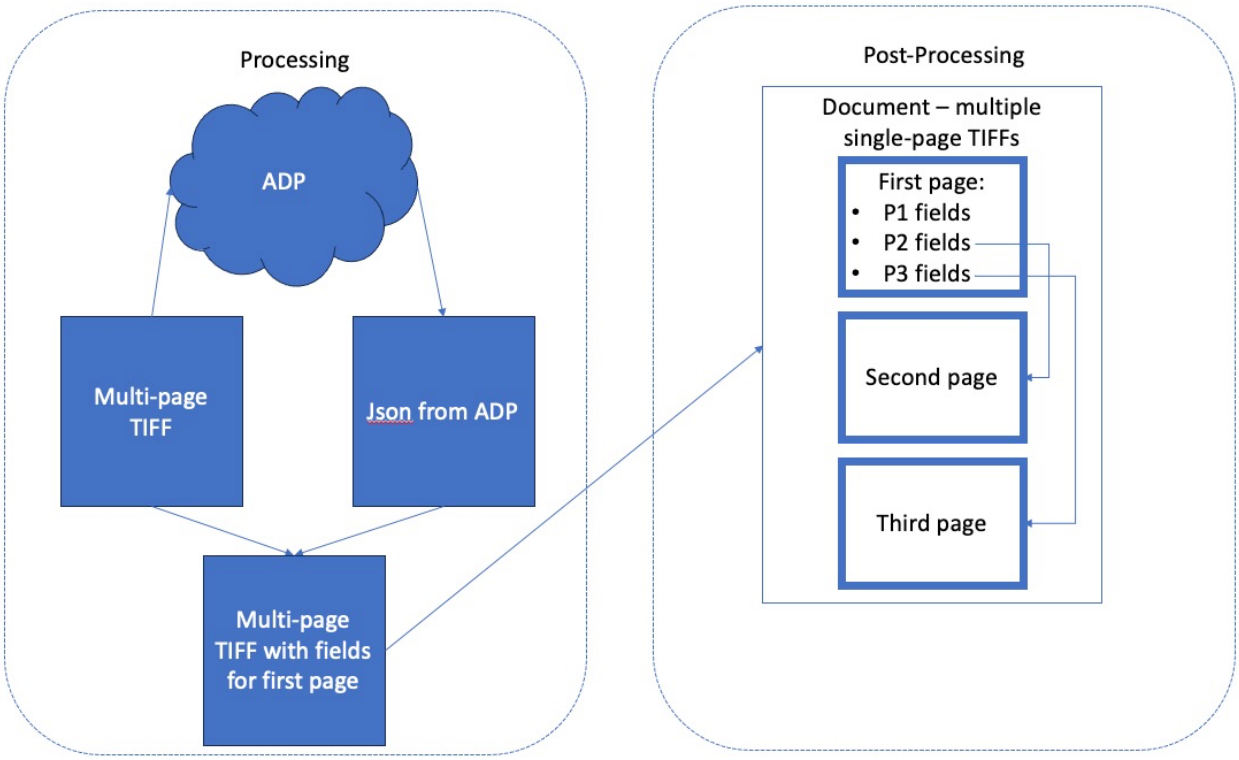
- The Scan step ingests the files into Datacap. The assumption for the sample application is that each input file is a single document – whether a single-page or multi-page file.



- The Processing step converts each scanned file to TIFF (either single-page if there was only one source page, or multi-page), sends the resulting file to ADP for processing, and sets the page type to the document classification assigned by ADP. It also dynamically creates fields (with their values) to the file based on what ADP found for the first page. Note that if the scanned file was a single-page document, you could take this result and perform validation on the data as-is, without the document-manipulation processing of the Post-Processing step.



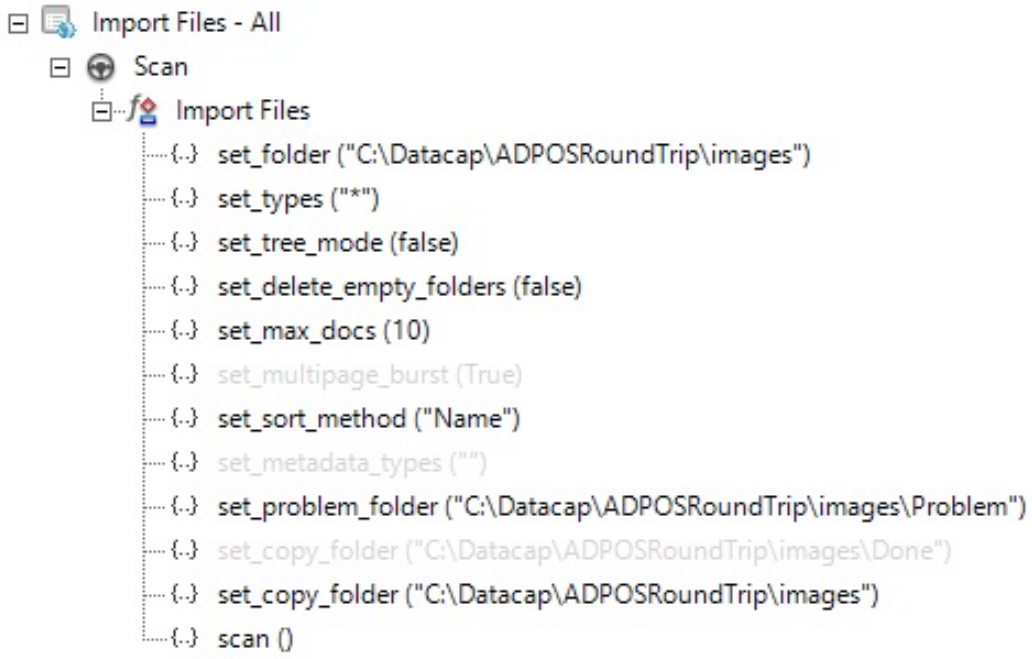
- The Post-processing step converts the multi-page TIFF file to a document with individual TIFF files as pages within the document. It then dynamically creates fields (with their values) to each page based on what ADP found for each page (see “Add ADP fields to individual pages”). It then moves all of the fields from the individual pages to the first page (see “Move ADP Fields To First Page”), with special properties set on the fields so that when the user tabs to the field in Content Navigator, the correct image will be displayed. Finally, validation is performed on the fields.



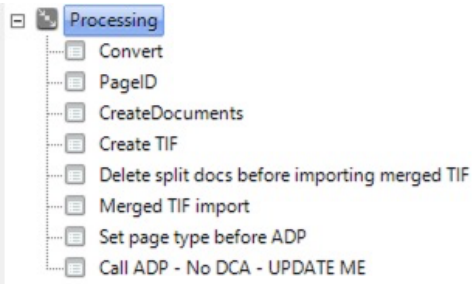
The following sections describe these steps in detail.

Scan (VScan or NScan)

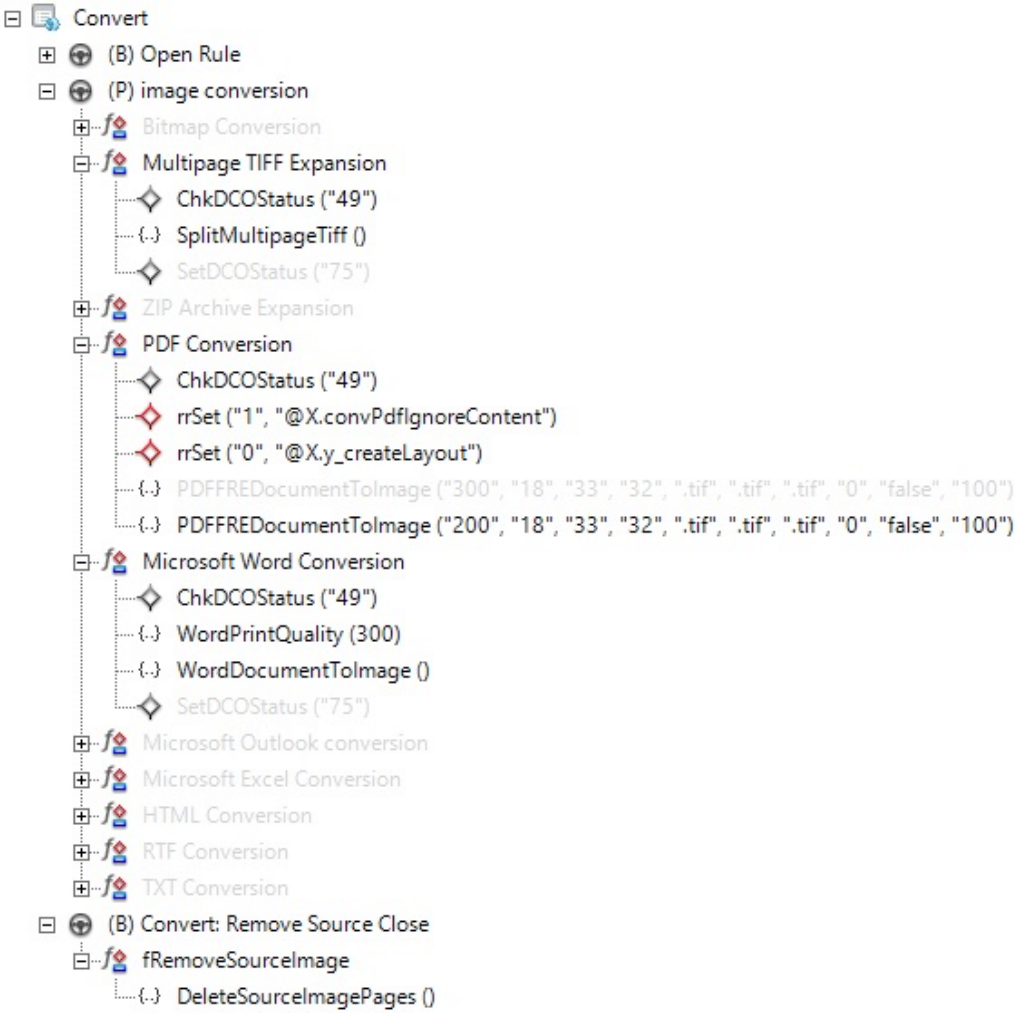
- Ingest files into Datacap. For demonstration purposes, these are assumed to be multi-page PDFs, multi-page TIFFs, or multi-page Word documents.



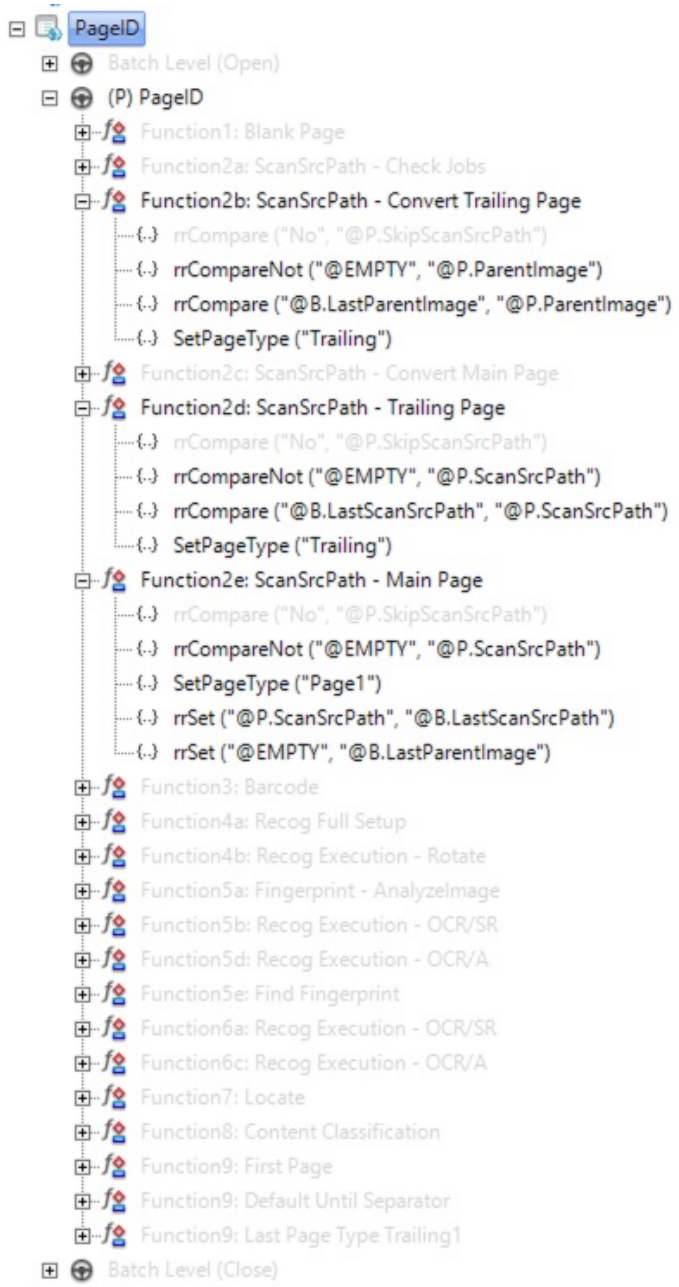
Processing



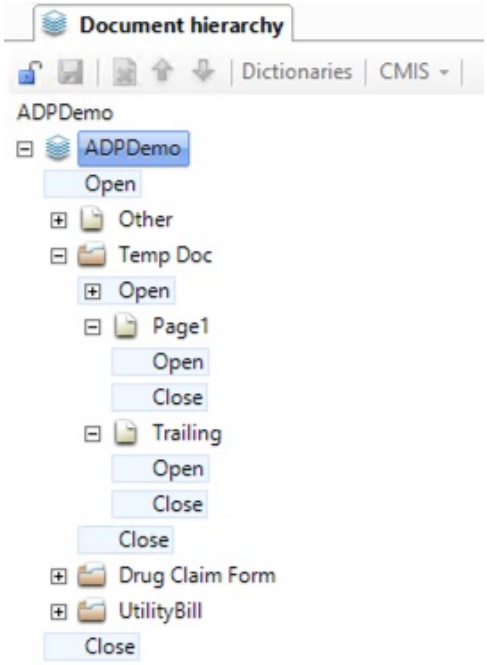
- Convert the input files into a consistent format:
  - Split the multi-page document into single-page TIFF images

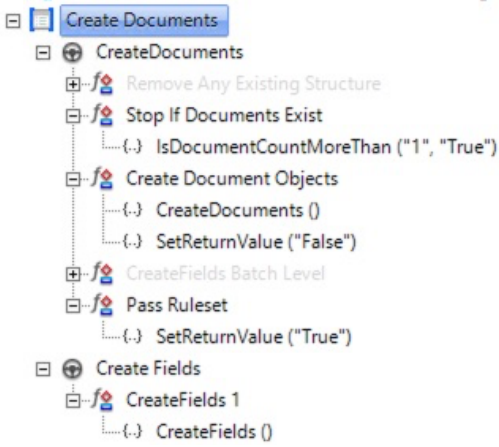


- For each image, set first pages to “Page1” and rest of pages to Trailing (yes, there’s a lot disabled in here that could be deleted…)



- Create Temp Docs from the pages

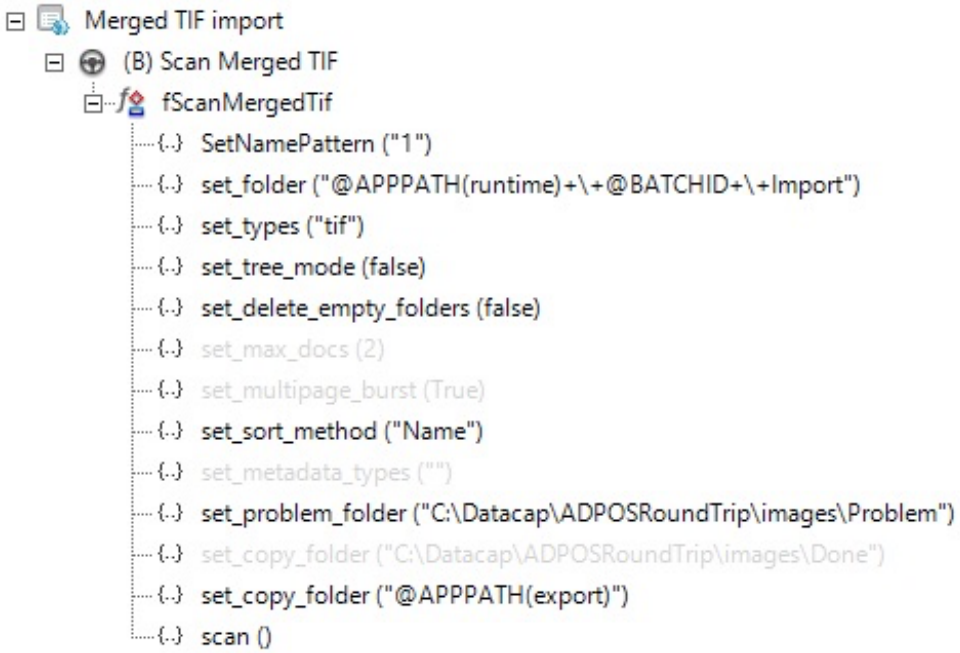




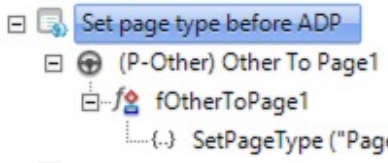
- Merge the individual pages into a multi-page TIFF

- Move original and split images to a subdirectory, then delete them from the batch directory so we can import the multi-page TIFFs we just created

- Add the multi-page TIFF files to the batch



- Set the page type of each page object



- Use the custom action SendPageToADP to send the multi-page TIFF(s) to ADP for processing.

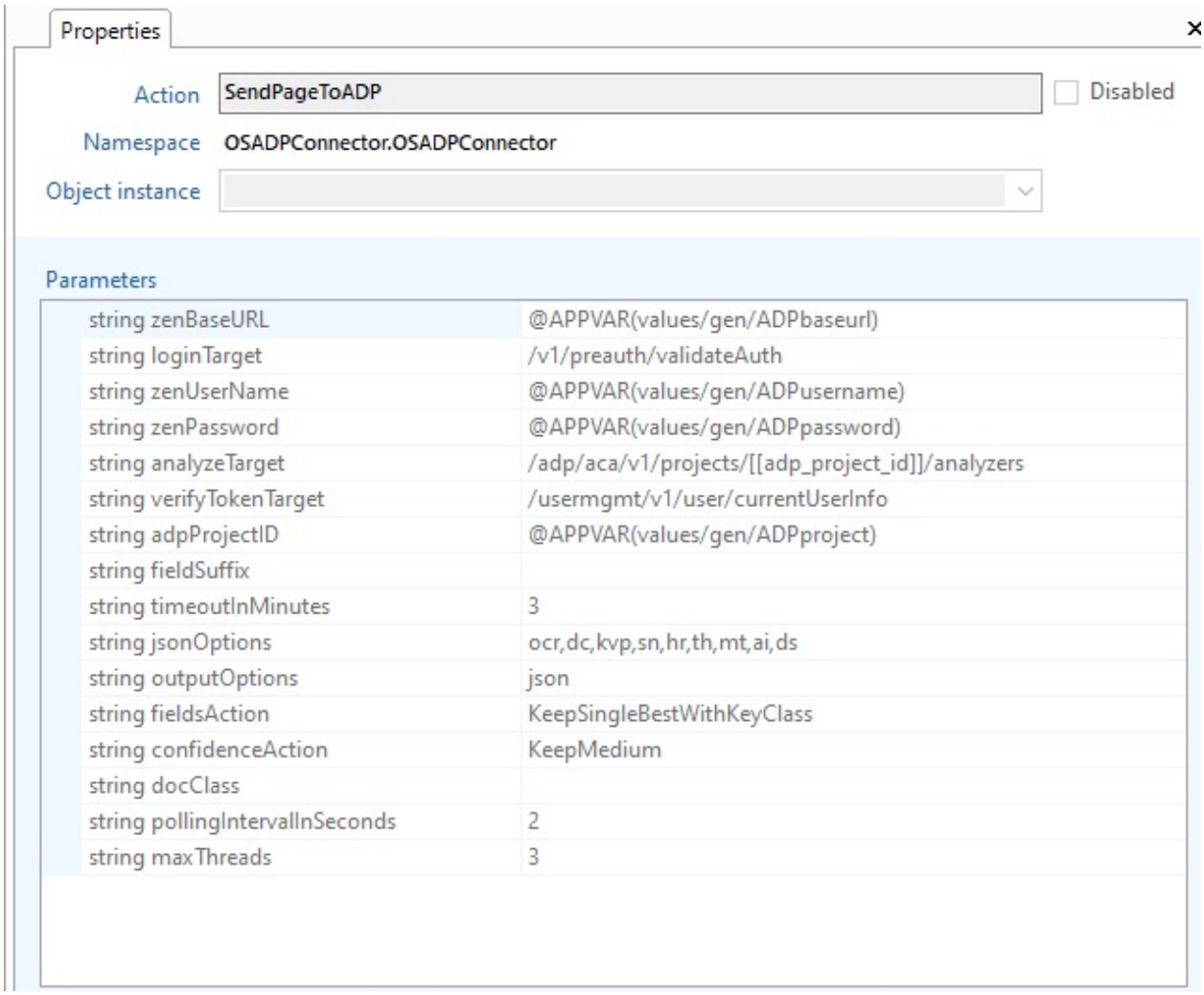
  - See the section below for a more detailed description of this ruleset and the SendPageToADP action.
  - If you'd like to send multiple documents in parallel to ADP, leave it as shown above.
  - If you'd like to ensure that only a single document is sent to ADP at a time, then disable the batch level function "UPDATE URL…" and enable the page level function "UPDATE URL…"

SendPageToADP action

The SendPageToADP action can be called at a batch level or a page level. If called at a batch level, it can run multi-threaded to send documents to ADP in parallel.

When the results come back, the ADP-provided information is written to a json file, the OCR results are used to create a layout xml file for the first page, the classification results are used to set the TYPE of the multi-page TIFF, and the key-value pairs ADP finds for the first page are added to the multi-page TIFF file.

Note that many of the parameters for SendPageToADP do not need to be changed. The parameters are described below:



NOTE FOR ADP EXTENSION: You will need to change the analyzeTarget parameter – see below.

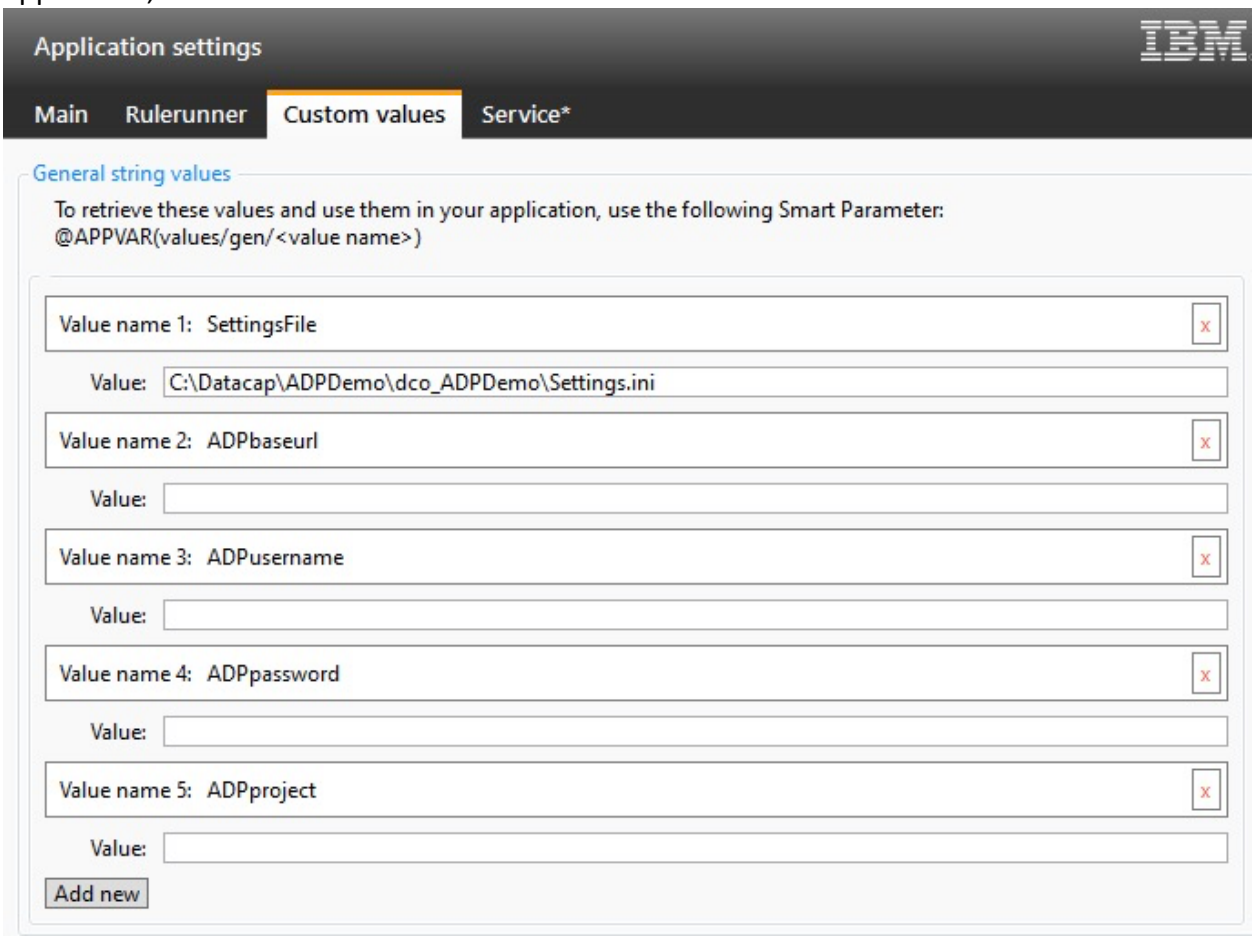
- zenBaseURL: This is the base URL (including port, if necessary) for accessing ADP via web service calls. Note that this URL has various other parameters appended to it for specific calls, such as loginTarget and analyzeTarget. It is important to ensure that this URL does NOT end with a forward slash (“/”), as the other parameters start with a forward slash
- loginTarget: This gets appended to the zenBaseURL for the login call. This should NOT be changed.
- zenUserName: The userid for logging in to ADP
- zenPassword: The password for logging in to ADP
- analyzeTarget: This gets appended to the zenBaseURL when sending documents to be processed by ADP. Note that the “[[adp_project_id]]” is automatically changed by the code to use the value of the adpProjectID parameter. This should NOT be changed. If you are using ADP Extension, however, you will need to remove “/adp/aca” from the beginning of the URL, leaving this parameter as: `/v1/projects/[[adp_project_id]]/analyzers`

- verifyTokenTarget: Currently unused.
- adpProjectID: The project ID associated with your ADP project. See the ADP documentation for how to find this (e.g., see the "Setting the Document Processing project ID" section in https://www.ibm.com/docs/en/cloud-paks/cp-biz-automation/22.0.2?topic=integrations-automation-document-processing-api). (Note: For ADP Extension, this is easy: Go to the project – the project ID is in the URL. For example: 2.118.165.188/?projectId=adp_base_sozxlk)
- fieldSuffix: Optional suffix to add to the end of fields created by ADP in case you need to distinguish them from fields created by Datacap.
- timeoutInMinutes: How many minutes should the code wait for a response from the ADP server.
- jsonOptions: Specifies what should be included in the json results from ADP. This should NOT be changed.
- outputOptions: Specifies that only json results should be returned from ADP. This should NOT be changed.
- fieldsAction: This specifies what to do when there are multiple fields with the same key class, or fields with no key class. Possible values are:
  - KeepAll: Don't remove any multiple fields.
  - KeepAllWithKeyClass: Keep all fields that have a key class from ADP.
  - KeepSingleBest: When there are multiple fields for an ADP key class, keep only the best one. Also keep fields without a key class.
  - KeepSingleBestWithKeyClass: When there are multiple fields for an ADP key class, keep only the best one. Don't keep fields without a key class.
- confidenceAction: This specifies how you want to handle fields based on how confident ADP is about the field. Possible values are:
  - KeepAll: Keep all fields, no matter how confident ADP is.
  - KeepMedium: Keep the fields ADP has high or medium confidence in.
  - KeepHigh: Keep only the fields ADP has high confidence in.
  - DeleteAll: Delete all fields from ADP
- docClass: This specifies the document class you want ADP to use for the document (if any). If this parameter is blank then ADP will determine the doc class.
- pollingIntervalInSeconds: This specifies the number of seconds the system should wait between asking ADP if it's done processing a page.
- maxThreads: This specifies the maximum number of threads to be used to send documents to ADP. This is only used if SendPageToADP is called from a batch level.

As shown in the screen shot above, the ADPDemo application uses custom values set in Datacap Application Manager for the main parameters that would need to be changed for your environment:

- zenBaseURL
- zenUserName
- zenPassword
- adpProjectID

These are found in the Custom Values tab of Datacap Application Manager for the ADPDemo application, as shown in the screen shot below.



Guidance for SendPageToADP

Although the four parameters mentioned above (the URL, credentials and project ID) are the only parameters you must set, it may be helpful to adjust some of the other parameters depending on your documents.

For a simple example, if you know that you will only be sending a single document type to ADP, then set the docClass parameter to the name of that document type in your ADP project. Doing so will save time in processing and will avoid any potential (though rare) mis-classifications in ADP.

Document classification considerations

When processing a document, ADP determines which document type is the best match. But ADP will evaluate other document types and return information about how confident it is that the document matches those other document types.

For example, in the screen shot below, the document type that was selected was UtilityBill – thus SendPageToADP set the page property ADPDocType as UtilityBill so that a later action could use that property to set the Datacap TYPE property.

But SendPageToADP also indicates that there were three document types evaluated: UtilityBill, Invoice, and BillOfLading. ADP's confidence that the document type is UtilityBill is high, and it's confidence that the document type is Invoice or BillOfLading is low.

ADPDocType	UtilityBill
ADPDocTypeConfidence	High
ADPDocType_0	UtilityBill
ADPDocType_0Confidence	High
ADPDocType_1	Invoice
ADPDocType_1Confidence	Low
ADPDocType_2	BillOfLading
ADPDocType_2Confidence	Low

In such a case, it may be appropriate to take ADP's selection and use it for the document type.

But in the example below, another document was processed with low confidence in all of the document types. The document was considered to be an Invoice for the purposes of field extraction, but since ADP had low confidence that it was an Invoice, the Datacap application might have been justified in ignoring the document classification and performed its own classification.

ADPDocType	Invoice
ADPDocTypeConfidence	Low
ADPDocType_0	Invoice
ADPDocType_0Confidence	Low
ADPDocType_1	UtilityBill
ADPDocType_1Confidence	Low
ADPDocType_2	BillOfLading
ADPDocType_2Confidence	Low

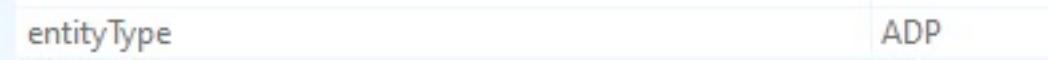
Note that in such a situation, the fields extracted by ADP could still be used by Datacap – but they would need to be used with care. (See below for more information on fields.)

Keeping ADP fields distinct from your own Datacap fields

The fieldSuffix parameter can be used to distinguish fields created from ADP vs fields created by other Datacap actions.

For example, you can use the fieldSuffix parameter to specify that the ADP-provided fields will all end with a specific string, say "_ADP". You can then use a tried-and-true set of actions to extract, say, a postal code from the document into your field "PostalCode" and build actions that compare the value of that field to the value that ADP provides in "PostalCode_ADP". If there's a match, you can mark the field as having high confidence, but you may want to mark it with lower confidence if there's a discrepancy. (Also see the section below about ADP's confidence in what it provides.)

Also note that you can tell that a field was created from ADP when the field has an entityType property with the value ADP.



Which fields should I use?

When you configure a document type in ADP, you can also specify which fields it should look for and extract.

You may not realize this, but when ADP processes a document, it typically does not restrict itself to just providing a single value for just the fields you've configured. Indeed, it may find multiple potential instances of the configured fields, and it may find potential fields that have not been configured.

All of these possible field values are returned from ADP, (along with OCR results, possible document classes, and more). And all of these possible field values are available to you when configuring the Datacap-ADP integration using this connector.

The parameters which control how those fields are used are the fieldsAction and confidenceAction parameters to SendPageToADP and SpreadTheADPResults. (Be sure to specify the same values for both actions for the best consistency.)

The fieldsAction parameter specifies what the connector should do when there are multiple fields returned from ADP that all have the same key class, as well as what the connector should do with fields that aren't associated with a key class.

Possible values for fieldsAction are:

- **KeepAll**
  Don't remove any multiple fields.
- **KeepAllWithKeyClass**
  Keep all fields that have a key class from ADP.
- **KeepSingleBest**
  When there are multiple fields for an ADP key class, keep only the best one. Also keep fields without a key class.

- **KeepSingleBestWithKeyClass**
  When there are multiple fields for an ADP key class, keep only the best one. Don't keep fields without a key class.

The confidenceAction parameter specifies how the connector should handle fields based on how confident ADP is about the field.

Possible values for confidenceAction are:

- **KeepAll**
  Keep all fields, no matter how confident ADP is.
- **KeepMedium**
  Keep the fields ADP has high or medium confidence in.
- **KeepHigh**
  Keep only the fields ADP has high confidence in.
- **DeleteAll**
  Delete all fields from ADP

Note that the fields that are placed on the page by the connector are the ones that satisfy what is specified for ***both*** of these parameters. So, for example, if you specify

- fieldsAction = KeepSingleBest
- confidenceAction = KeepHigh

and the best field ADP finds for a particular key class only has medium confidence, then that field will not be placed on the page.

Finally, note that the fields from ADP that are created by the connector have a confidence property, an ADPKeyClassName property, and an ADPKeyClassConfidence property that can be used in your actions to make your own decisions about how to handle them.

confidence	100
validityPercentage	100
label	AccountNumber
subMatch1	0068411356
ADPKeyClassName	AccountNumber
ADPKeyClassConfidence	High

Configuring for throughput

Generally speaking, getting the best throughput out of any system is more of an art than a science. Given that disclaimer, there are a few parameters to SendPageToADP that can affect your overall throughput positively or negatively.

These parameters are:

- Whether the action is called from a batch, document or page level
- timeoutInMinutes
- pollingIntervalInSeconds
- maxThreads

If SendPageToADP is called at a batch or document level, the action will spawn up to maxThreads to send all of the objects in that batch or document to ADP in parallel. While the object is processing in ADP, the thread will wait for pollingIntervalInSeconds seconds between checks to see if ADP has completed its work on that object. If the thread has waited more than timeoutInMinutes minutes, the connector deems that object as a failure and kills the thread that was processing that object. As each object completes, a new object is sent until all of the objects are processed. If there are any errors (including timeouts), the connector will return false so appropriate action can be taken by the application.

If SendPageToADP is called at a page level, the action will send each individual page object to ADP for processing as RuleRunner processes each page. This effectively means that each page object is processed serially. While each object is processing in ADP, the connector will wait for pollingIntervalInSeconds seconds between checks to see if ADP has completed its work on that object. If the connector has waited more than timeoutInMinutes minutes, the connector deems that object as a failure and returns false.

> Note in the above descriptions, we used the term “object” or “page object” rather than “page” or “document” or something else. This is because ADP can process a single or multi-page object, but it does assume that if the image or pdf file is a multi-page object, then all pages of that object belong together and are the same document type. ***This is a very important thing for you to consider as you evaluate how best to use ADP for your documents.***

We’ve described how the connector uses those parameters (called from batch, doc or page; timeout; polling interval; number of threads), but there are also several other factors to consider as you determine how to set these parameters to get the best throughput for your documents. For example:

- Your ADP cluster must be sized appropriately, and you need to carefully configure the number of RuleRunner servers and the number of sub-threads to process each batch (maxThreads above). If you’re processing a great many batches in Datacap, using a great many RuleRunner processes, and using many sub-threads to process each batch, then you will be dumping many documents in parallel into ADP. You will need to ensure your ADP cluster can keep up with the rate of documents being sent to it.
- The Datacap-ADP connector will hold a RuleRunner thread while ADP is processing the documents, locking out that RuleRunner thread from processing other batches. Because of this behavior, if you are using multiple Datacap applications, you may want to configure the application using the Datacap-ADP connector to only run on a subset of

your RuleRunner servers. You may also want to configure more RuleRunner threads than you otherwise would in order to keep RuleRunner processing at an appropriate level.

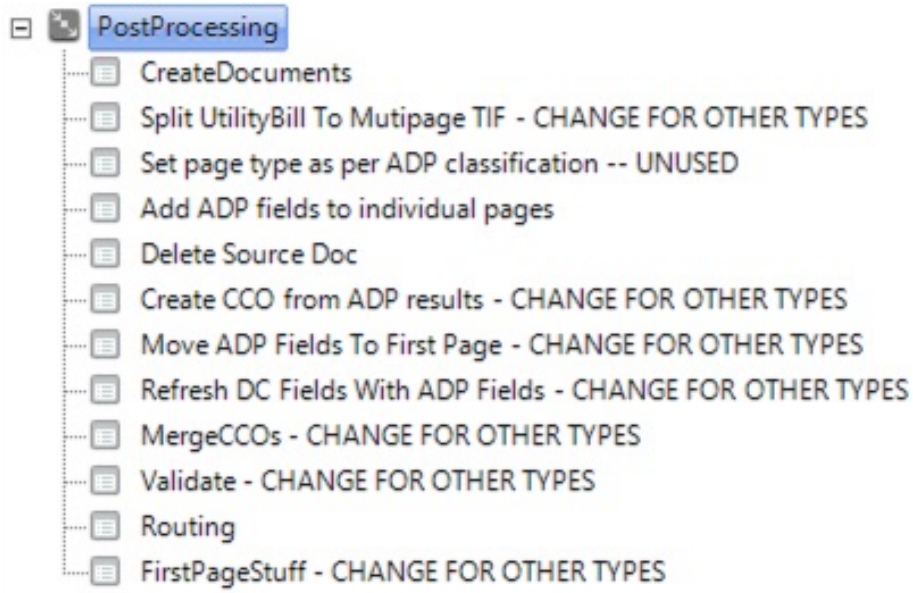
- The size of your documents may affect the timeout value – if you're sending huge documents to ADP, you may need a larger value for timeoutInMinutes.

Handling sensitive data

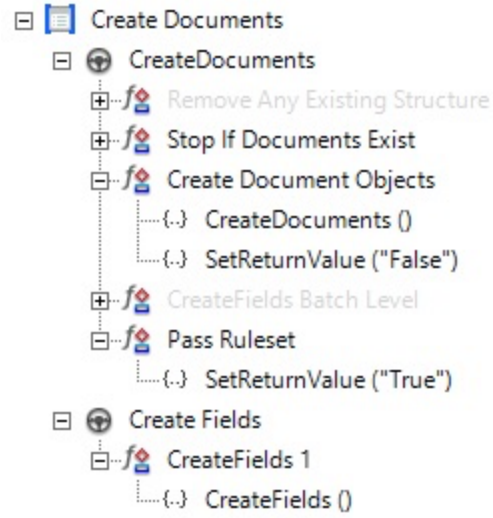
When you configure fields in ADP, you can mark them as Sensitive. When the Datacap-ADP connector creates the fields on the page, it sets a field-level property ADPSensitivity to True or False based on whether the field was marked as sensitive in ADP.

You can examine this value and use it to optionally redact the information.

PostProcessing



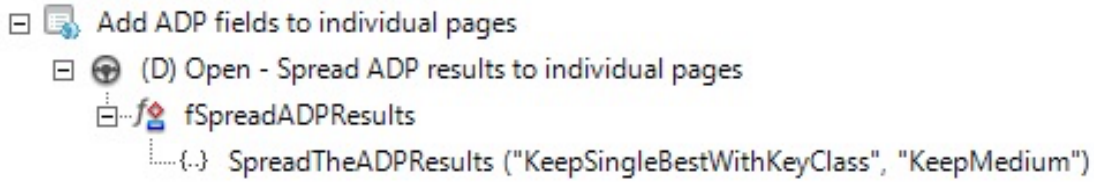
- Create documents using the multi-page TIFFs. This creates a document for each multi-page TIFF in the batch; the contents of which are the multi-page TIFF.



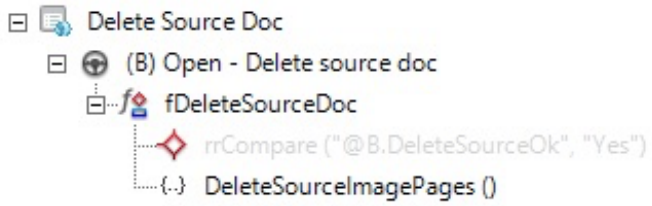
- For multi-page TIFFs that were identified as UtilityBill, split the file into individual single-page TIFFs. This is in ruleset “Split UtilityBill To Mutipage TIF - CHANGE FOR OTHER TYPES”; this should be changed to support other document types. This results in the document containing the multi-page TIFF and individual single-page TIFFs.

- Use the custom action SpreadTheADPResults to add the ADP results to the individual single-page TIFFs. Note that this creates a layout xml file for each page, creates the ADP fields on each page, and sets many properties for the pages and the fields.

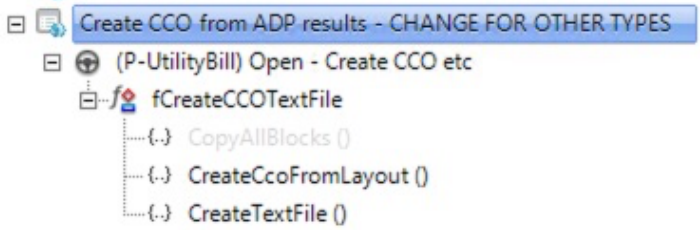
  Important note: See the discussion above regarding the fieldsAction and confidenceAction parameters.



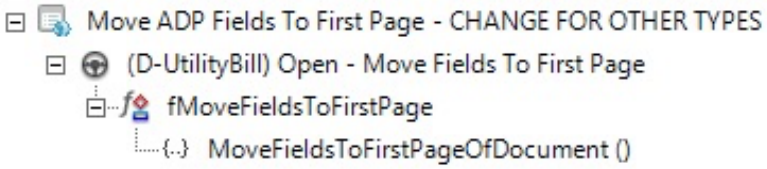
- Delete the multi-page TIFF from the document.



- Create the CCO from the ADP results – this is associated with page type UtilityBill; you will need to adjust for other types



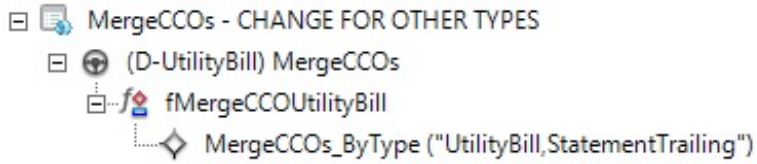
- Use the custom action MoveFieldsToFirstPageOfDocument. This action merges the fields from the individual pages to the first page of the document, sets the IsFirstPage property on each page (1 for the first page, 0 for following pages), and adjusts the properties so that when the document is viewed in Verify the image being viewed will automatically shift to the correct page. This is in ruleset “Move ADP Fields To First Page - CHANGE FOR OTHER TYPES”; this should be changed to support other document types.



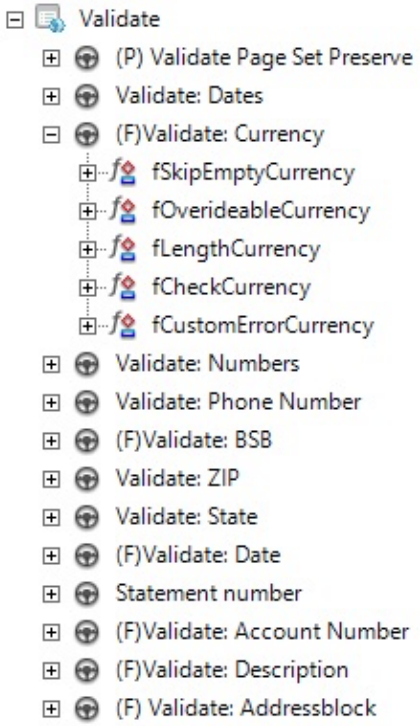
- Make sure all static DCO fields are present (keep the ADP fields intact). This is in ruleset “Refresh DC Fields With ADP Fields - CHANGE FOR OTHER TYPES”; this should be

changed to support other document types.

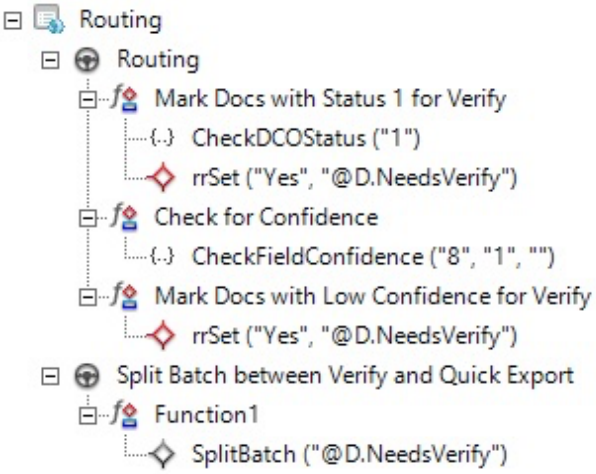
- Merge the UtilityBill pages into a single CCO. This is in ruleset “MergeCCOs - CHANGE FOR OTHER TYPES”; this should be changed to support other document types.



- Perform various types of validation – this should be enhanced per your application needs.



- Check confidence levels and route appropriately



- Set the first page status. This is in ruleset “FirstPageStuff - CHANGE FOR OTHER TYPES”; this should be changed to support other document types.