Capture and Indexing

Installation and Setup Guide

Version: Foundation 23.1

Written by: Product Knowledge, R&D
Date: June 2023

Hyland

Documentation Notice

Table of Contents

Overview

The Capture and Indexing application allows you to capture and index documents into Perceptive Content. Documents are organized into batches which flow through a workflow process composed of different indexing and validation queues designed based on customer requirements. The custom designed workflow process allows for different roles within your organization to handle different tasks during the life of a batch.

Technical Specifications

For Capture and Indexing technical specifications and system requirements, refer to the Perceptive Experience Content Apps Technical Specifications guide.

Prepare to install Capture and Indexing

These instructions assume that you are installing Capture and Indexing for the first time or that you have no earlier versions of Capture and Indexing running on your computer.

The following steps outline the high-level procedures that you need to perform to install and configure Capture and Indexing, and the information you need to verify before installation.

Install Capture and Indexing

During the installation of Capture and Indexing, you perform the following procedures.

1. Install prerequisite applications. To work with Perceptive Content related Apps you must have Integration Server and Apache Tomcat installed.
2. Install Hyland Preferences Service. To store and manage client preferences, you must install this service. For more information, refer to the Hyland Preferences Service Installation Guide.
3. Download and copy files to the web server directory.
4. Enable SSL.
5. Disable HTTP pipelining.
6. Optional. Configure SSO or OIDC.
7. Enable Expires header in the web application server.

Download the Capture and Indexing files

To obtain Perceptive product installation files, contact the Hyland Software Technical Support group. For a list of Technical Support phone numbers, go to hyland.com/pswtscontact.

Install the Capture and Indexing files

To install Capture and Indexing, complete the following steps.

1. Download the application ZIP file to a temporary directory on your computer.
2. Create a subdirectory in the webapps directory of the web application server. You may name the subdirectory with any name, for example, **captureindexing**.

Note The default location for the webapps directory is **[drive:]\Program Files\Apache Software Foundation\[Tomcat Installation]\webapps**.

3. Extract the files to the subdirectory you just created.
4. Restart the web application server service.

Configure Integration Server location

For more information on configuring the integration server location, refer to the *Configure Integration Server location* section in the Perceptive Experience Content Installation Guide.

Enable SSL

For more information on enabling SSL, refer to the *Set Up SSL* section in the Perceptive Experience Content Installation Guide.

Disable HTTP pipelining

For more information on disabling HTTP pipelining, refer to the *Disable HTTP pipelining* section in the Perceptive Experience Content Installation Guide.

Configure Perceptive Experience for single sign-on

For more information on configuring Perceptive Experience for single sign-on, refer to the *Configure Perceptive Experience for single sign-on* section in the Perceptive Experience Content Installation Guide.

Configure Perceptive Experience for OpenID Connect authentication

For more information on configuring Perceptive Experience for OpenID Connect authentication, refer to the *Configure Perceptive Experience for OpenID Connect authentication* section in the Perceptive Experience Content Installation Guide.

About using cache headers with Perceptive Experience

For more information on using cache headers, refer to the *About using cache headers in Perceptive Experience* section in the Perceptive Experience Content Installation Guide.

Enable Expires header

Refer to the *Enable Expires header* section in the Perceptive Experience Content Installation Guide.

Start Capture and Indexing

To start Capture and Indexing, complete the following steps.

Prerequisite Ensure that you have the latest Internet browser installed.

1. Open your browser and enter the Capture and Indexing URL in the following format:

```
<server name>:<port number>/<subdirectory>
```

where **<server name>** and **<port number>** are placeholders for the name of the computer and the port number where you installed Capture and Indexing. The **<subdirectory>** is the location within the web application where you previously unzipped the contents of the Capture and Indexing zip file.

Note The default port number for Apache Tomcat web application server is **8080**.

2. Enter your user name and password.
3. Click **Connect**.

Configure Perceptive Content Server

Install Capture and Indexing iScript bundle

Capture and Indexing relies on a set of iScripts, **CI_RouteBatch.js** and **CI_SubmitBatch.js**, to handle routing batches after capture and final submission. These files are located in the **packages\lesrdl-content-capture-indexing\script** folder of the Capture & Indexing installation. Copy and paste both files into the **inserver\script** folder of the Perceptive Content Server and confirm the names of those scripts are correctly configured in the **config.json** configuration file.

Define users and groups

We recommend defining user groups for each of the different roles or locations you want to use in Capture and Indexing. For example, for users that only perform scanning or capturing of documents, create a capturing user group and only assign users to that group that perform that role. Once you create that group, assign that group to the privileges for capture source, capture profiles, and application plans used during capturing. For more information, refer to the **Create the application plan for capturing** section. Create a similar user group for users that only perform Indexing. Allow that group privileges to the drawers, document types, and application plans that are only used for indexing. For more information, refer to the **Create the application plan for indexing** section.

You can break user groups down by location. Each location user group can only access or index documents that belong to their location. The user group origins, as defined in the next section, helps to define and associate a particular user group to their origin or location.

User group origins

When configuring user groups in Management Console, include origins associated with the user group by defining them in the group description, in the format **Origin: <origin name>**. You can define multiple origins by creating a new line in the group description.

Origins serve multiple purposes, first of which is to identify where and what group created the batch. After batch creation, origin-based routing allows batches to be filtered into a workflow queue/process that only specific user groups have privileges to access.

Note If you have access to multiple origins, either through belonging to multiple groups or through multiple origins being defined on a group, you can select the desired origin for the batch at the time of capture.

Define the workflow process

In Capture and Indexing, captured documents are contained in a batch that is processed through a series of workflow queues. You can define the steps in the workflow process to fit the business process. In the simplest form, you can perform the QA, Indexing and Validation functions in one step. In larger organizations, you can split those functions into multiple steps and locations, allowing distribution of work by role and location. For more information about configuring workflow processes in Perceptive Content, refer to the **Workflow Designer** documentation. To define the workflow process in Perceptive Content, complete the following steps.

1. Create a new workflow process.
2. Add queues or super queues that represent each step in the process. For example, **QA**, **Index**, and **Validate**.
3. If the Capture and Indexing process is not centralized, define sub-queues within each super queue to represent a location or department in the organization.
4. Assign users or groups to each sub-queue.
5. Add a route between queues.

What is the batch capture process?

The three steps that typically take place in the batch capture process are quality assurance (QA), indexing, and validation. Each process is organized by the origin and location where the document is captured. You can model the process as a workflow with three super-queues. Each super-queue contains sub-queues that represent the capture location.

What is the record correction process?

Record correction process queues are organized by the given reason for the problem. You can model the record correction process as either a super-queue containing sub-queues that represent correction reasons, or as individual work queues for each reason.

Assign queue privileges

To assign queue privileges, complete the following steps.

1. To open queue properties, double-click the queue.
2. Under **Users**, add users or groups to the workflow.
3. Select the **Add**, **Process**, or **Remove** privilege for each user.

Assign sub-queue privileges

To assign sub-queue privileges, complete the following steps.

1. To open super-queue properties, double-click the super-queue.
2. Under **Users**, in the entry queue, select **Add** to allow the user to create new workflow items in the sub-queue.
3. Select **Remove** to allow the user to remove items from workflow.

What is queue validation?

During the indexing process, Capture and Indexing can validate indexing values to ensure accuracy. When validation is enabled for a queue, fields that violate validation rules are highlighted and marked with a validation error message. You can configure whether or not validation errors should be corrected before the batch can be routed forward to the next queue or submitted.

Validation rules

Validation rules in Capture and Indexing are defined in Application Plans, as well as automatically being defined based on the fields data type. In the **config.json** configuration file, you can set the

enableValidation rules for each queue or sub-queue. Data type validations are always enabled to prevent bad data from being sent to the server.

Required fields

You can mark an indexing field as a required field by configuring the Manual Index Plan for the document type. You must edit the field within the application plan. In the **Field Attributes** dialog, clear the **Allow blank** check box to mark the field as required.

Define the content model

In Perceptive Content, documents that are captured and indexed in the application reside in a folder and are processed in a workflow process. The folder represents a collection of documents captured in a single capture session, typically referred to as a batch. This folder contains metadata required for processing, as well as additional metadata acquired during the capture process.

As part of the design process, you define the content model for the captured documents by designating the drawer and folder type for the capture and indexing process, and by assigning required privileges.

Define a drawer

To define the drawer in Perceptive Content, complete the following steps.

1. Create a new drawer and provide a unique name.
2. In **Management Console**, assign the following Content Drawer privileges for users and groups:
    - Open
    - Search
    - Create or append
    - Move
    - Rename
    - Delete
    - Edit custom properties
    - Edit type
3. Assign the **Edit Keys** privileges for users and groups.

Define batch custom properties

Folder types that represent the batch are recommended to have the Batch Capture - Documents composite custom property to store the order of documents in the batch. To successfully define batch custom properties complete the following steps, and assign the custom properties to the folder type created below.

1. Create a new string custom property named **Batch Capture – Document**.
2. Create a composite custom property named **Batch Capture – Documents** that contains the **Batch Capture - Document** property.
3. Create a new string custom property named **Batch Capture – Origin**.

Define batch folder type

To create the batch folder type for storing captured documents with the required custom properties, complete the following steps.

1. Create a new folder type.
2. Assign a unique name for the folder type, for example, **Batch**.
3. Assign the **Folder Type > Use** folder type privilege for users and groups.
4. Add batch custom properties: **Batch Capture – Documents** and **Batch Capture – Origin**.

Define document types

To configure the indexing scheme by document type, you can associate a manual application plan to a document type list.

In Perceptive Content, you can define document types for content that is captured and indexed. You can also add custom properties to document types, as applicable.

What are unknown and unindexed document types?

You can designate certain document types as unknown or unindexed. You can add these document types to a document type list, such as unindexed documents. In the application configuration, set the applicationPlan attribute for this document type list to an empty string. For more information, refer to the unindexed document example.

Define document type lists

To add document types to a document type list, complete the following steps.

1. Create a new document type list.
2. Add one or more document type to **List Members**.

Create the application plan for capturing

The Capture and Indexing application uses Perceptive Content capture profiles and application plans for capturing content.

Create a source profile

To create a source profile, complete the following steps.

1. In **Management Console**, in the left pane, click **Capture**.
2. On the **Source Profile** tab, click **New...**.
3. On the **Source Profile Definition** page, select **Document** and then click **Next**.
4. Under **Name**, enter a name for the source profile.
5. Under **Source**, select **External Scanner** and click **Next**.
6. On the **Scanner Options** page, select the appropriate scan properties and click **Next**.
7. On the **Scanner Barcode Options** page, select the appropriate source profile barcode properties and click **Next**.

8. On the **Scanner Image Processing Options** page, select the appropriate source profile image properties and click **Next**.
9. Verify the configuration settings and click **Finish**.
10. Click **Modify** and, on the **Security** tab, assign the **Manage** privilege to the appropriate users or groups.

Create an external user application plan

1. In **Management Console**, in the left pane, click **Application Plans > External User**.
2. On the **Applications** tab, click **New...**.
3. In the **Plan Settings** dialog, on the **General** tab, select the **Document** content type.
4. In the **Name** field, enter a name for the application plan and click **OK**.
5. On the **Security** tab, add users or groups and allow them to **Link Documents** and **View**.
6. In the **Application Plan Designer**, create dictionary entries for any field that will be manually entered by the capturing user.

   Note “Origin” is a special dictionary entry that will map to the user selected Origin value during capture
7. Map all dictionary entries to folder or document field values.
8. Select **Folder Name**, click  , and assign the folder name to either a unique ID or a sequence number.
9. Select **Folder Type**, click  , select the appropriate folder type for the batch, and click **OK**.
10. Configure folder and document fields as desired.

Create a capture profile

To create a capture profile, complete the following steps.

1. In **Management Console**, in the left pane, click **Capture**.
2. On the **Capture Profile** tab, click **New > External Scanner**.
3. On the **Capture Profile Definition** page, select **Document** and then click **Next**.
4. Under **Name**, enter a name for the capture profile and click **Next**.
5. Select an existing source profile and click **Next**.
6. Select an application plan and click **Next**.
7. On the **Capture Profile Options** page, click **Next**.
8. On the **Capture Profile Definition** page, review the configuration settings and click **Finish**.
9. Assign the **Use** privilege to the appropriate users or groups.

Create the application plan for indexing

For each indexing scheme, you must define a manual application plan in Perceptive Content. To create a manual application plan, complete the following steps.

1. Create a new manual application plan that contains a unique name.

2. Add the **Link Documents** privilege to users and groups that will use the application plan.
3. Open the **Manual Application Plan Designer** for the newly created application plan and navigate past the **Drawer** section.

   Note Within the application, the documents being indexed are contained in the batch container (folder).
4. Perform one of the following actions to map one or more document index keys (Field1-Field5) or custom properties:
   - Map document name to unique ID.
   - Select either **User Entry** or **Predefined List** for fields that are completed by users.
   - Optionally enter a label for fields completed by a user.
   - Select the **Allow** checkbox to mark the field as required, or clear the **Allow** checkbox to mark the field as optional.
   - Map the document type to the applicable document type list.

Configure document indexing

After defining document type lists and manual application plans in Management Console, you must associate them within the **config.json** configuration file. To configure document, complete the following steps.

1. Add the name of the document type list to the **documentTypeList** object.
2. Enter the name of the manual application plan to the **applicationPlan** attribute.

Configure print privileges

To print documents in Capture and Indexing, users must have print privileges. To configure print privileges, complete the following steps.

1. To allow printing all documents in a drawer, under drawer privileges for the user or group enable **Viewer > Print Document**.
2. To allow printing all documents of a document type, under the document type's privileges for the user or group enable **Viewer > Print Document**.

Configure Capture and Indexing

Capture and Indexing application configurations can be found in the following configuration files:

- **config.json** – The default location is **[drive:]\Program Files\Apache Software Foundation\[Tomcat Installation]\webapps\[Capture and Indexing Installation]**.
- **batchHeader.html** – The default location is **lesrdl-content-capture-indexing\src\config\templates\batchHeader.html**.
- **keyboardShortcutsConfig.json** – The default location is **lesrdl-content-capture-indexing\src\config\keyboardShortcutsConfig.json**.

Batch index header

In Capture and Indexing, the batch index header contains information about the batch that is currently open. You can configure the header for each workflow process.

The information displayed in the header is defined using a Handlebars template. The header can be customized to add the metadata for a batch. The batch object passed to the template contains a combination of workflow item and folder information. The following attributes may be useful in the header.

Note The default header template,batchHeader.html, displays the name, queue, description and the content for the batch.

Attribute	Description	Handlebar Expression
name	Name of the batch.	{{name}}
origin	Scanned origin.	{{Batch Capture - Origin}}
creationDate	The date and time the batch was created	{{date creationDate datetime='medium'}}
creationUserName	The user who created the batch	{{creationUserName}}
numDocs	The number of documents in the batch	{{numDocs}}
numPages	The number of pages in the batch	{{numPages}}
priority	The priority of the workflow item	{{priority}}
workflowQueueName	The full name of the queue	{{queue.name}}
workflowQueueStartDate	The start date and time for the workflow item	{{date workflowQueueStartDate datetime='medium'}}

Add custom properties to the header

You must provide a label and name for each folder custom property you want to view in the batch index header. To add custom properties to the header, complete the following steps.

1. For each string custom property you want to make editable, add the editable class to properties-item div.

   Note Editable string custom properties can contain up to 128 characters.

2. Optional. If the custom property contains a long string, modify the width as shown in the following example.

Example

```
<div class="properties-item editable">
     <label class="property-label">Description</label>
     {{#with [Batch Capture - Description]}}
        <div prop-id={{id}} class="property-value" style="width:300px">{{value}}</div>
     {{/with}}
 </div>
```

Server actions (iScripts)

Route batch

When the system completes the capture process, the system invokes an iScript specified in the **config.json** configuration file under routeBatch. The specified file must exist in the script directory of the content server.

Example

```
"routeBatch": {
    "serverAction": <iScript name>
}
```

Submit batch

When the user submits a batch after completing indexing and validation, the system invokes an iScript that is specified in the **config.json** configuration file.

The following is an example of an iScript specified in the the **config.json** configuration file configuration file.

Note The specified file must exist in the script directory of the content server.

```
"submitBatch": {
       "serverAction": <iScript name>
 }
```

If the system cannot find the specified file, the batch cannot be submitted and the user is notified of the issue.

Configure capture

You can include multiple capture sources in the application by specifying more than one capture source module. By default, the application includes scanner and file capture sources. When multiple sources are registered, a dialog prompts the user to select one of the sources. To bypass this dialog, refer to options in the captureSourceMapping section.

Configure barcode processing

During the capture process, indexing values can be read from barcodes. To assign indexing values from barcodes, complete the following steps.

1. In **Management Console**, in the **External User Application Plan**, assign one or more indexing keys to the Barcode source.
2. In the **Value** field, enter a unique name for the barcode.

3. In the **config.json** configuration file found in the web application root, navigate to the **lesrdl-content-capture-indexing** section and perform the following actions.
4. In the **barcodeSeparatorMapping** section, specify one or more regular expression patterns used to identify separator pages.
5. In the **barcodeNameMapping** section, specify one or more barcode name mapping.

Appendix A: config.json properties

The **config.json** configuration file contains the main configuration for the Capture and Indexing package and is located at **<application-root>/config.json**.

lesrdl-content-properties

This section contains the capture and indexing configurations that are part of the **lesrdl-content-properties** section of the **config.json** configuration file.

typeLabel

The typeLabel attribute is an optional setting to override the label for the type in the property pane.

Note If you set the attribute, it will override the view type column label for content that uses view columns.

Example

The following example

```
"lesrdl-content-properties": {
       "typeLabel" : "Forms or Widget"
       …
}
```

lesrdl-content-capture-indexing

This section contains all configurations that are part of the **lesrdl-content-capture-indexing** section of the **config.json** configuration file.

CI-split-above-action

The CI-split-above-action configures visibility and priority (sort order) for the split above action toolbar group. You can hide the entire group or you can configure each individual option.

Options

Attribute	Data type	Description
priority	Integer	Allows you to indicate the location of an action or group of actions on the toolbar relative to other actions or groups.

Attribute	Data type	Description
isVisible	Boolean	Allows you to configure the visibility of actions on the toolbar. If you set isVisible to true, the system displays the action on the toolbar. If you set it to false, the system does not display the action. The default is true.

Example

The following example

```
"CI-split-above-action": {
     "priority": 60,
     "isVisible": true,
     "CI-split-and-clear": {
          "priority": 1,
          isVisible": true
     },
     "CI-split-keep-type": {
          "priority": 2,
          "isVisible": true
     },
     "CI-split-keep-index": {
          "priority": 3,
          "isVisible": true
     },
     "CI-split-keep-type-index": {
          "priority": 4,
          "isVisible": true
     }
},
```

dateFormat

The dateFormat attribute defines the default date format for the application. The value can be a string, or a JSON object that describes formatting options. Strings that are specified for the dateFormat attribute are passed as a raw pattern. If a string is not specified, the default format is “MM/dd/yyyy”. Note that the default format supports the Unicode standard and is case-sensitive.

The following table lists a summary of example format strings and their results.

Format	Example
yyyy-M-d	2021-6-23
M/d/yyyy	6/23/2021

Format	Example
d/M/yyyy	23/6/2021
dd.MM.yyyy	23.06.2021
dd/MM/yyyy	23/06/2021
yyyy//M/d	2021/6/23
yyyy-MM-dd	2021-06-23
yyyyMMddTHH:mmzzz	20210623T13:22-0500

The following example defines the date in a short format, such as 07/04/15.

```
"dateFormat": { "date":"short" }
```

The following example defines the date in a date and time format, such as 07/04/2015 13:21.

```
"dateFormat": "MM/dd/yyyy HH:mm"
```

defaultDocumentType

The defaultDocumentType object defines the default document type to use when indexing if document type is cleared.

Options

Attribute	Data type	Description
<document type>	String	The document type name.

Example

The following example defines the default document type of unindexed.

```
"defaultDocumentType": "Unindexed document type"
```

documentTypeList

The documentTypeList object contains one or more document type list objects. Each document type list object is keyed by the document type list name in Perceptive Content. In Capture and Indexing

application, indexing rules can be defined for each document type via a document type list. For each class of document type, you can define an indexing application plan, database mappings, and one or more contexts. These contexts provide additional contextual metadata for aiding the user with indexing and validation.

Options

Attribute	Data type	Description
applicationPlan	String	A manual application name that defines the indexing scheme for a document type.

Examples

Unindexed documents

You can configure specific document types to contain no indexing information, as shown in the following example.

```
"Unindexed Documents" : {
 "applicationPlan" : ""
 }
```

Document with manual indexing

You can require that users manually provide values for indexing fields without external database lookup, as shown in the following example. This option does not require database mappings for indexing fields.

```
"Manager Review Documents": {
 "applicationPlan": "Manager Review Indexing"
 }
```

saveWithPrompt

The saveWithPrompt option defines whether the system prompts users to save a modified document when switching to a new document or executing actions from the toolbar. If set to false, the system saves documents without prompting the user, with the exception of the Back action or any action that navigates the user away from the currently opened viewer. The default is false.

Example

```
"saveWithPrompt": true
```

applicationPlan

The applicationPlan object contains one or more application plan objects and is used to configure additional data lookups during capture and indexing.

A typical use case for configuring database lookup during capture is when the captured documents contain one or more barcodes that provide initial values for indexing fields. The barcode values can then be used as inputs to a database search that can provide values for the remaining indexing fields.

Capture

Contains mappings of the application plans. Each application plan configuration contains a set of validation rules and capture lookups to perform when capturing with this specified application plan.

Options

Attribute	Data type	Description
barcodeDocSplitMode	Boolean	Creates a new document on reading a barcode, subsequent pages are added to the same document until a new barcode is read
originDrawerMapping	Object	An object containing a mapping of Origin to Drawer. Allows capturing to a specific drawer based on the selected Origin.
validation	Validation	A set of validation rules, keyed by propertyId
lookups	Lookups	An array of capture lookups, executed in order

Validation

A set of validation rules, keyed by propertyId.

Attribute	Data type	Description
flagUpdate	Boolean	A set of validation rules, keyed by propertyId

Lookups

An array of capture lookups to execute in order. Each capture lookup specifies which property values can be used as search parameters to the webservice lookup. The updates section also allows index keys or custom properties to be updated with values from data returned by the webservice lookup.

Attribute	Data type	Description
name	String	Name of the data provider lookup to execute

Attribute	Data type	Description
parameters	Object	Contains parameters to pass to the lookup
<parameters.name>	Key	Parameter name matching the URL or query parameter for the data provider lookup
parameters.path	String	PropertyId for the parameter value (field1, field2, custom property name)
parameters.isRequired	Boolean	Flag indicating if the value is required to execute the lookup
updates	Object	Contains all updates to apply to the document, using the response from the webservice
<updates.propertyId>	Key	PropertyId to update (field1, field2, custom property name)
updates.sourceType	Enum	RECORD or LITERAL
updates.source	String	Path in response object or the exact value to set if sourceType is LITERAL
updates.default	Any	Default value if no value is set by the webservice

Example

In the example below, the “HIM Capture” application plan populates `field3` from a CSN barcode.

This CSN value in `field3` is then used to look up an encounter and then populate `field1` using the MRN record from the encounter lookup.

```
“applicationPlan” : {
    “capture”: {
        “HIM Capture” : {
            "originDrawerMapping": {
            "CI Origin Name 1": "CI Drawer Name 1"
        },
```

```
            "lookups": [{
                "name": "lookupEncounterByCSN",
                "parameters": {
                    "csn": {
                    "path": "field3"
                    }
                },
                "updates": {
                    "field1": {
                        "sourceType": "RECORD",
                        "source": "mrn"
                    }
                }
            },
    "indexing": {
        "HIM Indexing" : {

        }
    }
}
```

indexing

Contains mappings of the application plans. The application plan mapping determines which capture lookup and update will be triggered when the configured property (PropertyId) or properties are modified. The mapping also determines which data columns from the lookup or advanced search will be displayed to the user if multiple entries are returned by the lookup. The advanced search section allows configuration of a lookup allowing the user to search for specific records. The advanced search also has its own update section which allows properties to be updated based on the record selected during the search.

Attribute	Data type	Description
lookup	Lookup	Capture lookup
columns	Column	An array of columns returned by webservice displayed to the user
advancedSearch	AdvancedSearch	An array of capture lookups, executed in order

Lookup

A single capture lookup. See the lookup table in the capture applicationPlan section for the definition.

Columns

A subset of data returned from the lookup to display when multiple records match. The source maps to the attribute name in the lookup record to display. Label allows for alternate text for the field. If label is not specified, the source attribute name will be used.

Attribute	Data type	Description
source	String	Attribute name of the record to display
label	String	Alternate text to display for the column
format	String/Object	Set of formatting rules to apply to the value of the displayed column.

Advanced search

Allow the user to search for a specific record by the specified lookup and populate property values based on the record selected.

Attribute	Data type	Description
lookup	SearchLookup	Lookup used for the search

Search lookup

The advanced search lookup format is similar to the capture lookup, however, it also allows an additional searchParameters section which can specify an id which is used for the name and path and specifying the data type for the search parameter so that the data values are compared correctly for different data objects. The property updates are performed after a record is selected from the search.

Search parameters

Attribute	Data type	Description
searchParameters	SearchParameter	An array of search parameters used by the lookup

Search parameter

Attribute	Data type	Description
id	String	Path for the lookup

Attribute	Data type	Description
name	String	Specifies the search constraint name displayed when source is configured and search constraint is pre-populated.
type	String	Search data type, DATE, NUMBER or STRING. STRING is the default if not specified
source	String	Specifies the attribute name used to set the initial search constraint value.
isRequired	Boolean	Flag indicating if the value is required to execute the lookup

Example

In the example below, when the DOB property is changed for a document type belonging to the “HIM Indexing” application plan, a lookup is performed using the value entered into the DOB property. If there are multiple records matching that DOB, the columns configured are displayed to the user. When the user selects a record, `field1` is updated with the MRN value.

The user can also search for a record by clicking the search button on the DOB property. The search would also use the lookup, lookupPatientByDOB. The search parameters require that the user create a search condition dob which is a DATE which will be passed to the lookup. When the search if performed and the user selects a matching record, `field1` is updated with the MRN value, `field2` is updated with the Facility Id and the DOB property is updated the DOB value.

```
"applicationPlan" : {
    "capture": {
...
    "indexing": {
       "HIM Indexing": {
        "DOB": {
            "lookup": "lookupPatientByDOB",
            "parameters": {
                "dob": {
                "path": "DOB"
                }
            },
            "columns": {
                {
                    "source": "mrn",
                    "label": "MRN"
                },
                          {
                    "source": "mrn",
                    "label": "MRN"
                },
                {
```

```
                    "source": "facilityId",
                    "label": "Facility ID"
                },
                {
                    "source": "firstName",
                    "label": "First Name"
                },
                {
                    "source": "lastName",
                    "label": "Last Name"
                },
                {
                    "source": "dob",
                    "label": "Date of Birth"
                    "format": "MM/dd/yyyy"
                },
                {
                    "source": "balance",
                    "label": "Balance"
                           "format": {
                           "minimumFractionDigits": 2,
                           "useGrouping": true
                           }
                },
            },
            "updates":
                    {
                "field1": {
                    "sourceType": "RECORD",
                    "source": "mrn"
                }
            },
            "advancedSearch": {
                "lookupPatientByDOB": {
                    "name": "Patient By DOB",
                    "searchParameters": {
                        {
                            "id": "dob",
                            "type": "DATE",
                            "isRequired": true
                        }
                    },
                    "updates": {
                        "field1": {
                            "sourceType": "RECORD",
                            "source": "mrn"
                        },
                        "field2": {
                            "sourceType": "RECORD",
                            "source": "facilityId"
                        },
                        "DOB": {
                            "sourceType": "RECORD",
                            "source": "dob"
                        }
                    }
                }
            }
        }
    }
```

```
}
```

captureSourceMapping

The captureSourceMapping array defines mapping rules for a capture profile name to a capture source. If the rule matches the selected capture profile and a registered capture source, then the capture source is automatically selected. If no matching rule is found and there are multiple sources registered in the system, then the user is prompted to select a source when capturing.

Options

captureSourceMapping can contain one or more rules to map a capture profile name to a capture source.

Example

In the example below, capture profile names containing “scan” will use the scanner source, while profile names containing the word “file” will use the file source. Any capture profile name that does not match any of the rules will prompt the user to select a capture source.

```
"captureSourceMapping": [
"/scan/i:/lesrdl-framework-capturesource-scanner/",
"/\bfile/i:/lesrdl-framework-capturesource-file/"]
```

captureProfileMapping

The captureProfileMapping array defines rules for filtering capture profiles. Only capture profile names matching one of the regular expressions will be displayed in the application.

Options

The filters are regular expressions that are matched to capture profile names. Multiple regular expressions can be defined, a capture profile displays when its name matches any of the regular expressions.

Example

In the example below, only capture profile names that start with “CI” or contain “HIM” are displayed. To display all capture profile names use “/.*/”.

```
"captureProfileMapping": [
"/^CI.*/", "HIM” ]
```

originDrawerMapping

The originDrawerMapping object defines mappings for origin to drawer. It allows capturing to a specific drawer based on the selected origin. The system uses this mapping if a match is not found under the application plan’s originDrawerMapping. The origin can be a partial match.

Example

In the example below,

```
"originDrawerMapping" : {
       "partial origin" : "Target Drawer"
}
```

originMapping

The originMapping array defines rules for filtering origins. Only origin names matching one of the regular expressions will be displayed in the application.

Options

originMapping contains filter rules for each capture mode. The filter rules are regular expressions that are matched to origin names. If multiple regular expressions are defined, an origin displays when its name matches any of the regular expressions.

Example

In the example below, only origins with names starting with "IC" are displayed.

```
"originMapping" : ["^IC"]
```

routeBatch

routeBatch serverAction defines the name of the iScript to route a batch after the capture process. The iScript expects the batchFolderId single input parameter, and returns four output parameters.

Options

Parameter name	Parameter type
batchFolderId	Input
“SUCCESS: <success message>” or “ERROR: <error message>”	Output
workflowItemId	Output
workflowQueueId	Output
workflowQueueName	Output

Example

The following example defines the iScipt, CI_RouteBatchFolder.js, used to route the batch after the capture process.

```
"routeBatch" : {
       "serverAction" : "CI_RouteBatchFolder.js"
}
```

submitBatch

submitBatch serverAction defines the iScript to be invoked after the batch is completely indexed and submitted for processing. The iScript expects three input parameters, and returns two output parameters.

Options

Parameter name	Parameter type
workflowItemId	Input
batchFolderId	Input
username	Input
status: SUCCESS or ERROR	Output
error message	Output

Example

The following example defines the iScipt, CI_SubmitBatch.js, to use after the batch is indexed and submitted for processing.

```
"submitBatch" : {
       "serverAction" : "CI_SubmitBatch.js"
}
```

workflowProcess

The workflowProcess object defines one or more workflow processes that are displayed in Capture and Indexing module. Currently, only folder-centric workflow processes are supported.

Options

Configurations shared between workflowProcess and workflowQueue are overwritten by workflowQueue.

Note Perceptive Content formats the sub-queue name as "`<sub queue> (<super queue>)`" which you can see in the Workflow Designer super-queue properties dialog.

Attribute	Data type	Description
batchHeaderTemplate	String	The name of banner template for indexing screen. Template must exist in the config/templates folder.
enableValidation	Boolean	When set to true, validation is performed on load and when index values change. Invalid fields are highlighted. The default value is true.

Attribute	Data type	Description
properties	Object	Map of properties to strings
properties[Key ID]	string	Forces this doc key to be enabled/disabled/hidden
properties.customProperties[custom property name]	string	Forces this custom property to be enabled/disabled/hidden

Example

The following example defines the workflow process configuration.

```
"workflowProcess": {
    "Batch Capture": {
        "batchHeaderTemplate": "batchHeader.template",
        "enableValidation": false,
        "properties": {
            "field5": "hidden",
            "customProperties": {
                "Cost": "disabled"
            }
        }
    }
}
```

workflowQueue

The workflowQueue object defines one or more queue settings. This configuration can be specified at the super-queue and sub-queue level. Sub-queue configuration overrides super-queue's.

Options

For each queue (super queue or sub queue), define whether validation should be enabled, and set property editability/visibility.

Note Perceptive Content formats the sub-queue name as "`<sub queue> (<super queue>)`" which you can see in the Workflow Designer super-queue properties dialog.

Attribute	Data type	Description
enableValidation	Boolean	When set to true, validation is performed on load and when index values change. Invalid fields are highlighted. The default value is true.
properties	Object	Map of properties to strings

Attribute	Data type	Description
properties[Key ID]	String	Forces this doc key to be enabled/disabled/hidden
properties.customProperties[custom property name]	String	Forces this custom property to be enabled/disabled/hidden

Examples

In the example below, settings are defined at the super-queue and sub-queue levels and the following occurs:

* All `1 QA` queues except `HIM` have validation not enabled

* All `1 QA` queues have field5 hidden

* All `1 QA` queues except `HIM` have the "Cost" custom property disabled

* All `3 Validate` queues have document type disabled

```
"workflowQueue" : {
    "1 QA" : {
        "enableValidation" : false,
        "properties": {
            "field5": "hidden",
            "customProperties": {
                "Cost": "disabled"
            }
        }
    },
    "2 Index" : {
        "enableValidation" : true
    },
    "3 Validate" : {
        "enableValidation" : true,
        "properties": {
            "type": "disabled"
        }
    },
    "HIM (1 QA)": {
        "enableValidation": true,
        "properties": {
            "customProperties": {
                "Cost": "enabled"
            }
        }
    }
}
```

print

The print object defines the print options used to print a document.

Options

For each document, the following attributes determine what printing functionality is enabled or enforced.

Attribute	Data type	Description
enableAnnotations	Boolean	When set to true, this attribute determines if annotations display and can be printed. The default is true.
pagesize	String	Determines the size of the print. Print sizes can be A4, A3, or letter. The default value is letter.
maxWidth	String	The maximum width, in pixels, for printed pages.
maxHeight	String	The maximum height, in pixels, for printed pages.

Example

The following example defines the appropriate print options.

```
"print" : {
      "enableAnnotations" : true,
      "pageSize" : "letter",
      "maxWidth" : "",
      "maxHeight" : "",
}
```

workflow

The workflow object defines the options used in the workflow process.

Options

Attribute	Data Type	Description
openNextItem	Boolean	Defines the next action after submitting or routing a batch. If set to true, the next available batch in the queue is fetched automatically and opened for indexing. If not set or set to false, the current queue is displayed. The default is false.

Example

The following example defines the appropriate workflow options.

```
"workflow" : {
         "openNextItem": true
     }
```

batch

The batch object defines the options used in the indexing screen to enable or disable the delete pages confirmation prompt.

Options

Attribute	Data Type	Description
confirmPageDeletion	Boolean	If the attribute is set to true, a delete pages confirmation dialog displays after the user triggers the delete action. If the attribute is set to false, a delete pages confirmation dialog does not display after the user triggers the delete action. The default is false.

Example

The following example configures the delete page confirmation dialog to display.

```
"batch": {
         "confirmPageDeletion": true
     }
```

lesrdl-content-capture-indexing-dataprovider

This section contains all configurations that are part of the lesrdl-content-capture-indexing-dataprovider section of the **config.json** configuration file.

authentication

A collection of Authentication objects, keyed by a unique authentication profile name.

Options

Attribute	Data type	Description
type	Enum	Type of authentication. Supported values: OIDC
profileId	String	(OIDC) Defines the profileId to use configured in the lesrdl-content-oidc-client section of the **config.json** configuration file.

Example

The following is an example of the authentication configuration.

```
"authentication" : {
    "defaultOIDC" : {
        "type" : OIDC,
        "profileId": "default"
    },
}
```

lookups

A collection of lookupConfigurations.

Options

Attribute	Data type	Description
type	Enum	WEBSERVICE, SCRIPT Defaults to WEBSERVICE.
scriptName	String	Name of the iScript to execute for SCRIPT lookups.
inputParameters	Parameter	Object containing a set of input parameters to be passed to the iScript lookup for SCRIPT lookups.
url	String	URL template.
method	Enum	GET, POST – HTTP method
authentication	String	Name of the authentication profile to use, configured in the authentication section.
urlParameters	Parameter	Object containing parameters injected into the url string. All urlParameters are considered required.
queryParameters	Parameter	Object containing a set of input parameters to be appended to the url as query parameters. All query parameters are considered optional unless specified otherwise.
response	Response	Object containing values to pull from the response json.

Parameters

Attribute	Data type	Description
source	String	Name of the parameter passed in to the request.
type	Enum	STRING, DATE, NUMBER, FLAG
isRequired	Boolean	Indicates if the parameter is required to have a value.

Response

Attribute	Data type	Description
dataRoot	String	Path to the root element in the json response. If the root element is not an array, it will be treated as a single response object.
recordStructure	RecordStructure	Object containing the structure of the response json.

RecordStructure

Attribute	Data type	Description
path	String	Path to the value in the response, relative to dataRoot.
type	Enum	STRING, DATE, NUMBER, BOOLEAN. Defaults to STRING.

Examples

The following is an example of the lookups configuration.

```
{
    "lookups": {
        "getVendorById": {
            "url": "/api/vendor/<%= vendorId %>",
            "method": "GET",
            "authentication": "defaultOIDC",
            "urlParameters": {
                "vendorId": {
```

```
                    "source": "vendorId",
                    "type": "STRING"
                }
            },
            "queryParameters": {
                "companyCode": {
                    "source": "companyCode",
                    "type": "STRING"
                }
            },
            "response": {
                "dataRoot": "results",
                "recordStructure": {
                    "vendorName": {
                        "path": "vendorName",
                        "type": "STRING"
                    },
                    "address1": {
                        "path": "address1",
                        "type": "STRING"
                    },
                    "creationDate": {
                        "path": "created",
                        "type": "DATE",
                        "format": "MM-dd-yyyy"
                    },
                    "payImmediately": {
                        "path": "payImmediately",
                        "type": "BOOLEAN"
                    }
                }
            }
        },
        "getVendorSearch": {
            "type": "SCRIPT",
            "scriptName": "VendorLookup.js",
            "inputParameters": {
                "lookupType": {
                    "source": "VendorSearch",
                    "type": "LITERAL"
                },
                "vendorName": {
                    "source": "vendorName",
                    "type": "STRING"
                }
            },
            "response": {
                "dataRoot": "results",
                "recordStructure": {
                    "vendorName": {
                        "path": "vendorName",
                        "type": "STRING"
                    },
                    "address1": {
                        "path": "address1",
                        "type": "STRING"
                    },
                    "creationDate": {
                        "path": "created",
                        "type": "DATE",
```

```
                        "format": "MM-dd-yyyy"
                    },
                    "payImmediately": {
                        "path": "payImmediately",
                        "type": "BOOLEAN"
                    }
                }
            }
        }
    }
}
```

The following is an example snippet from a lookup iScript.

```
try {
    var inputParams = getInputParams();
    var inputData = JSON.fromText(inputParams[0]);
    var outputParams = [];

    switch (inputData.lookupType) {
        case 'VendorSearch':
            // This is where we would query to an external service using
            // values from the input params (i.e. inputData.vendorName)
            //
            // Success state sent back to C&I requires a status (SUCCESS)
            // and a JSON.stringified object/array.
            outputParams[0] = 'SUCCESS';
            outputParams[1] = JSON.toText({
                results: [
                    {
                        vendorName: 'ACME Corp',
                        address1: '1234 Main Street',
                        created: '12-31-2021',
                        payImmediately: true
                    },
                    {
                        vendorName: 'Express Corp',
                        address1: '4500 N 1st Street',
                        created: '09-15-2021',
                        payImmediately: true
                    }
                ]
            });
            break;
    }
}
catch (exception) {
    // Error state sent back to C&I requires a status (ERROR) and an
    // optional exception message
    outputParams[0] = 'ERROR';
    outputParams[1] = exception;
}

setOutputParams(outputParams);
```

Appendix B: keyboardShortcutsConfig.json

The **keyboardShortcutsConfig.json** configuration file contains the keyboard shortcuts configuration for Capture and Indexing application. The configuration file for keyboard shortcuts can be found at l**esrdl-content-capture-indexing/src/config/keyboardShortcutsConfig.json**.

Schema

```
{    <shortcut name>: <shortcut>,    ...}
```

Supported keys

The following is a list of modifiers that can be used as shortcuts.

- Shift
- Alt or option
- Control, CTRL, or ^
- Command

Special keys

The following special keys can be used for shortcuts.

- Backspace
- Tab
- Clear
- Enter
- Return
- ESC
- Escape
- Space
- Up
- Down
- Left
- Right
- Home
- End
- Page-up
- Page-down
- DEL
- Delete

- Keys F1 through F19

The following numeric pad keys can be used as shortcuts.

- Equals
- Minus
- Divide
- Multiply
- Subtract
- Add
- Numbers 0 through 9

Defined shortcuts

The following shortcut keys are defined in the application.

Note This configuration is optimized for the Chrome browser. In other browsers, some shortcuts may conflict with the browser or local system shortcuts. Ensure that keyboard shortcuts are tested thoroughly and customized for each implementation.

Name	Description	Default
CAPTURE	Initiates a capture.	alt+ctrl+n
DELETE_BATCH	Deletes the batch.	delete
CLOSE_BATCH	Close a batch.	alt+ctrl+backspace
SUBMIT_BATCH	Submit a batch.	alt+ctrl+g
COPY_CLEAR	Copy selected pages into a new document. Sets new document's document type to default and clears its index keys and custom properties.	alt+ctrl+d
COPY_KEEP_DOC_TYPE	Copy selected pages into a new document. Keeps selected document's document type and clears other document keys and custom properties.	alt+ctrl+shift+t
COPY_KEEP_INDEX	Copy selected pages into a new document. Keeps selected document's index keys, sets its document type to default and clears its custom properties.	alt+ctrl+shift+i

Name	Description	Default
COPY_KEEP_TYPE_INDEX	Copy selected pages into a new document. Keeps selected document's document type, keys, and custom properties.	alt+ctrl+shift+d
DELETE_PAGE	Delete page. Deletes document if all pages are selected	delete
PRINT	Prints the current document.	alt+ctrl+p
ROTATE_180	Rotates a page 180 degrees.	alt+shift+l
ROTATE_LEFT	Rotates a page counter-clockwise.	alt+shift+,
ROTATE_RIGHT	Rotates a page clockwise.	alt+shift+.
SPLIT_CLEAR	Splits selected pages into its own document. Sets the document type of the selected document to default, clears index keys, and clears custom properties.	alt+shift+[
SPLIT_DOCUMENT	Splits selected pages into their own, separate document.	alt+ctrl+s
SPLIT_KEEP_DOC_TYPE	Splits selected pages into its own document. Does not modify selected content.	alt+shift+]
SPLIT_KEEP_INDEX	Splits selected pages into its own document. Keeps the index keys for the selected document, sets the document type to default, and clears custom properties.	alt+shift+=
SPLIT_KEEP_TYPE_INDEX	Split selected pages into its own document. Keeps selected document's document type, keys, and custom properties.	alt+shift+-
ANNOTATION_ARROW	Create an arrow annotation.	alt+ctrl+shift+1

Name	Description	Default
ANNOTATION_ARROW	Create an arrow annotation.	alt+ctrl+shift+1
ANNOTATION_HIGHLIGHT	Create a highlight annotation.	alt+ctrl+shift+3
ANNOTATION_LINE	Create a line annotation.	alt+ctrl+shift+4
ANNOTATION_OVAL	Create an oval annotation.	alt+ctrl+shift+5
ANNOTATION_PEN	Create a pen annotation.	alt+ctrl+shift+6
ANNOTATION_PICTURE_STAMP	Create a picture stamp annotation.	alt+ctrl+shift+7
ANNOTATION_RECTANGLE	Create a rectangle annotation.	alt+ctrl+shift+8
ANNOTATION_STICKY_NOTE	Create a sticky note annotation.	alt+ctrl+shift+9
ANNOTATION_TEXT	Create a text annotation.	alt+ctrl+shift+0
ANNOTATION_TEXT_STAMP	Create a text stamp annotation.	alt+ctrl+shift+-
ANNOTATION_URL	Create a url annotation.	alt+ctrl+shift+=
FOCUS_PROPERTIES	Set the focus to the properties pane.	alt+ctrl+shift+p
NEXT_DOC	Selects the next document.	alt+ctrl+.
NEXT_PAGE	Selects the next page.	alt+ctrl+/
PREVIOUS_DOC	Selects the previous document.	alt+ctrl+,
PREVIOUS_PAGE	Selects the previous page.	alt+ctrl+m