opentext™

OpenText™ Intelligent Capture

Image Handling Modules Guide

This guide contains information about the Intelligent Capture image handling client modules.

ECPCORE220300-CIH-EN-1

OpenText™ Intelligent Capture
Image Handling Modules Guide
ECPCORE220300-CIH-EN-1
Rev.: 2022-June-13

This documentation has been created for OpenText™ Intelligent Capture CE 22.3.
It is also valid for subsequent software releases unless OpenText has made newer documentation available with the product, on an OpenText website, or by any other means.

Open Text Corporation

275 Frank Tompa Drive, Waterloo, Ontario, Canada, N2L 0A1

Tel: +1-519-888-7111
Toll Free Canada/USA: 1-800-499-6544 International: +800-4996-5440
Fax: +1-519-888-0677
Support: https://support.opentext.com
For more information, visit https://www.opentext.com

Table of Contents

Chapter 1

Image Handling Modules

Starting in 22.3, this guide contains a compilation of the Intelligent Capture image handling client modules. In previous releases, these were available as separate guides.

For information about common features and functionalities in the Intelligent Capture client modules, see *OpenText Intelligent Capture - Module Reference (ECPCORE-CMD)*.

For information about Intelligent Capture image handling and operating specifications, see *OpenText Intelligent Capture - Operating Specifications (ECPCORE-RLI)*.

Table 1-1: Image Handling Modules

Module	Original module ID	Link
Image Converter	ecpcore-cic	See "Image Converter Module"
Image Processor	ecpcore-cip	See "Image Processor Module"

Chapter 2

Image Converter Module

This section includes the Intelligent Capture Image Handling module: **ecpcore-cic**.

Image conversion is a mechanism that lets you convert files from one format to another and transform files from one type to another. This conversion is done by the Image Converter module.

The Image Converter module implements the following features:

Performs file conversion

Depending on the Image Conversion profile defined for a step, conversion can include:

- changing image properties including file format, color format, and compression;
- converting electronic textual documents to *PDF* and *PDF/A* formats;
- converting Microsoft Office, Text, and *HTML* documents to image or PDF formats without having to install Microsoft Office;
- opportunity to specify additional options when generating output files of such types as PDF, *TIFF*, or *BMP*;
- merging single-page files to multi-page documents and splitting multi-page documents into single pages;
- merging annotations added to TIFF images by other modules, such as Image Processor or Collection, into the output image;
- merging and splitting text-based PDF and PDF/A documents;
- processing PDF files that contain annotations and attachments;
- extracting PDF attachments and saving them to IA values, that is, PDF input file nodes.

Supports processing of image and non-image files

- Supports converting a wide range of image formats, Microsoft Office documents, PDF, and HTML files.
- Auto detects the type of content for the output PDF page based upon the following types: text, image, mixed content, or unknown type (e.g. blank page).

Creates thumbnails for all pages with the single-page output type
Generates thumbnails of the pages processed.

2.1 Setting Up Image Converter

Most configuration information for Intelligent Capture client modules is defined during *module setup*. You can run a module in setup mode from Intelligent Capture Designer, from Intelligent Capture Administrator, or from a command prompt with appropriate arguments. For more information, see *OpenText Intelligent Capture - Module Reference (ECPCORE-CMD)*.

When launched in setup mode, a client module displays the **Setup** window in which you can specify the module's configuration settings.

2.1.1 Setting Up an Image Converter Step

Before running Image Converter for setup, ensure the IA value assignments between the Image Converter module and the batch creating module are provided properly. For more information, see "IA Value Assignments Between Image Converter and Batch Creating Modules" on page 23.

To set up an Image Converter step:

1. Run the Image Converter module for setup. For more information on running a module for setup, see *OpenText Intelligent Capture - Module Reference (ECPCORE-CMD)*.

2. To set up a step, select one of the existing Image Conversion profiles, created using the Image Conversion editor of Intelligent Capture Designer. Configured for every specific business scenario, an Image Conversion profile defines how input files must be converted into the specified output format. For more information on configuring Image Conversion profile properties, see *OpenText Intelligent Capture - Designer Guide (ECPCORE-CPD)*.

   Note: The list is automatically filtered to include only those profiles that are compatible with the Image Converter step trigger level.

   "Trigger Levels" on page 8 explains how trigger level depends on the input and output types selected in the profile.

Table 2-1: Trigger Levels

Input	Output	
	Single-page File	Multi-page Document (Task Level)
Single-page Image	Level 0 or Level 1	Level 1 or higher
Multi-page Document (Page Level)	Level 1	

Input	Output	
	Single-page File	Multi-page Document (Task Level)
Multi-page Document (Document Level)	Level 1	

The list of available profiles may include either profiles created and deployed using Intelligent Capture Designer or profiles generated by upgrading a previous version of Image Converter. For more information on upgrading module step settings created in the previous Intelligent Capture versions, see *OpenText Intelligent Capture - Installation Guide (ECPCORE-IGD)*.

2.2 Running Image Converter in Production

After the module has been set up, tested, and placed in a production environment, operators and administrators will run the module in production.

For information related to common production tasks for unattended modules, see *OpenText Intelligent Capture - Module Reference (ECPCORE-CMD)*.

Note: The *Running Modules in Production Mode* section contains production topics that are common to many unattended modules and may not apply to this module.

2.2.1 Understanding Task Processing in Image Converter

Depending on the batch creating module used in a process, the Image Converter module reads input data for processing at page level (level 0) or at document level (level 1). Image Converter can handle single-page images or multi-page documents as input files.

Note: For information about post installation steps needed to run Image Converter as a service and the configuration steps needed to process various types of files, see *OpenText Intelligent Capture - Installation Guide (ECPCORE-IGD)*.

"Behaviors of Input/Output Types" on page 10 provides a description of the Image Converter module behavior for different combinations of input and output types.

Table 2-2: Behaviors of Input/Output Types

Input Source	Single-page File Output Destination	Multi-page Document (Task Level) Output Destination
Single-page Image	The module performs the following: 1. For each page in the task, converts input files (at level 0) to the output format as it is defined in the profile setup for the step. 2. Places the result to the `Level0_OutputImage` IA value of the level 0 node. 3. The content of each page is detected and the result is placed to the `Level0_OutputContentType` IA value of the level 0 node.	The module performs the following: 1. Converts input files (at level 1) to the output format as it is defined in the profile setup for the step, and merges all input images from this task into a single multi-page document. 2. Places the result to the `OutputFile` IA value of the task node.
Multi-page Document (Page Level) or **Multi-page Document (Document Level)**	The module performs the following: 1. For each page in the task, splits the file into pages, and converts to the output format as it is defined in the profile setup for the step. For more details, see "Understanding Splitting of Multi-Page Documents" on page 11. 2. Places the resulting pages to the `Level0_OutputImage` IA value at page level.	The module performs the following: 1. For each document in the task, splits the file into pages, converts to the output format as it is defined in the profile setup for the step, and merges the resulting pages. 2. Places the result to the `OutputFile` IA value at task level.

Notes

- You can exclude pages from being processed by the Image Converter module by setting the `Skip` IA Value. For information on setting the `Skip` IA Value, see "IA Values" on page 14.
- After splitting or merging, a natively created resulting *PDF* document does not contain any action requirements and all its document restrictions are set to **Allowed**.

- Not all PDF documents can be converted into the PDF/A format. For more information, see *OpenText Intelligent Capture - Operating Specifications Help (ECPCORE-H-RLI)*.

2.2.1.1 Understanding Splitting of Multi-Page Documents

Examples of processing multi-page input files.

Multi-page documents at document level

A 10-page *PDF* document is imported by the Standard Import module.

The Image Converter module takes multi-page documents from a document level and splits them into pages. Each page is processed and converted to the output format as it is defined in the profile. All documents under the task node are considered.

Multi-page documents at page level

A Microsoft Word document is imported by the ScanPlus module.

The Image Converter module takes multi-page documents from a page level (each page can be a `.doc` file) and splits them into pages. Each page is processed and converted to the output format as it is defined in the profile. All pages under the task node are considered.

2.2.1.2 Configuring the workflow to capture page counts for a TIF file with multiple pages

This section describes the procedure to configure the workflow to capture the information about the page count when you process a TIF image with multiple pages.

An *`IAValue`* in the Image Converter module is used for processing the input file name. The input file name identifies the input source image that contains multiple pages. Figure 2-1 shows the value assignment mapping of *`StandardImport:1:SourcePath`* to `ImageConverter:1:InputFilePath`.

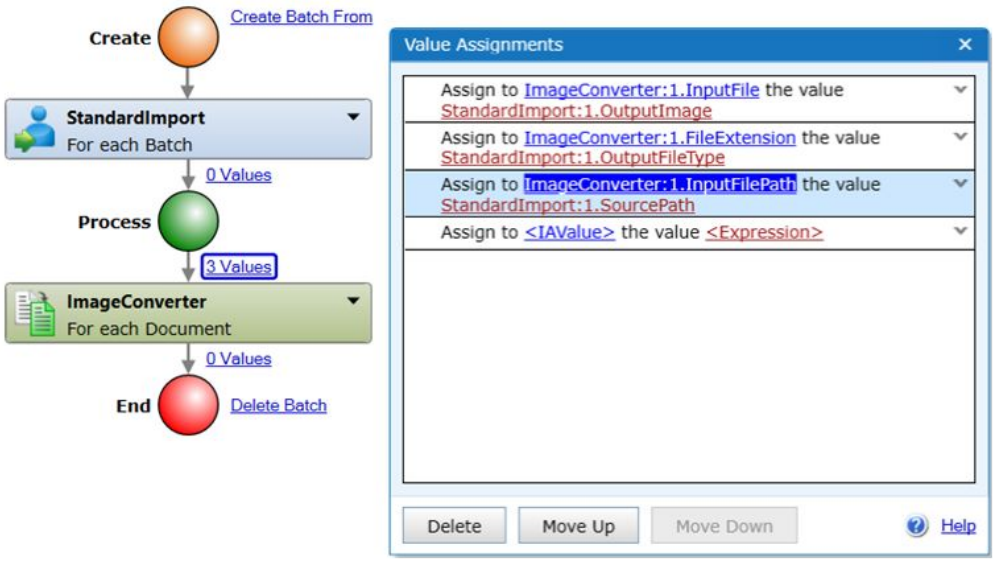

Figure 2-1: Value assignment map

Both the Standard Import and Image Converter modules log activities to the `Tbl_ReportCreatePages` database table. To configure this logging, you must enable the following Log Rules in `Intelligent Capture Administrator > Reports/Logs > View Log Rules`:

- ReportBatchCreate
- ReportNodeCreate
- ReportTaskFinishCreatePage
- ReportTaskFinishDonePage
- ReportTaskFinishPage
- ReportTaskFinishTask

The following SQL query retrieves the number of pages processed by the Image Converter in a TIF file containing multiple pages.

```
SELECT COUNT(*) as PageCount, RCPBatchUUID as BatchUUID, RCPSource as Source from
(select RCPSource, RCPBatchUUID from Tbl_ReportCreatePages (nolock) where
RCPModule='IMGCONV') as TableQuery group by RCPBatchUUID, RCPSource
```

2.2.1.3 Changing the Structure of Nodes

Image Converter changes the node structure after splitting multi-page documents at page level. The module adds additional nodes between first pages of each document. The number of added nodes in each case corresponds to the number of pages of the document. The new created nodes have the blank IA values for previous steps, while each first page contains the full IA value information from the previous steps. This information can be used for merging pages and restoring the structure of the source multi-page document.

In the example Figure 2-2, Node 2 and Node 9 store the first pages of the documents. After splitting, they contain the full information about the documents. Additional Node 102 and Node 103 are added for Document 1, and Node 104 for Document 2, appropriately.

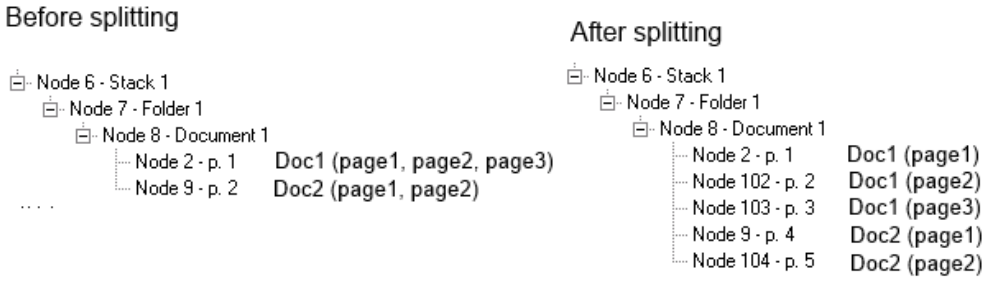

Figure 2-2: Changing the node structure

2.2.2 Recognizing Non-Image Files

The Image Converter module processes non-image files in different ways depending on the type of input non-image files. The type of imported file is recognized by its extension based on the `FileExtension` IA value. The `FileExtension` IA value is required to be non empty for non-image files, as it impacts these files being processed.

Additionally, you can select an appropriate rendering engine for processing non-image files. When **Embedded Image Converter Engine** is selected, it does not require installing any third-party applications for processing input files. In case of selecting any other engine, virtual printing technology is used and it requires installing either Microsoft Office or Internet Explorer.

Caution

The Image Converter module cannot process non-image files if the Image Converter machine is accessed using Remote Desktop Connection (RDC). This limitation is due to the way RDC handles mapping of printers to the host machine.

Considerations when using virtual printers

Intelligent Capture virtual printers process all supported non-image files, except for *PDF* files.

Managing Intelligent Capture virtual printers is available in the Windows standard **Devices and Printers** window.

In case you need to run multiple instances of Image Converter simultaneously, using virtual printers will have a number of specifics. To learn about the collaborative work of multiple instances of Image Converter and virtual printers, and peculiarities of connecting virtual printers to ports, see *OpenText Intelligent Capture - Installation Guide (ECPCORE-IGD)*.

2.3 Reference—Image Converter

The topics within this section provide descriptions of windows accessible from the application. The topics list the user interface element name and include a brief description of actions available from the window.

2.3.1 IA Values

The topics in this section describe IA values that can be used with this application.

Intelligent Capture provides both common and module-specific IA values. For more information on common IA values valid for all modules, see *OpenText Intelligent Capture - Module Reference Help (ECPCORE-H-CMD)*.

This section describes the IA values defined in `imgconv.mdf`, the Image Converter module's Module Definition File (*MDF*). Although you can add custom variables to `imgconv.mdf`, this file is overwritten each time you upgrade the product. Therefore, it is recommended that you either use dynamic values in your *IPP* or create a custom MDF for your variables so they are not overwritten.

IA values that are specific to Image Converter are described in "IA Values for Image Converter" on page 15. The **Type** column identifies the required data type (file, string, and so on) for each IA value followed by the type of IA value (input and/or output).

Table 2-3: IA Values for Image Converter

IA Value	Level	Type	Description
`InputFile`	1 or 0	File, Input	The input file of the task on the capture server. It is a multi-page document at document level (level 1), and either a single-page or a multi-page document at page level (level 0). The input file can be an image or non-image file type.
`InputFilePath`	0 – 6	String, Output	The source path for the image file, including the file name and its extension.

IA Value	Level	Type	Description
`FileExtension`	1 or 0	String, Input	A string value indicating the type of input file, either an image or non-image file. Valid values are specified in the lists of supported image and non-image file formats. **Note:** The value is required when processing non-image files. Moreover, if the value is empty or the value is an extension that is unknown to Image Converter, then the module will try to process input files as images. If input files are non-image files, the processing cannot be performed. `FileExtension` is not case-sensitive.
`Skip`	0	Boolean, Input, No Trigger	The value defines pages to be excluded from the output file. If the value contains **True**, the corresponding page will be skipped.

IA Value	Level	Type	Description
`Level0_OutputContentType`	0	String, Output	Indicates the type of content for each output page. The Image Converter profile's **Output Destination** property must be set to **Single-page File**; otherwise, if it is set to **Multi-page Document (Task Level)**, then this value would be empty. Values generated by Image Converter are the following: • `Image` The value, if any one of the following is true: – The input file is one of the following: ○ image format file; ○ a PDF file or page that contains only images. – The input file is *HTML*, *HTM*, or *MHT* and the **HTML Rendering Engine** is **Internet Explorer**. – The output format file type is not PDF or PDF/A. – If the output format file type is PDF or PDF/A, but **PDF Options** > **Keep Input**

IA Value	Level	Type	Description
			Textual Data is disabled. • `Mixed` The value, if a PDF input page or a non-image format file has both images and text. **Note:** **PDF Options** > **Keep Input Textual Data** must be enabled. • `Text` The value, if a PDF input page or a non-image format file contains only text. **Note:** **PDF Options** > **Keep Input Textual Data** must be enabled. • `Unknown` The value if a PDF input page or a non-image format file does not contain text or images. For example, the PDF input file might contain blank pages or pages with graphical objects (for example, lines and curves) only. **Note:** In PDFs, some graphical objects (for example, lines and curves) are not considered

IA Value	Level	Type	Description
			to be images. Therefore, if all images in a PDF input page are graphical objects, then this value would be one of the following: • `Unknown`, if, in addition, the page does not contain text. • `Text`, if, in addition, the page contains text.
`Level0_OutputImage`	0	File, Output	The single-page output image file for the task.
`Level0_OutputFileExtension`	0	String, Input	A string value indicating the type of output file that is automatically generated based on the output file format.
`Level0_TimeTotal`	0	Long, Output	The total processing time for a single page in milliseconds.
`AttachmentsExtractedCount`	1 or 0	Long, Output	Number of attachments that were extracted.
`AttachmentsDocumentCount`	1 or 0	Long, Output	Number of attachments that were embedded in the input PDF.

IA Value	Level	Type	Description
`AttachmentFile<[n]>`	1 or 0	File, Output	Attachment file that was extracted from the input PDF. *<[n]>* is an integer that identifies the file according to the order in which it was extracted. For more information about extracting PDF attachments, see *Intelligent Capture Designer Guide*.
`AttachmentFileName<[n]>`	1 or 0	String, Output	File name of the attachment that was extracted. *<[n]>* is an integer that identifies the file according to the order in which it was extracted. For more information about extracting PDF attachments, see *Intelligent Capture Designer Guide*.
`AttachmentFileType<[n]>`	1 or 0	String, Output	File type of the attachment that was extracted. *<[n]>* is an integer that identifies the file according to the order in which it was extracted. For more information about extracting PDF attachments, see*OpenText Intelligent Capture - Designer Guide (ECPCORE-CPD).*
`OutputFile`	Trigger	File, Output	The multi-page output file for the task.

IA Value	Level	Type	Description
`OutputFileExtension`	Trigger	String, Output	The output file extension (automatically generated based on the output file format).
`ImageConverter_ConversionTimeout`	Trigger	Long, Input	The value in seconds to attempt converting of non-image files. Overrides the corresponding setting in setup.
`Profile`	Trigger	String, Input, No Trigger	The value allows to replace the setup profile. If the value is empty, then the setup profile is used. If the value contains an existing profile, this profile is used. If the value contains a profile which does not exist, the error message is shown.
`Title`	Trigger	String, Input	*PDF* output only. A document title for the document information source set to IA value. This value is taken from the Info Dictionary of the input PDF file to be included to the output PDF file, if it is defined in Image Conversion profile.
`Subject`	Trigger	String, Input	PDF output only. A document subject for the document information source set to IA value. This value is taken from the Info Dictionary of the input PDF file to be included to the output PDF file, if it is defined in Image Conversion profile.

IA Value	Level	Type	Description
Author	Trigger	String, Input	PDF output only. A document author for the document information source set to IA value. This value is taken from the Info Dictionary of the input PDF file to be included to the output PDF file, if it is defined in Image Conversion profile.
Keywords	Trigger	String, Input	PDF output only. A document keywords for the document information source set to IA value. This value is taken from the Info Dictionary of the input PDF file to be included to the output PDF file, if it is defined in Image Conversion profile.
DivKeywords	Trigger	String, Input	PDF output only. A separator to concatenate keywords with.
TimeAvgTotal	Trigger	Long, Output	The average page processing time in milliseconds.
ErrorName	Trigger	String, Output	The name of any error that occurs during processing.
ErrorNumber	Trigger	Long, Output	The number of any error that occurs during processing.
ErrorText	Trigger	String, Output	The text of any error that occurs during processing.

2.3.1.1 IA Value Assignments Between Image Converter and Batch Creating Modules

When assigning IA values between the Image Converter module and the batch creator module steps in your CaptureFlow, consider the **Input Source** value of Image Conversion profile that you will use for the step setup.
For example:

- If the Standard Import module's MDW or Email type is used, the **Input Source** value of the profile must be **Multi-page Document (Document Level)**.
- If ScanPlus module is used, the **Input Source** value of the profile must be either **Single-page Image** or **Multi-page Document (Page Level)**.

This table contains the information on the IA value assignments that must be provided for different combinations of Image Converter and a batch creating module. For more information on assigning IA values in a CaptureFlow, see *Adding Value Assignments* section of the *OpenText Intelligent Capture - Designer Guide (ECPCORE-CPD)* Guide.

Table 2-4: IA Value Assignments Between Image Converter and Batch Creating Module

Batch Creating Module	IA Value Assignment
Standard Import (MDW)	• Assign the value *`MultiDirectoryWatch:1.OutputImage`* to *`ImageConverter:1.InputFile`*. • Assign the value *`StandardImport:1:SourcePath`* to *`ImageConverter:1:InputFilePath`*.
Standard Import (Email)	Assign the value *`EmailImport:1.OutputFile`* to *`ImageConverter:1.InputFile`*. **Note:** The *`EmailImport:1.OutputFile`* parameter can differ according to the actual output of the Standard Import (Email) module (for example, *`AttachmentName`*, *`AttachmentType`*, and others)
ScanPlus	• Assign the value *`ScanPlus:0.OutputImage`* to *`ImageConverter:0.InputFile`*. • Assign the value *`ScanPlus:0.OutputFileType`* to *`ImageConverter:0.FileExtension`*.

Note: The Image Converter module is compatible with the ScanPlus module to process files from the level 0 that is different than Image Converter trigger

level. Thereby, when a CaptureFlow includes both ScanPlus and Image Converter steps, Image Converter, triggered at document level (level 1), can take files for processing from the page level (level 0).

2.3.2 Image Size and Resolution Limits

When using Image Converter to process images, consider image size and resolution limits specified in *OpenText Intelligent Capture - Operating Specifications (ECPCORE-RLI)*.

2.3.3 File Formats Supported by Image Converter

For the full list of supported image and non-image file formats that the Image Converter module can handle, see *OpenText Intelligent Capture - Operating Specifications (ECPCORE-RLI)* and *OpenText Intelligent Capture - Operating Specifications (ECPCORE-RLI)*.

Chapter 3

Image Processor Module

This section includes the Intelligent Capture Image Handling module: **ecpcore-cip**.

The Image Processor module can be used in a capture process for the following image handling tasks:

- **Image processing**: by improving the quality of and detecting certain kinds of information in image data. Improving the images quality can result in smaller files sizes and improved OCR results, because images with clean backgrounds and text at right angles to the sides of the page compress to smaller sizes and provide faster and more accurate recognition.
- **Image detection**: by detecting image features, such as blank pages, percentage of color in an image, and the presence of patch codes and barcodes which can later be used for automatic classification and indexing.
- **Applying annotations** to *TIFF* images.

To perform the preceding operations, Image Processor needs to link an *Image Processing profile* that defines image filters. A profile must be created in Intelligent Capture Designer and uploaded to the server. During step setup, the Image Processing profile links the "image processor" step of the process uploaded to the same server.

Image Processor ships with a collection of predefined filters which you can customize in Image Processing profiles. You can extend this collection as well by creating custom filters. For details on creating custom filters, see *OpenText Intelligent Capture - Scripting Guide (ECPCORE-PSC)*.

When a filter is added to a profile, its filtering parameters are configured according to business needs. During production stage, the filters in the profile are applied to all images with the same set of pre-configured filtering settings. To make image filtering more flexible, you can create a script that will dynamically replace the predefined filtering settings of the profile by custom settings when necessary. For details, see *OpenText Intelligent Capture - Scripting Guide (ECPCORE-PSC)*. For details on available filters and annotations, see *OpenText Intelligent Capture - Designer Guide (ECPCORE-CPD)*.

3.1 Setting Up Image Processor

Most configuration information for Intelligent Capture client modules is defined during module setup. You can run a module in setup mode from Intelligent Capture Designer, from Intelligent Capture Administrator, or from a command prompt with appropriate arguments. Find the details in *OpenText Intelligent Capture - Module Reference (ECPCORE-CMD)*.

When launched in setup mode, a client module displays the **Setup** window in which you can specify the module's configuration settings.

3.1.1 Setting Up Image Processor Step

A step of the Image Processor module is usually included in a process at the point where improving image quality or detecting image data makes sense. For example, a process that uses ScanPlus, NuanceOCR, and Documentum Advanced Export could benefit from an Image Processor step either before the indexing step (to extract the barcode whose value can be pre-populated into an index field) or before the NuanceOCR step (to improve image quality for better text recognition results).

A single step of Image Processor can perform several operations on an image in sequence. For example, it can deskew and remove the binder hole images, then detect and remove blank pages, and then detect and capture barcode data. Alternatively, multipleImage Processor steps can be used in a single process to perform different operations at different places in the process. For example, one step performing blank page removal and binder hole removal before a NuanceOCR step.

Note: Image Processor should always run at level 0 only (page level), because it can only process tasks that consist of a single page.

To set up the module, run it in setup mode and specify the required steps.

Set up the Image Processor step as follows:

1. Run the Image Processor module for setup. For the detailed information on running a module for setup, see *OpenText Intelligent Capture - Module Reference (ECPCORE-CMD)*.

2. To set up a step, select one of the existing Image Processing profiles, created using the **Image Processing** area of Intelligent Capture Designer. For more information, see *OpenText Intelligent Capture - Designer Guide (ECPCORE-CPD)*.

3. (Optional) If you use Image Processor scripting, use the **Custom Values** field to enter the custom filtering parameters that will replace the predefined ones. Use the format that is supported by the scripting logic.

4. Click **OK** to save the changes and exit the setup window.

3.2 Running Image Processor in Production

After the module has been set up, tested, and placed in a production environment, operators and administrators will run the module in production.

For information related to common production tasks for unattended modules, see *OpenText Intelligent Capture - Module Reference (ECPCORE-CMD)*.

Notes

- The *Running Modules in Production Mode* section contains production topics that are common to many unattended modules and may not apply to this module.
- If you pass a color image to Image Processor during processing and have a binary filter configured for the profile, the module will convert the image to binary (black and white) image before processing it.

3.3 Reference—Image Processor

The topics within this section contain reference information useful while using the application in setup or production.

3.3.1 IA Values

The topics in this section describe IA values that can be used with this application.

Intelligent Capture provides both common and module-specific IA Values and Common IA Values described in the *OpenText Intelligent Capture - Module Reference (ECPCORE-CMD)*. These values are valid for all modules.

This section describes the IA Values defined in `CPIMGPRO.mdf`, the Image Processor module's Module Definition File (*MDF*). Although you can add custom variables to `CPIMGPRO.mdf`, this file is overwritten each time you upgrade the product. Therefore, we recommend that you either use dynamic values in your *IPP* or create a custom MDF for your variables so they are not overwritten.

IA Values that are specific to Image Processor are described in "IA Values for Image Processor" on page 28. The **Type** column identifies the required data type (file, string, and so on) for each IA Value followed by the type of IA Value (input and/or output). Ensure, IA values are assigned in your CaptureFlow as follows. For the detailed information on assigning IA values in a CaptureFlow, see *OpenText Intelligent Capture - Designer Guide (ECPCORE-CPD)*.

Note: The Image Processor module runs at level 0 only.

Table 3-1: IA Values for Image Processor

IA Value	Level	Type	Description
`InputImage`	0	File, Input	Input image
`OutputImage`	0	File, Input	Processed output image
`Profile`	T	String, Input	At runtime, if the Image Processing profile is not set, then use the one defined in the module setup. If the Profile is set to `$nop`, skip processing (the task status is *successful*) and do not log an error. If the Profile is selected but it does not exist or is corrupted, log an error and put the task in the *error* state.
`StepCustomValue`	T	String, Input, Output, NoTrigger	Used as scripting parameters and overrides those defined in the step setup.

In addition to Image Processor specific IA Values, there are several out-of-the-box IA Output values produced from Image Processing profile:

- If it is an array value (for example, color marks and barcodes), the results are added using the index number, and each of them has a count.
- For non-array values, the last filter result overwrites any previous filter results.

Each value is level 0 and has output type with data type defined in "Out-of-the-box IA Output Values" on page 29. The default value is used is the natural default of the data type (meaning *0* for numbers, *blank* for strings).

Table 3-2: Out-of-the-box IA Output Values

Name	Type	Filter Used (the IA value is set if that named filter is used in the profile that is executed)	Description
`BlankPage`	Long (-1 to 1)	DetectBlankPage	• 0 if the filter is not used. • 1 if blank page is found. • -1 if non-blank page is found.
`Patchcode`	String	DetectPatchcode	Last patch code found. Blank may mean that the filter was not used.
`Barcode{0}_Text`	String	DetectBarcodes	Return the first detected text value from the DetectBarcodes filters, in detection sequence.
`Barcode{0}_Conf`	Long	DetectBarcodes	Return the first confidence value from the DetectBarcodes filters, in detection sequence.
`Barcodes_Count`	Long	DetectBarcodes	Return the number of detected barcodes.
`ColorMarks_Count`	Long	DetectColorMarks	Return the number of detected color marks.
`Color`	Long (-1 or 1)	ConvertToBlackWhiteAdvanced	0 if the filter was not used. If ConvertToBlackWhiteAdvanced filter returns color=true, then set to 1. Otherwise, set to -1.
`DeskewAngle`	Double	Deskew	Last detected skew angle. Can be 0 if the filter is not used.
`RotationAngle`	Long (0, 90, 180, 270)	Rotate	Last detected rotation angle. If set according to Rotate result: 0–set 0, 1–set 270, 2–set 90, 3–set 180.

Name	Type	Filter Used (the IA value is set if that named filter is used in the profile that is executed)	Description
`ImageCompression`	Long, Output	ImageCompression as Long,Output **Note:** This value is always set.	The compression type to be used for saving images. • No compression. • Modified (no *EOL*) *CCITT* Group 3 1-Dimensional Huffman run-length encoding; each line starts on a new byte. BitsPerSample and SamplesPerPixel must be 1, because this type of compression is defined only for binary images. • Standard CCITT Group 3 coding, 1-Dimensional Huffman run-length encoding. BitsPerSample and SamplesPerPixel must be 1, because this type of compression is defined only for binary images. • Standard CCITT Group 4 encoding. BitsPerSample and SamplesPerPixel must be 1, because this type of compression is defined only for binary images. • *TIFF* images with *JPEG* compression identical to files

Name	Type	Filter Used (the IA value is set if that named filter is used in the profile that is executed)	Description
			produced in earlier versions of Intelligent Capture. • TIFF images with JPEG compression using the latest recommendations from the TIFF Advisory Committee. • 34661: *JBIG* compression, typically used for binary and grayscale only. • 32771: No compression, each line padded to two bytes. • 32773: Packbits compression. • 50000: Compression scheme used in *PCX* files. • 50001: No compression, each line padded to four bytes. • 50006: Compression used in some *BMP* files. • 50014: Wang format JPEG encoding, typically used for grayscale and color images only.

Name	Type	Filter Used (the IA value is set if that named filter is used in the profile that is executed)	Description
`ImagePhotometric`	Long, Output	ImagePhotometric as Long,Output  **Note:** This value is always set.	The photometric interpretation which specifies how to interpret the image data. Possible values are: • 0 for WHITE0 (binary images). • 1 for WHITE1 (16- or 256-level gray images). • 2 for RGB (3-, 24-, or n-bit color images). • 3 for PALETTE(16- or 256-color palette).
`ImageSamplesPerPixel`	Long, Output	ImageSamplesPerPixel as Long,Output  **Note:** This value is always set.	The number of values each pixel uses to represent its data.
`ImageResolution`	Long, Output	ImageResolution as Long,Output  **Note:** This value is always set.	The image's X resolution in *DPI*.
`ImageWidth`	Long, Output	ImageWidth as Long,Output  **Note:** This value is always set.	The width of the image, in pixels.
`ImageLength`	Long, Output	ImageLength as Long,Output  **Note:** This value is always set.	The length of the image, in pixels.

Name	Type	Filter Used (the IA value is set if that named filter is used in the profile that is executed)	Description
`OriginalSize`	Long, Output	OriginalSize as Long,Output **Note:** This value is always set.	Amount of memory or disk space occupied by the compressed original image in bytes.
`FinalSize`	Long, Output	**Note:** This value is always set.	Image size after all filters have been applied, in bytes.

You can use Intelligent Capture Designer (or modify an MDF file) to add more IA values. The return values are generated and named in the way they are provided in "Return IA Values" on page 33.

Table 3-3: Return IA Values

Name	Type	Filter Used	Description
`Barcode{0}_Text`	String	DetectBarcodes	Return text values from the DetectBarcodes filters, in detection sequence. {0} is 1 to 9.
`Barcode{0}_Conf`	Long	DetectBarcodes	Return confidence values from the DetectBarcodes filters, in detection sequence. {0} is 1 to 9.
`Barcode{0}_Height`	Long	Detect Barcodes	Return result from the DetectBarcodes filters, in detection sequence. In pixels. {0} is 0 to 9.
`Barcode{0}_Width`	Long	Detect Barcodes	Return result from the DetectBarcodes filters, in detection sequence. In pixels. {0} is 0 to 9.
`Barcode{0}_X`	Long	Detect Barcodes	Return result from the DetectBarcodes filters, in detection sequence. In pixels. {0} is 0 to 9.

Name	Type	Filter Used	Description
`Barcode{0}_Y`	Long	Detect Barcodes	Return result from the DetectBarcodes filters, in detection sequence. In pixels. {0} is 0 to 9.
`Barcode{0}_Type`	Long	Detect Barcodes	Return result from the DetectBarcodes filters, in detection sequence. Format barcode type name. {0} is 0 to 9.
`CropTop`	Long	Crop	Last crop margins in pixels. 0 may mean filter was not used.
`CropBottom`	Long	Crop	Last crop margins in pixels. 0 may mean filter was not used.
`CropLeft`	Long	Crop	Last crop margins in pixels. 0 may mean filter was not used.
`CropRight`	Long	Crop	Last crop margins in pixels. 0 may mean filter was not used.
`BlackBarLeft`	Long	Remove Black Bars	Last detected black bar margins in pixels. 0 may mean filter was not used.
`BlackBartTop`	Long	Remove Black Bars	Last detected black bar margins in pixels. 0 may mean filter was not used.
`BlackBarRight`	Long	Remove Black Bars	Last detected black bar margins in pixels. 0 may mean filter was not used.
`BlackBarBottom`	Long	RemoveBlackBars	Last detected black bar margins in pixels. 0 may mean filter was not used.
`ColorAmount`	Long (0 to 100)	ConvertToBlackWhiteAdvanced	Last detected percentage of color pixels (not black or white) to all pixels.

Name	Type	Filter Used	Description
Colorfulness	Double	DetectOverallColorfulness	Last detected percentage of image that is colorful (not black, white, or grey) Value from 0 to 1000, which matches old IE module's ColorContent value.
ColorMark0_Bottom	Long	DetectColorMarks	Result from filter, in detection sequence. In pixels.
ColorMark0_Color	Long	DetectColorMarks	Result from filter, in detection sequence. Format is RGB packed into 32-bit integer.
ColorMark0_Left	Long	DetectColorMarks	Result from filter, in detection sequence. In pixels.
ColorMark0_Right	Long	DetectColorMarks	Result from filter, in detection sequence. In pixels.
ColorMark0_Top	Long	DetectColorMarks	Result from filter, in detection sequence. In pixels.

3.3.2 Image Size and Resolution Limits

When using Image Processor to process images, consider image size and resolution limits. For information about image size and resolution specifications, see *OpenText Intelligent Capture - Operating Specifications (ECPCORE-RLI)*.

3.3.3 TIFF Image Formats Supported by Image Processor

Image Processor supports only input *TIFF* images. Output TIFF images have the same color format; however, for some images, the compression type cannot be preserved.

Note: The third-party filters included with Image Processor may have the capability to support input and output image formats other than TIFF, although it cannot be guaranteed.

For the full list of supported image and non-image file formats that the Image Processor module can handle, see *OpenText Intelligent Capture - Operating Specifications (ECPCORE-RLI)*. For annotations, use the same file formats as for

Completion described in *OpenText Intelligent Capture - Operating Specifications (ECPCORE-RLI)*.

Glossary

BMP

Bitmap file extension

CCITT

Comité consultatif international téléphonique et télégraphique (French for International Telegraph and Telephone Consultative Committee, became the ITU Telecommunication Standardization Sector, ITU-T, in 1992)

dpi

Dots Per Inch

EOL

End Of Life

HTM

HTML file extension.

HTML

HyperText Markup Language

IPP

Integrated ProcessFlow Project

JBIG

Joint Bi-level Image Experts Group

JPEG

Joint Photographic Experts Group

MDF

Module Definition File

MHT

Microsoft Hypertext Archive (file type/extension). This is a Web page archive file format.

PCX

PC Paintbrush bitmap format file extension

PDF/A

Portable Document Format Archive

PDF

Portable Document Format

TIFF

Tagged Image File Format