opentext™

OpenText™ Intelligent Capture

Extraction Guide

This guide describes the extraction module which is an optional step. You can use this Intelligent Capture client module to automatically extract data from documents using optical character recognition.

ECPCORE220200-CEX-EN-01

OpenText™ Intelligent Capture
Extraction Guide

ECPCORE220200-CEX-EN-01
Rev.: 2022-Mar-17

This documentation has been created for OpenText™ Intelligent Capture CE 22.2.
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

Introduction

This guide describes the extraction module. Extraction is an optional step in a process. You can use this Intelligent Capture client module to automatically extract data from documents using optical character recognition (*OCR*).

At run time, the Extraction module identifies the printed or handwritten characters that make up a document and returns the information in the form of text. Extraction is triggered at any level (0 through 7).

For more information see *OpenText Intelligent Capture - Module Reference (ECPCORE-CMD)*.

1.1 Relationship with Recognition Projects, Document Types, and Templates

During CaptureFlow design, you can set up the required components for the extraction step. Begin by creating a *recognition project* for a paper form in Recognition Designer. A recognition project identifies the templates, base images, and rules for classifying documents. It also identifies the *OCR* engine(s) to use for extracting data from documents. You can use a single recognition project for all document capture jobs or create different projects based on specific processing needs.

Next you create a *document type* for the paper form using the Document Designer and associate it with the recognition project. The document type defines the data entry form that Completion module operators use for indexing and validation. Document type definition includes designing the layout of the data entry form and specifying document-level validation rules.

When you save the new document type, Document Type Editor generates an *indexing family* within the recognition project. The indexing family contains all the index fields for all pages of the document.

After document type creation, use the Recognition Designer to create *page templates* for the document type and then map them to the indexing family. A page template identifies an image. It includes the following:

- A label for the template.
- Information that identifies the template for classification purposes. For example, an IBM logo and text such as **Total Amount Due** can identify a page image as an IBM invoice.
- Layout information that specifies where data is located on the form (zones).
- Configuration information that specifies how to extract the data, including which OCR engines to use and how to pre-process the image.

Each page of the document can be associated with a different page template.

After configuring page templates for a form, place index family fields on the template and configure the extraction settings in Recognition Designer. Fields can also be placed on the template automatically using the Template Wizard. This information, which is saved in the recognition project, identifies which data should be extracted from the image and written to the data entry field.

For example, assume you have an index family called **Telephone Bill**. You can position the **Total Amount Due** field on the bill on the **Telephone Bill Page 1** template, the **Usage Charges** field on page 2 on the **Telephone Bill Page 2** template, and so on.

A field can be positioned on multiple pages in the document. For example, the **Total Amount Due** field might appear on both the first and last page. In this case, the data from the first extracted page is written to the field.

Prior to Extraction module setup, ensure that the location of the recognition projects folder is defined on the **System Configuration** tab of the **System** area in Intelligent Capture Designer. The Extraction module uses the specified path to locate your recognition projects. Then during Extraction module setup, select the recognition project to be used for the step. This setting, along with other setup options, determines how the Extraction module recognizes pages when it processes tasks in production mode.

1.2 Process Flow

The process flow for the Extraction module at run time is as follows:

1. The document is pre-separated, for example, using Classification, Identification, or some other intelligent document assembly mechanism.
2. Extraction sets the internal document type to construct the internal document type data object to the first of:
   - The document-level `UimDocumentType` if non-blank and valid (that is, found in the list of document types). This case is used when the structured documents are classified by setting the document type directly (such as through a barcode) rather than by using Classification.
   - The document type associated with the template of the first page.
   - If the above methods fail, the document does not have a document type. However, the task does not fail.
3. Extraction chooses the template for each page as follows:
   - Extraction determines the page template from the `PageTemplateId` value. This value either skips the page from data extraction, or contains a valid template ID which is taken, or tells Extraction to proceed to input XML data.
   - If specified by the `PageTemplateId` value, Extraction parses input XML data (page-level `InputPageDataXml` value). Input *XML* must be set to the output

of Classification or Identification. If the XML or the page template ID that it contains is invalid, nothing is extracted from that page.

- If `InputPageDataXml` is blank, Extraction determines the page template from the document type (as determined by the steps in the previous bullet) as follows:
  - At design time, it assigns the ordered set of template IDs using Recognition Designer. This set of template IDs is stored as part of the document type definition.
  - The first page in the document is assigned the first template ID, the second page is assigned the second template, and so on. For example, if a document type has template IDs 10, 11, 12, and 13 and `InputPageDataXml` does not exist for the third page in the document, Extraction would use template ID 12 (the third item in the list) regardless of whether the first two pages had `InputPageDataXml`.
  - If the number of document pages exceeds the number of templates, the extra pages are not associated with a template and therefore are not recognized. If there are more templates than pages, the extra templates are not used.
- If none of the above (`PageTemplateId`, `InputPageDataXml`, or a document type indicated by `UimDocumentType`) includes a valid page template, the page is not associated with a template and no data is extracted from the page. However, the task does not fail.

4. The Extraction module extracts data from each page using *OCR*.
5. The Extraction module reconciles data from the extracted pages into a document-level object.

   For more information, see "Inputs and Outputs" on page 8.
6. The data is then validated against what is defined at design time:
   - Field validation (required fields, masking, and so on).
   - Document-level validation rules, both declarative and scripted.
   - Validations at a higher level than document, such as verifying that a field that should be common across a set of all documents, are handled by Completion rather than Extraction.
7. The Extraction module generates `OutputPageDataXml`, which is bound to the Completion module's `PageDataXml` in a typical workflow process.

1.3 Inputs and Outputs

The Extraction module uses the following as input:

- *Page images* for a document. The document must have been pre-separated in some way using Classification, Identification, or some other intelligent document assembly mechanism.

  Note: You can restrict extraction to specific pages. For more information, see the **Pages to Process** option in "Setting Up Extraction" on page 14 and the `PagesToProcess` IA value in "IA Values" on page 23.

- *Document type*. At run time, the document type can be determined in the following ways:

  – By classification if the optional Classification module is used. In this case, the page template for the first page of the document identifies the document type for the document.

  – By manual separation and barcode. In this case, manual separation provides boundaries for the document and the barcode sets the IA value for the document type. When the document type is set using an IA value and the page template is left blank, the document type also determines the page template that is used. Templates are assigned to document pages based on the alphanumeric order of the template name as defined in Recognition Designer. For example, templates named **1**, **2**, and **10** would be assigned to pages as follows: Page 1 to **1**, Page 2 to **10**, and Page 3 to **2**.

  – By Completion module operators if the ability to change or assign a document type during production is enabled during Completion setup. Assuming that your business process requires documents with changed types to be routed back to Extraction (to extract data for the new document type), you must map the newly assigned document type from the Completion output to the input document type for the Extraction step. This procedure is necessary because the Extraction step does not use a document type object as input. All steps in the process that follow the second Extraction step should use only the new output data from Extraction.

- *Page template(s)* associated with the indexing family (an internal representation of the document type in the recognition project), and field placements that identify which data should be extracted from the paper image and written to the data entry fields. If a page template is set, the document type is determined by the first page.

The Extraction module extracts field data from the document on a page-by-page basis using the field placements found for each page template. Field data is extracted as follows:

- Single values for the same field (for example, the same invoice number on multiple pages) are reconciled into one value as follows:

  - If the per-page value is blank on all pages, the document type field is blank.
  - If the per-page value is the same on all pages that have a non-blank value, that is the value for the document type field.
  - For primary (non-array) fields, either the first or last found field (as set by **Extract Page** property in the document type definition) determines the value assigned to the field. In other words, if the same field appears on multiple images in the document, the data from the first or last extracted page is written to the field. In addition, if the values differ from page to page, then the field is flagged for manual confirmation independent of the confirmation setting in the document type. For example, if the **Total Amount Due** field is found on both the first and last page and **Extract Page** is set to first, then the value from the first page is used.

- For array fields, the row data is concatenated. For example, if a table in an **Invoice** document type that contains a list of purchase items is split across two pages, the row data is concatenated together to produce the full table.
- Table data is constructed by populating the document type data with only the extracted values where the line type is primary. Secondary lines may be processed in scripting but are dropped by Extraction.

The Extraction module then generates a document type object containing the extracted field data in the form of IA values, including:

- Field data: Data from fields defined by the page template(s).
- Table data: The name and zone coordinates of the table and the values of all the cells in the table.

The document type object produced by the Extraction module serves as input to module steps later in the process such as Completion or Standard Export. Completion uses this object to identify the document type and index fields for the document. It then retrieves the index values from the document type object and pre-populates the data entry form using data from the recognized pages. Operators can then verify the accuracy of the extracted data.

1.4 Relationship with Scripting

You can use profile scripting to manipulate the document data or data entry form. For example, you may need to validate a certain extracted field value against other document values or external data. Or, if extraction of a particular field was not successful, you may need to calculate that value or retrieve it from an external source and store the result in the document data structure.

The order of execution for scripting events in Extraction is:

1. `DocumentExtracted`: Executed after all extracted values have been copied into the document type data.
2. `ExecuteValidationRule`, `ExecuteTableRowValidationRule`: Executed once per validation rule.

3. `DocumentUnload`: Executed just before the task is completed; if document type data is changed in this event, the rules that depend on the affected fields are not rerun.

For detailed information, see *OpenText Intelligent Capture - Scripting Guide (ECPCORE-PSC)*.

1.5 Document Type Statistics

Both the Extraction and Completion modules provide statistics. These statistics, which pertain to production data, can help you identify issues that affect productivity. You can:

- Measure operator productivity and performance tuning.
- Find classification problems: Identify pages for which classification failed or documents for which the document type was changed.
- Find extraction problems: Identify fields that contained low-confidence characters or for which the value was changed after extraction.
- Find throughput bottlenecks: Identify steps where extraction or validation were slow.

You specify whether statistical data is collected as part of the step configuration settings in each process. The Extraction module provides options for collecting template and document data. Completion provides options for collecting document and field data.

1.5.1 Where Data is Stored

Statistical data is stored in the following tables:

- `Template`: Stores information about the pages processed by the Extraction module. You can use this data to improve extraction results for a particular page template. Information is divided by process to enable you to organize your data by line of business in cases where templates are shared.
- `Document Type`: Stores information about the documents processed by the Extraction module and Completion. You can use this data to determine which document types take the longest time to process or have high reclassification rates. Information is divided by process to enable you to organize your data by line of business in cases where templates are shared. Capturing character counts enables accuracy measurement by calculating the percent of characters that were changed.
- `Field`: Stores information about the fields changed in Completion; records are created only for fields that the operator changes. You can use this data to determine which fields take the longest time to process or have high data correction rates. Capturing character counts enables accuracy measurement by calculating the percent of characters that were changed.

Each table contains three tags (`ReportTag1`, `ReportTag2`, and `ReportTag3`) that can be used to further divide the statistical data according to business needs. You can use these tags to capture separate data in cases where capture services are shared across an organization. For example, you could organize the statistics by functional area by setting `ReportTag1` to the functional area for each task, such as `Accounting` for one task and `Sales` for a second task. Another example might be to organize by geographical region such as `North America` versus `Europe`. You could group the statistics by these tags to show operator productivity for each functional area or geographical region.

Note: The string values you specify for these tags are written to the database exactly as entered. No processing is performed on the values. The tags are set by the IA process as task-level values with the same name.

Use these tags only when you need to further break down statistical data. Capturing additional, unused information increases the number of records in the database unnecessarily.

For a description the report tables, see *OpenText Intelligent Capture - Administration Guide (ECPCORE-AON)*

1.5.2 How Data is Captured

Each running module creates new records in the three reports tables (`Template`, `Document Type`, and `Field`) as needed, but no less frequently than one per hour. The cumulative statistics for a given date and hour are computed by aggregating all rows for that time. For example, the number of pages processed by Extraction on 1/23/2012 between 10:00 and 11:00 is the total of all records where the `Date` is set to `1/23/2012 10:00`.

Chapter 2

Setup

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 Understanding Data Flattening

The Extraction module stores document data in a single *`UimData`* IA value that contains all fields in the document. Extraction can output document data to the next CaptureFlow step in the original UIM data structure, or you can choose to convert document data into a set of dynamic values, each holding a single extracted field, in addition to outputting the data in its original structure.

Note: This option is enabled only when setup is launched from Intelligent Capture Designer. It cannot be configured if setup is run from Intelligent Capture Administrator.

The process of converting UIM data into "flat" IA values is referred to as *data flattening*. During setup, you can choose to flatten the extracted document data or to output it in its original structure. When making your choice, consider the following:

- In a typical CaptureFlow, data extraction is followed by validation or export. If you use the Completion module for data validation, you can output the *`UimData`* IA value as is. Completion supports UIM data and can flatten document data before output if necessary.
- If the next step is Standard Export then you do not need to flatten document data either. However, pre-7.0 exporters (for example, Documentum Advanced Export) cannot read data from UIM structures, and document data flattening is required.
- Flattening is required if the extracted document data is used in a CaptureFlow for making decisions, such as choosing the next step. CaptureFlow Designercannot read data from UIM structures.
- Flattening index data can increase batch size, affecting performance.

If you need individual values, you must configure them using the setup settings for IA values. During step configuration, you can choose to create new values or to only populate existing ones. You can also specify the step destination into which the values should be written. For example, you can use custom values in CaptureFlow

Designer when creating the CaptureFlow, which are then populated by the module. Another option is to use a separate data-only *MDF* that is populated by the module.

2.2 Setting Up Extraction

Configure options on the **Setup** pane in module setup to specify how the Extraction module should recognize pages when it processes tasks in production mode.

To set up Extraction:

1. Run the Extraction module for setup.

   Note: If you need to output the flattened document data, launch setup from Intelligent Capture Designer. Data flattening cannot be configured if setup is run from Intelligent Capture Administrator.

2. Specify the required settings as described in the table:

Table 2-1: Extraction module: Setup Settings

Setting		Description
Recognition Project		Expand the drop-down list and select the recognition project to use for this step. The list contains all projects found under the path defined by the **RecognitionProjectSharedDir** property in Intelligent Capture Designer.
Pages to Process	**First Pages**	Starting from the first page, specify the number of pages on which to perform extraction. **Note:** You can also use the `PagesToProcess` IA value. For more information, see "IA Values" on page 23.
	Last Pages	Starting from the last page and counting backwards, specify the number of pages on which to perform extraction.

Setting		Description
Output IA Values	**Destination**	Expand the drop-down list and select the destination and format for outputting the document data. Values: • **(Do not write to IA values)**: Select this option to output the document data in its original structure. When selected, the rest settings in the **Output IA Values** group are disabled. • **<step name>**: Select the step to which you need to output document data after flattening. • **Custom Values**: Select this option to output flattened document data into custom values available in your CaptureFlow.

<table>
<tr><th colspan="2">Setting</th><th>Description</th></tr>
<tr><td></td><td>Output Array Fields</td><td>If data flattening is enabled in the Destination setting, specify flattening for array data (extracted table data). Values:
Value Per Row: Output each array item as a separate IA value with the <FieldName>_<row #> name where the row number starts at 1.
Value Per Array Field: Output each table field (all rows) as a single <CR><LF> delimited string value with the <FieldName> name. The last item in the string is not followed by the delimiter.
In both cases, the number of table records is output to IA value <TableName>_count.
The Fruits table includes the following fields and content:<table><tr><th>ID</th><th>Description</th></tr><tr><td>1</td><td>Orange</td></tr><tr><td>2</td><td>Apple</td></tr></table>If you select the Value Per Row option, the table content will be output as follows:
Id_1=“1”
Description_1=“Orange”
Id_2=“2”
Description_2=“Apple”
Fruits_count=“2”
If you select the Value Per Array Field option, the</td></tr>
</table>

Setting		Description
		table content will be output as follows: • `Id`="1<CR><LF>2" • `Description`="Apple<CR><LF>Orange" **Note:** This value can be mapped to repeating attributes when usingDocumentum Advanced Export. • `Fruits_count`="2"
	Output Level	Expand the drop-down list and select the node to output the flattened document data: • **Document Only**: set by default. Select to output data at document level. • **Field Index Level Only**: select to output data in accordance with the **Index Level** specified in Intelligent Capture Designer for each field in a document type. • **Document And Field Index Level**: select to use both approaches to output data. **Note:** When outputting to the index level, the Extraction module ignores empty output values so that the preceding outputs are not overwritten with blanks.

Setting		Description
	Always import higher level values	This setting lets you propagate data between tasks. Disabled if the **Document Only** output level is selected. Check this option to auto-populate document fields with the common output values if any exist in the level higher than the task level. When selected, after recognition is completed, data is propagated in accordance with the selected output destination and at the index level specified for this field in the document type. When using **CustomValues** for data import, consider the UimDataImportMode allowed values.

Setting		Description
	Output As Dynamic Values	Check this option to create a dynamic value for each field that does not have a matching IA value defined in the selected destination (such as an MDF value or a custom value). Example: If a document type has `Name` and `Date` fields but the output destination has only a `Name` IA value, then selecting this option would populate the `Name` value and create a dynamic `Date` value. If you choose not to use dynamic values, the `Name` field's value would be written to the existing destination IA value and the `Date` field value would be skipped. Using dynamic values is also necessary if arrays are flattened with the **Value Per Array Field** option (where all values cannot be pre-created because the number of items in the array is not known in advance).
Statistics	**Collect template statistics**	Check this option to log statistics about the processed templates.
	Collect document type statistics	Check this option to log statistics about the processed documents.

3. Click **OK** to save the configuration settings.

Chapter 3

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.

Chapter 4

Reference

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

4.1 IA Values

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

IA values that can be used with the Extraction module are described in the following table. 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):

- Input IA values include input file variables, which serve as pointers to stage files, and input processing variables, which store setup data that a module uses to process tasks, including default and user-defined module settings.
- Output IA values include output processing variables, which store data that was generated by the module during processing. Some of the following output values are created dynamically and do not need to be specified in the `MDF` file.

Table 4-1: IA Values for Extraction

IA Value	Level	Type	Description
`DispatcherProjectName`	T	String, Input	A task level value that holds the name of the recognition project (`DPP` file) to be used by default. This value should only contain the DPP name and not the path. If the value is empty, then the `DPP` specified in Extraction setup is used. If the value contains a `DPP` which does not exist, an error message is shown.

IA Value	Level	Type	Description
`DocStatus`	1	DocumentStatus, Output	Object describing the state of each document, which can be used for routing decisions in the workflow. The document status is needed following a Classification, Recognition, or Validation step. The value is created when the task is closed, independent of whether it is completed or canceled, and is not automatically updated. The document status is made up of the following properties (and types), per document: • `CharactersInvalid` (Long): Number of remaining questionable characters. This is initially set during recognition, but a given operator may not finish all of them. If this value is nonzero, the document contains work that can be corrected by a step configured for Character work level or higher. • `FieldsInvalid` (Long): Number of fields that failed field-level validation, either due to built-in validation such as

IA Value	Level	Type	Description
			data type, range, or required properties; or due to a validation rule with only a single associated field. This number is zero if there are no single-field validation errors in the document. If this number is nonzero, it contains work that can be corrected by a step configured for Field work level or higher. • `FieldsUnconfirmed` (Long): Number of unconfirmed fields. • `FieldsFlagged` (Long): Number of fields that are flagged. • `PagesFlagged` (Long): Number of pages that are flagged. • `TotalFlagged` (Long): Number of fields and pages that are flagged, plus one if the document is also flagged. • `RemainingWork` (Long): Sum of the values for `FieldsFlagged` + `FieldsUnconfirmed` + `FieldsInvalid` + `CharactersInvalid` + [number of fields failing

IA Value	Level	Type	Description
			Document-level validation]. A value of zero means the document should be complete and ready to export. Processes can use this to determine if the document has any remaining work rather than manually adding up the individual values in a CaptureFlow Designer expression. If this value is nonzero, the document contains work that can be corrected by a step configured for the Document work level. It might also contain work applicable to lower work levels; check the other status values to find out. **Note:** Fields may be counted multiple times if multiple states apply. For example, if a field is both in error and has a flag, `FieldsInvalid` is 1, `FieldsFlagged` is 1, and `RemainingWork` is 2,

IA Value	Level	Type	Description
			although only one field needs work. Routing decisions based on information in the document status should take this into account.
`ExternalValues`	0	String	Contains a list of IA Values to be used as external values from the Extraction module in the recognition VBA scripting. To assign the list of IA Values, create the `ExternalValues` custom value for Extraction (level 0) in the CaptureFlow. Then specify the comma-separated list of required IA values enclosed in double quotes. For example, to obtain the FolderID and MachineID values, in the CaptureFlow for Extraction set them as follows: `Extraction:0. ExtervalValues = "FolderID, MachineID"`

IA Value	Level	Type	Description
`Image`	0	File, Input	The input image, generally a `TIF` file in the color and compression format output by the preceding CaptureFlow step.
`InputPageDataXml`	0	String, Input, NoTrigger	*XML* data generated by the preceding step, such as Classification or Identification, and passed in to the Extraction step. Pre-indexed fields already recognized in Classification and validated in Identification are stored in this XML object.
`OcrDataCache`	0	File, Input, Output	As an input, the optional full text *OCR* data cache, typically created by Classification. After extraction, contains the Extraction-created OCR data on output.
`OcrLocale`	1	String, Input	Contains the locale, for example `de-DE` that will be used by Information Extraction for extracting text from the images. If empty, the locale is defined by the first locale in the Information Extraction profile for the selected document type.

IA Value	Level	Type	Description
`OutputPageDataXml`	0	String, Output, NoTrigger	XML data produced by the Extraction module. This XML data serves as input to the Completion module. In a typical CaptureFlow, `OutputPageDataXml` is bound to `PageDataXml` in Completion.

IA Value	Level	Type	Description
`PageTemplateId`	0	String, Input, NoTrigger	The page template ID that will be placed in the `OutputPageDataXml` value and used for data extraction. Values: • Greater than "0": The specified value is a valid page template ID (plain non-xml text) that must be used for data extraction. This value overrides the template ID passed in the `InputPageDataXml` value or in the document type. • "0" (zero) or less than "–1": The template ID is determined from the `InputPageDataXml` value or the document type. In this case, if detected that a count of templates of the document type is less than a count of pages in the document, then: – for `PageTemplateId` equal to "0", the remaining pages of the document are skipped in Extraction; – for `PageTemplateId` less than

IA Value	Level	Type	Description
			"–1", the remaining pages of the document are assigned to the last template of the document type defined in the `UimDocumentType` IA Value. • "–1": The page must be skipped from data extraction. If `PageTemplateId` is not set (empty), it is considered to be "0" (zero). Consider showing a valid template ID if Extraction gets an empty input XML (`InputPageDataXml`), for instance, when Classification and Identification are not used in a CaptureFlow. Extraction will create `OutputPageDataXml` automatically, based on the specified template ID (`PageTemplateId`) and without `InputPageDataXml`.

IA Value	Level	Type	Description
PagesToProcess	1	String, Input	This value holds a comma-delimited string with page numbers to be extracted. For example: • "1": Extraction is performed on only page 1. • "3–5": Extraction is performed on pages 3, 4, and 5. • "L2": Extraction is performed on the last two pages. • "1,3–5,L2": Extraction is performed on pages 1, 3, 4, 5, and the last 2 pages. If this value is empty, then extraction is performed on all pages.
ReportTag1	Task	String, Input	Used for statistical data.
ReportTag2	Task	String, Input	Used for statistical data.
ReportTag3	Task	String, Input	Used for statistical data.
StepCustomValue	T	String, Output	Used as scripting parameters. If set to a non-empty value, overrides those values defined in the step setup.

IA Value	Level	Type	Description
`UimData`	1	String, Output	Information extracted from the document. Within the batch, every extracted document (level 1 node) holds the document type object with extracted data in this value. The document type object produced by the Extraction module serves as input to the Completion module that uses this object to identify the document type and index fields for the document.

IA Value	Level	Type	Description
`UimDataImportMode`	1	Long, Input, Output	The mode that defines or disallows additional data import to `UimData`. Input data is bound to the step regardless its index level. Allowed values are: • 0 - Do not bind additional data to `UimData`. • 1 - One-timely bind dynamic field data input to `UimData`. The data should be in values `InUimData_<field>` or `InUimData_<field>__<Index>`. • 2 – Always bind index values higher than the task level from the flattened output IA values back to `UimData`. The data should be in values named exactly as their target fields: `<field>`. When the field is an array (table column), all array items are expected, separated with a new line. Additionally, the "per row" array values are supported: `<field>_<row>`. In case IA values contain both "per array" and "per row" flattened

IA Value	Level	Type	Description
			data, the last one takes precedence. • 3 – The combination of modes 1 (“one-timely bind”) and 2 (“always bind”). After a value has been copied into the `UimData` object, `UimDataImportMode` is reset to “2”. In case of conflict between prefixed “one-time” and undecorated “always” values, the last one takes preference. To learn more about data flattening and passing data between IA values and UIM data, see *OpenText Intelligent Capture - Designer Guide (ECPCORE-CPD)*.

IA Value	Level	Type	Description
`UimDocumentType`	1	String, Input	The name of the document type. You can use this value to assign a particular document type to the Extraction step directly. Otherwise, Extraction will use the page template (identified and passed in by a preceding step, such as Classification) to determine the document type/index family for data extraction. When working with an IEE project, you must set the level 1 value to specify the document type to use. IEE does not work at a page level, so you do not have to specify level 0 values.

Glossary

MDF

Module Definition File

OCR

Optical Character Recognition

XML

Extensible Markup Language