

## 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 screenshot displays the ADPDemo application interface, which is divided into three main sections: a document preview on the left, a 'Field Details' section in the center, and a 'Batch Structure' table on the right.

**Document Preview (Left):** Shows a utility bill for HughesNet Cable. The bill includes the account number 0068411356, the customer name Brenda J Briggs, and the service address 3009 Luke Lane, Oklahoma. The bill period is from February 1 to March 1, 2015. The bill due date is March 25, 2015. The bill shows a previous balance of \$34.67, a payment received of \$34.67, and a total amount due of \$55.18. The bill also includes a 'Bill Saving Tips' section and a 'Ways to Pay' section.

**Field Details (Center):** This section displays the extracted fields from the document. The fields are organized into a form-like structure with labels and values. The fields include: AccountNumber (0068411356), CustomerName (Brenda J Briggs), ServiceAddress (3009 Luke Lane), DueDate (March 25 2015), PreviousBalance (\$34.67), PaymentReceived (34.67), TotalAmountDue (\$55.18), StatementDate (Internet speed), and UtilityCompanyName (Internet speed).

**Batch Structure (Right):** This section displays a table of the batch structure. The table has three columns: ID, Type, and Status. The table contains 16 rows of data, showing the results of the document processing. The table is as follows:

| ID                 | Type              | Status  |
|--------------------|-------------------|---------|
| 20230526.000005    | ADPDemo           | OK      |
| 20230526.000005.01 | UtilityBill       | Problem |
| 01030000           | StatementTrailing | Scanned |
| 20230526.000005.02 | UtilityBill       | OK      |
| 02030000           | StatementTrailing | Scanned |
| 20230526.000005.03 | UtilityBill       | OK      |
| 03030000           | StatementTrailing | Scanned |
| 20230526.000005.04 | UtilityBill       | OK      |
| 04030000           | StatementTrailing | Scanned |
| 20230526.000005.05 | UtilityBill       | OK      |
| 05030000           | StatementTrailing | Scanned |
| 20230526.000005.06 | UtilityBill       | OK      |
| 06030000           | StatementTrailing | Scanned |
| 20230526.000005.07 | UtilityBill       | OK      |
| 07030000           | StatementTrailing | Scanned |
| 20230526.000005.08 | UtilityBill       | OK      |
| 08030000           | StatementTrailing | Scanned |

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

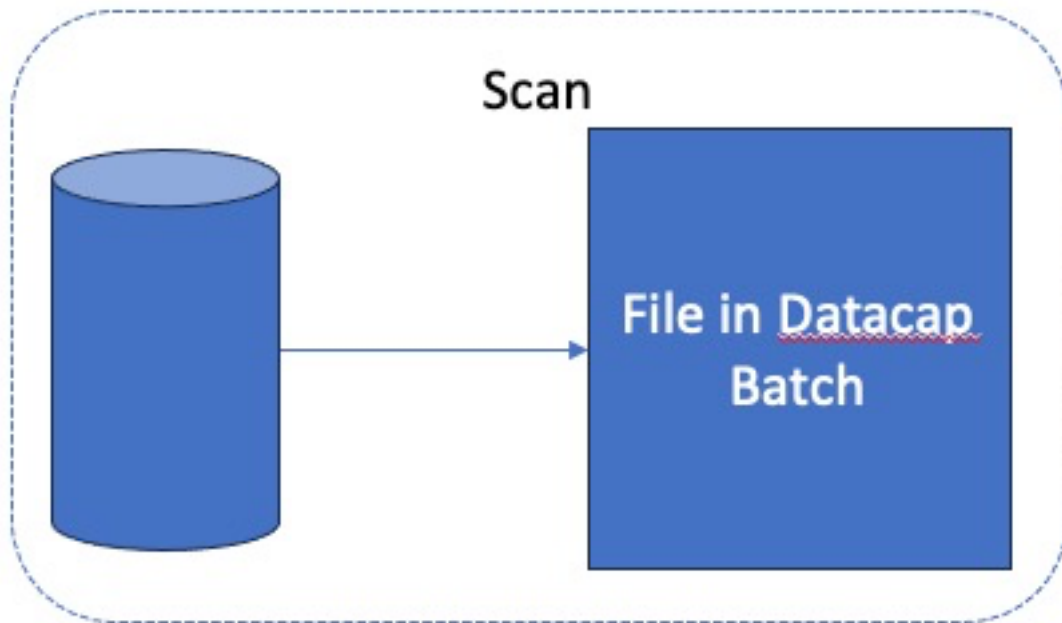
Scan, Processing and Post-Processing are described in more detail in the following sections, but it will be helpful to have an overview of the processing before getting into the details.

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

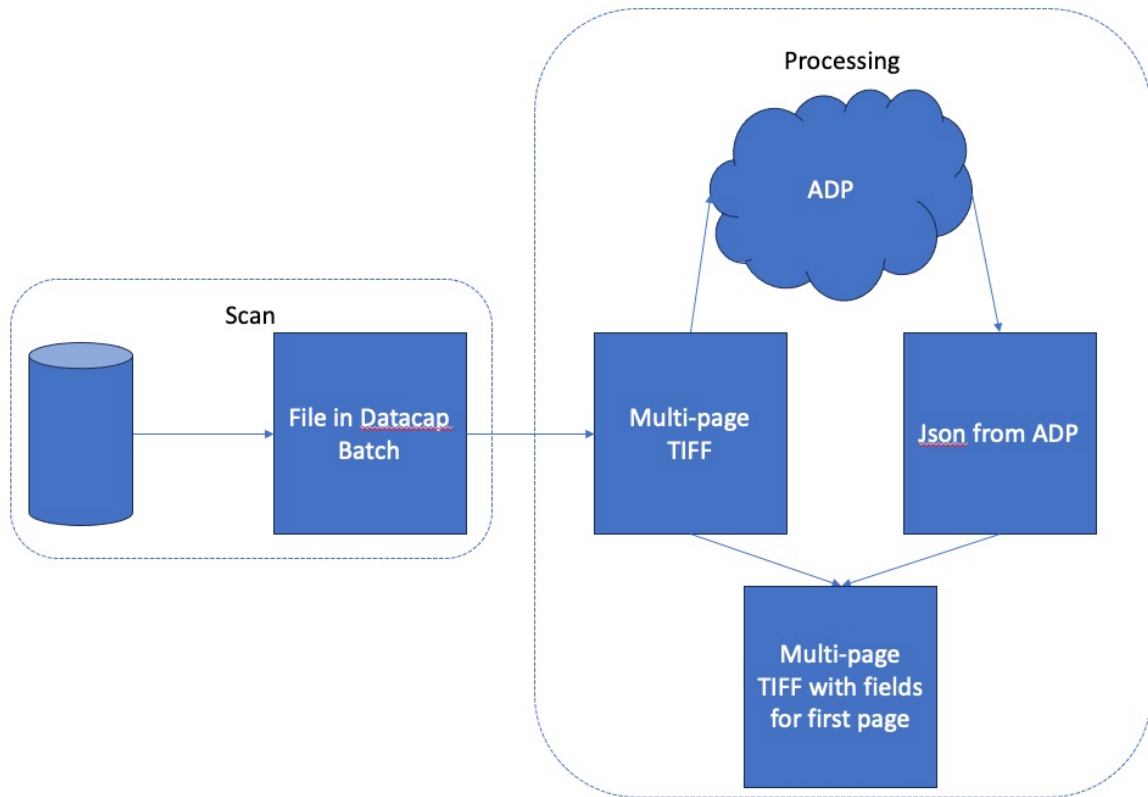
associated with the document. Furthermore, when the results are finally displayed to the user for verification, it is helpful to have all of the fields shown together, rather than shown page by page.

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

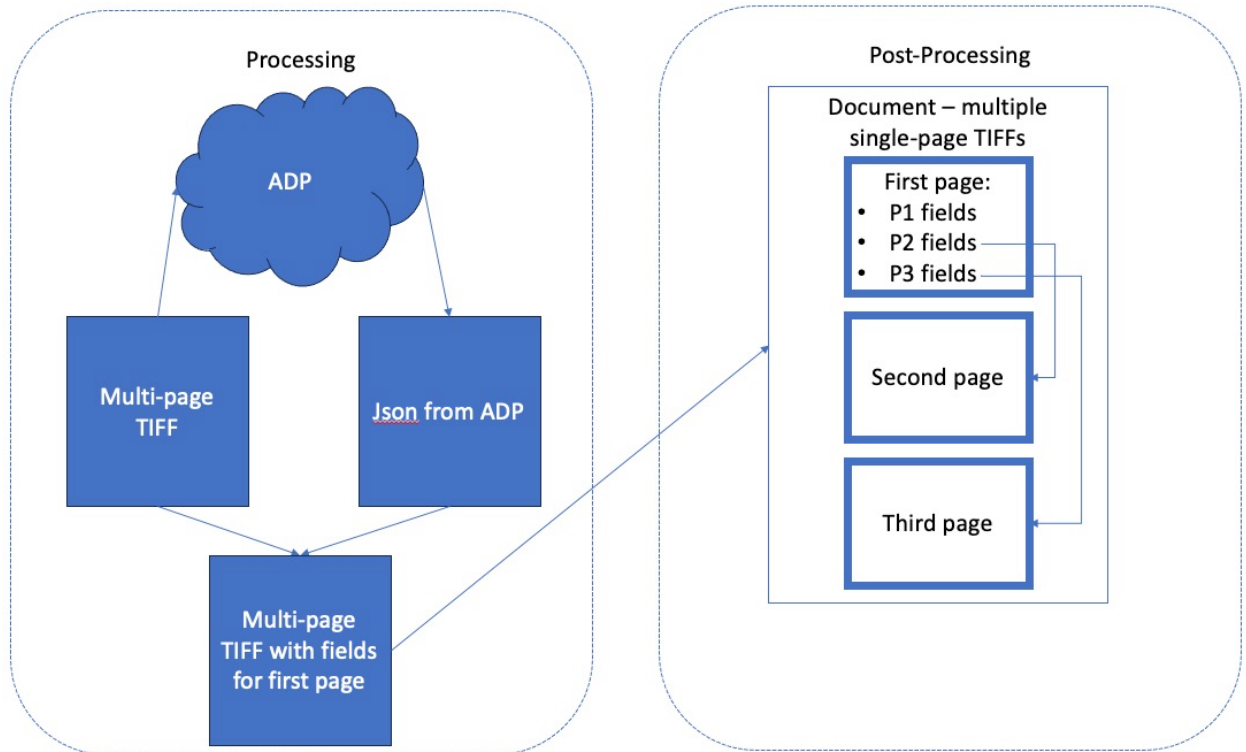
- The Scan step ingests the files into Datacap. The assumption for the sample application is that each input file is a single document – whether a single-page file or a multi-page file.



- 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.

Import Files - All

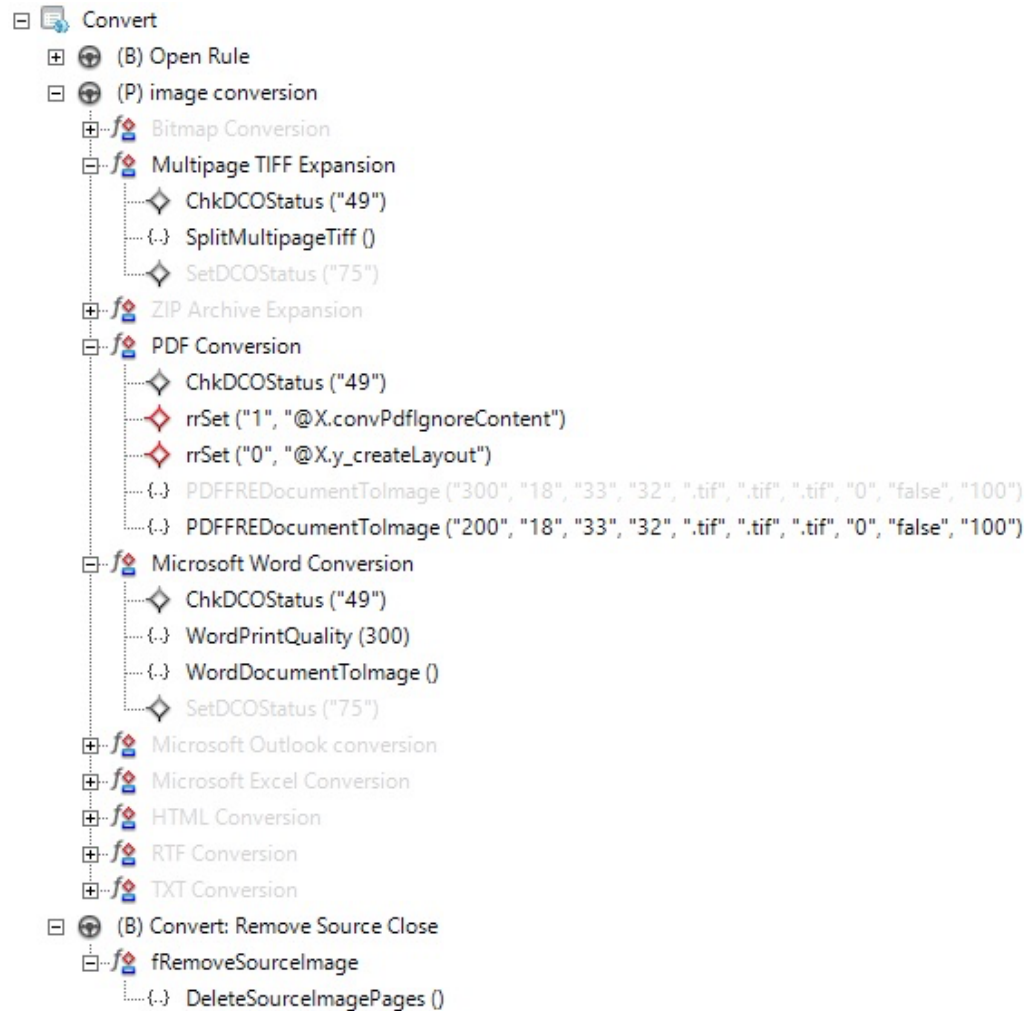
Scan

Import Files

```
{-} set_folder ("C:\Datacap\ADPOSRoundTrip\images")
{-} set_types ("")
{-} set_tree_mode (false)
{-} set_delete_empty_folders (false)
{-} set_max_docs (10)
{-} set_multipage_burst (True)
{-} set_sort_method ("Name")
{-} set_metadata_types ("")
{-} set_problem_folder ("C:\Datacap\ADPOSRoundTrip\images\Problem")
{-} set_copy_folder ("C:\Datacap\ADPOSRoundTrip\images\Done")
{-} set_copy_folder ("C:\Datacap\ADPOSRoundTrip\images")
{-} scan ()
```

## Processing

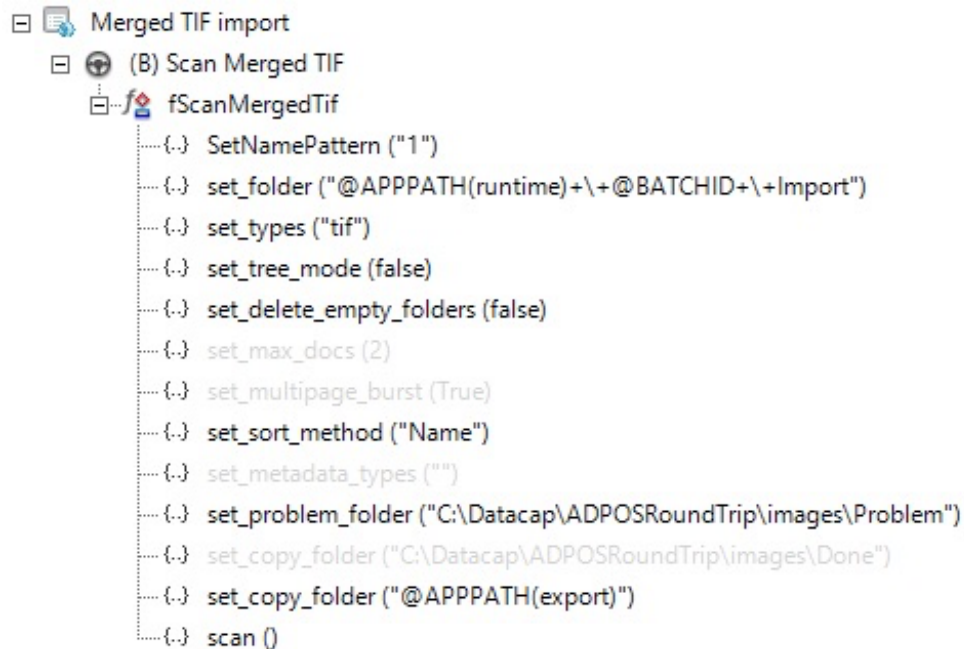
- Convert the input files into a consistent format:
  - Split the multi-page document into single-page TIFF images



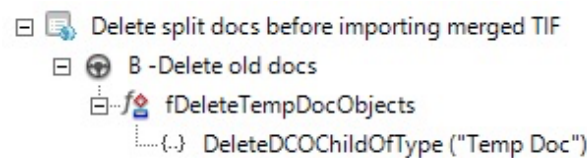
- Merge the individual pages into a multi-page TIFF



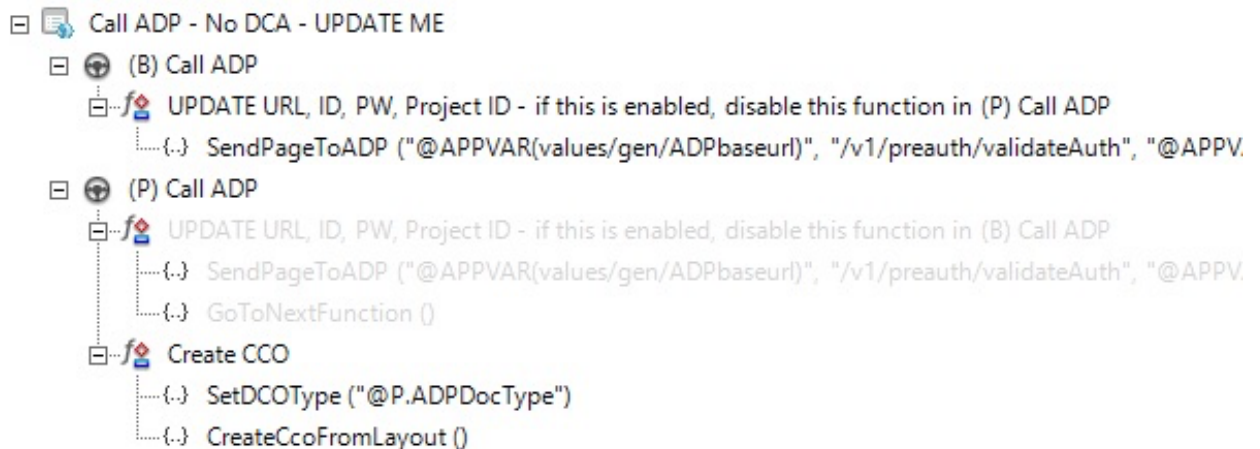
- Add the multi-page TIFF files to the batch



- Delete the temporary files



- 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..."

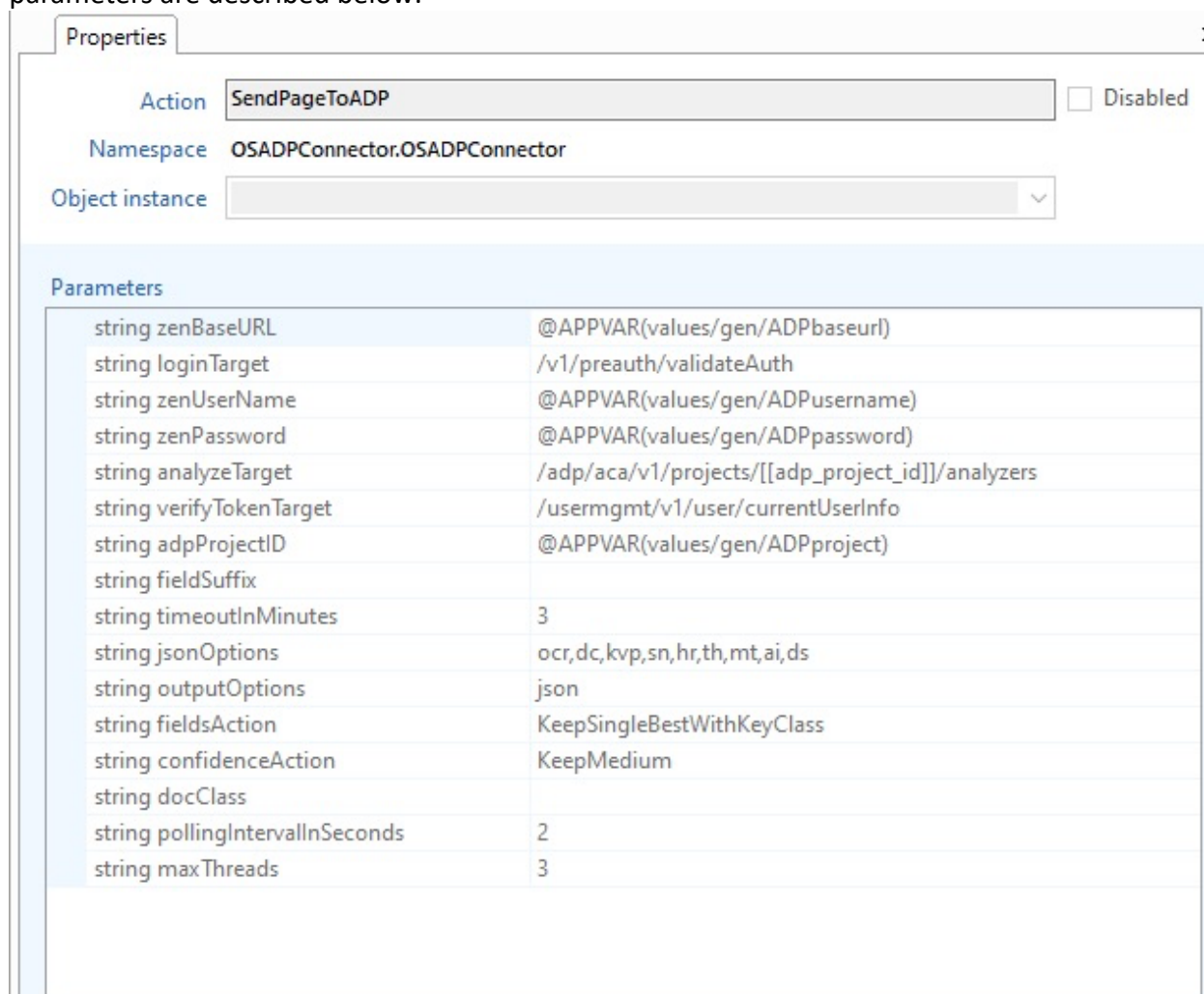


## SendPageToADP action

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:



| Parameters                      |                                                   |
|---------------------------------|---------------------------------------------------|
| string zenBaseURL               | @APPVAR(values/gen/ADPbaseurl)                    |
| string loginTarget              | /v1/preauth/validateAuth                          |
| string zenUserName              | @APPVAR(values/gen/ADPusername)                   |
| string zenPassword              | @APPVAR(values/gen/ADPpassword)                   |
| string analyzeTarget            | /adp/aca/v1/projects/[[adp_project_id]]/analyzers |
| string verifyTokenTarget        | /usermgmt/v1/user/currentUserInfo                 |
| string adpProjectID             | @APPVAR(values/gen/ADPproject)                    |
| string fieldSuffix              |                                                   |
| string timeoutInMinutes         | 3                                                 |
| string jsonOptions              | ocr,dc,kvp,sn,hr,th,mt,ai,ds                      |
| string outputOptions            | json                                              |
| string fieldsAction             | KeepSingleBestWithKeyClass                        |
| string confidenceAction         | KeepMedium                                        |
| string docClass                 |                                                   |
| string pollingIntervalInSeconds | 2                                                 |
| string maxThreads               | 3                                                 |

- 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.
- 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>).
- 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

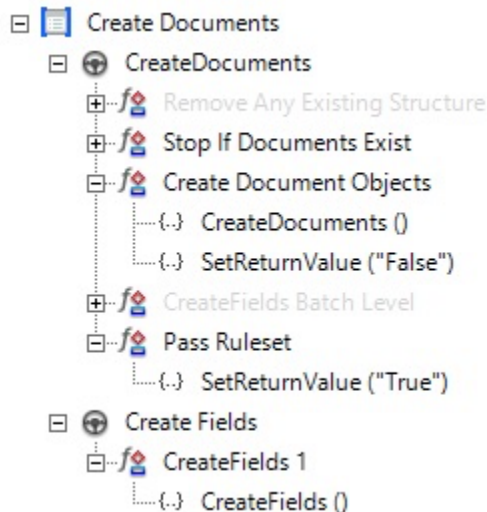
These are found in the Custom Values tab of Datacap Application Manager for the ADPDemo application, as shown in the screen shot below.

The screenshot shows the 'Application settings' window for the ADPDemo application. The 'Custom values' tab is selected, displaying a list of custom string values. The IBM logo is visible in the top right corner. Below the tab headers, a section titled 'General string values' provides instructions on how to retrieve these values using a Smart Parameter: `@APPVAR(values/gen/<value name>)`. The list contains five entries, each with a 'Value name' and a 'Value' field. The first entry, 'SettingsFile', has a value of 'C:\Datacap\ADPDemo\dco\_ADPDemo\Settings.ini'. The other four entries ('ADPbaseurl', 'ADPusername', 'ADPpassword', and 'ADPproject') have empty value fields. Each entry has a red 'X' button to its right. An 'Add new' button is located at the bottom left of the list.

| Value name                 | Value                                       |
|----------------------------|---------------------------------------------|
| Value name 1: SettingsFile | C:\Datacap\ADPDemo\dco_ADPDemo\Settings.ini |
| Value name 2: ADPbaseurl   |                                             |
| Value name 3: ADPusername  |                                             |
| Value name 4: ADPpassword  |                                             |
| Value name 5: ADPproject   |                                             |

## PostProcessing

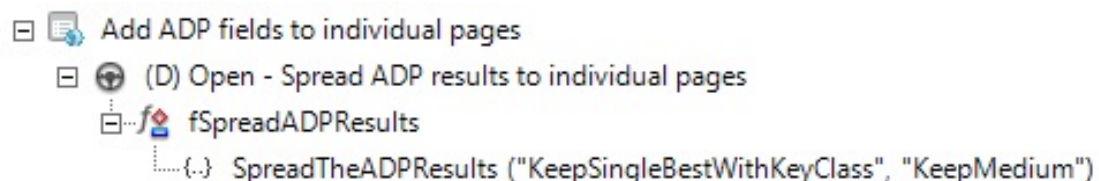
- Create documents using the multi-page TIFFs. This creates a document for each multi-page TIFF in the batch; the contents of which are the multi-page TIFF.



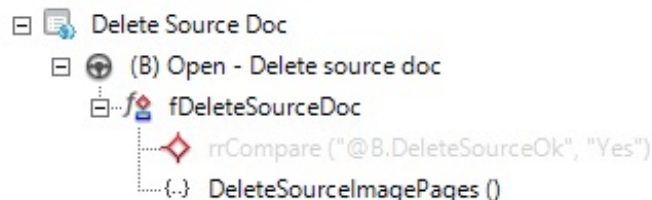
- For multi-page TIFFs that were identified as UtilityBill, split the file into individual single-page TIFFs. This is in ruleset “Split UtilityBill To Mutipage TIF - CHANGE FOR OTHER TYPES”; this should be changed to support other document types. This results in the document containing the multi-page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



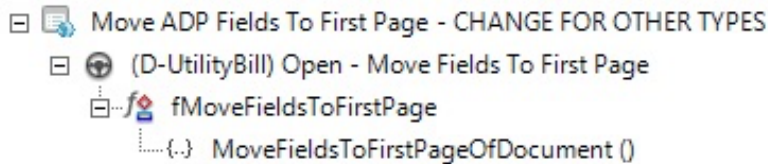
- Use the custom action SpreadTheADPResults to add the ADP results to the individual single-page TIFFs. Note that this creates a layout xml file for each page, creates the ADP fields on each page, and sets many properties for the pages and the fields.



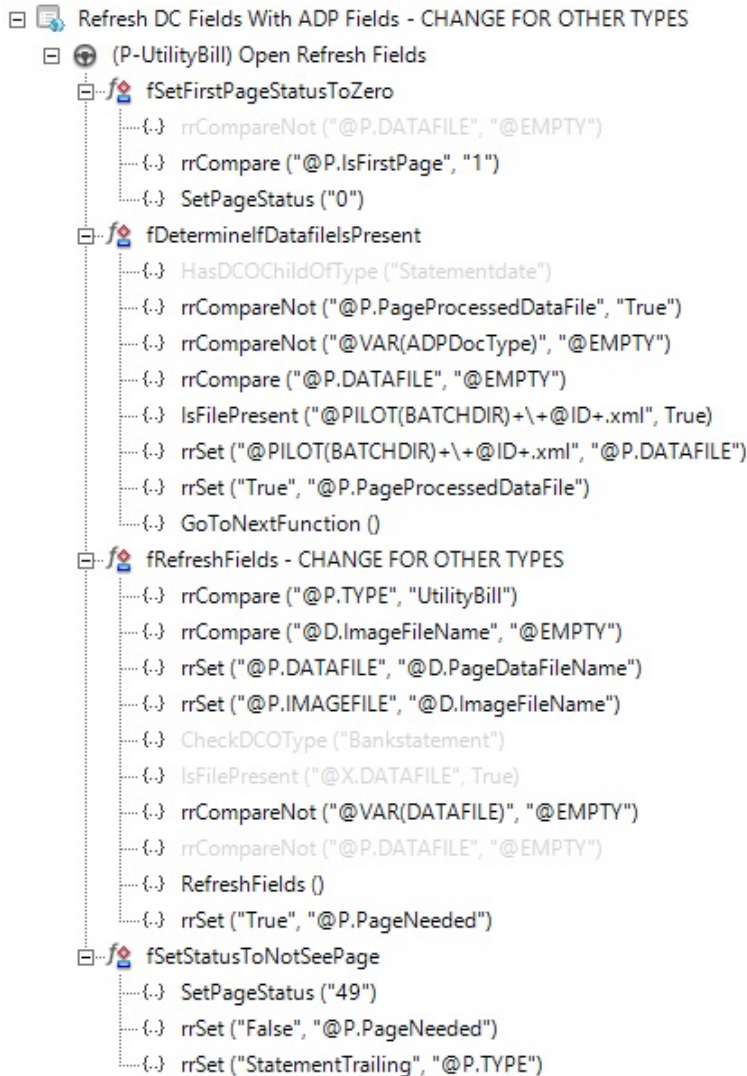
- Delete the multi-page TIFF from the document.



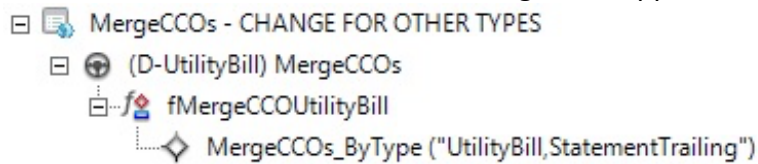
- Use the custom action MoveFieldsToFirstPageOfDocument. This action merges the fields from the individual pages to the first page of the document, sets the IsFirstPage property on each page (1 for the first page, 0 for following pages), and adjusts the properties so that when the document is viewed 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.



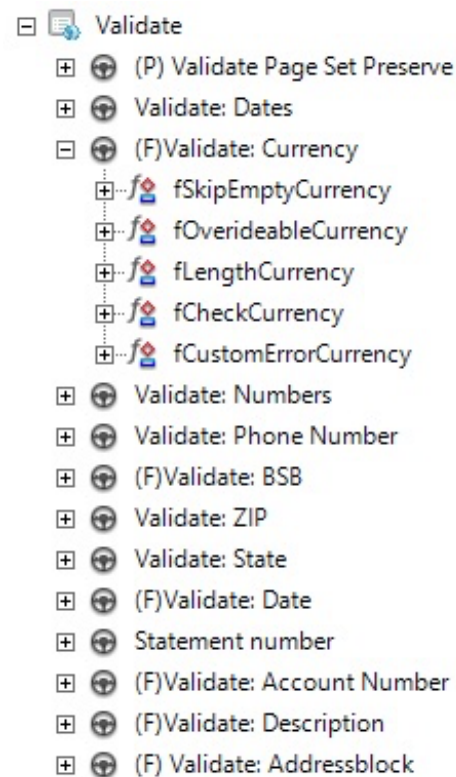
- Make sure all static DCO fields are present (keep the ADP fields intact). This is in ruleset "Refresh DC Fields With ADP Fields - CHANGE FOR OTHER TYPES"; this should be changed to support other document types.



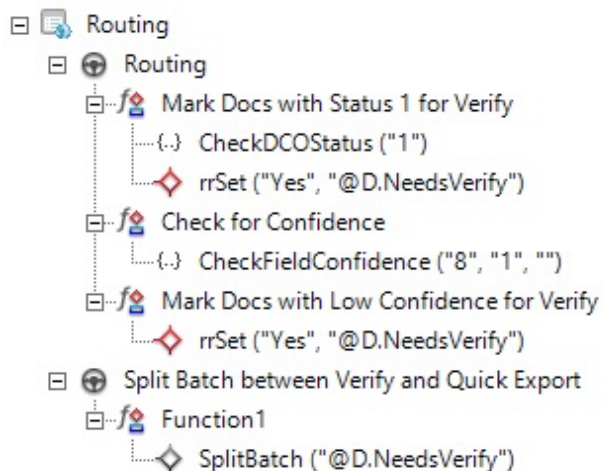
- Merge the UtilityBill pages into a single CCO. This is in ruleset “MergeCCOs - CHANGE FOR OTHER TYPES”; this should be changed to support other document types.



- 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.

