Oracle® Fusion Middleware

Developing Scripts for Oracle WebCenter Enterprise Capture

12c (12.2.1.4.0)
E95386-02
May 2021

Oracle Fusion Middleware Developing Scripts for Oracle WebCenter Enterprise Capture, 12c (12.2.1.4.0)

E95386-02

Primary Author: Puneeta Bharani

Contributors: Oracle WebCenter development, product management, and quality assurance teams

Contents

4 Creating Recognition Processor Scripts

5 Creating Import Processor Scripts

6 Creating Document Conversion Processor Scripts

7 Creating Commit Processor Scripts

8 Working with Common Capture Classes

A Keycodes

Preface

This guide contains information to develop scripts to customize Oracle WebCenter Enterprise Capture components.

Audience

This guide is intended for developers responsible for customizing Oracle WebCenter Enterprise Capture functionality.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

The complete Oracle WebCenter Content documentation set is available from the Oracle Help Center at http://www.oracle.com/pls/topic/lookup?ctx=fmw122140&id=wcc-books.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Convention	Meaning
`getter/setter pattern`	Getter/setter pattern indicates properties that uses this pattern. For example, to access the title property of the Document class from code use the following: `title = document.getTitle();`

1

Introduction to Developing Scripts with Oracle WebCenter Enterprise Capture

This chapter provides an introduction to developing scripts for Oracle WebCenter Enterprise Capture.
A script is a custom piece of code consumed by the Capture client or a batch processor (Import or Recognition) that allows you to customize functionality beyond existing configuration settings. For example, you might incorporate a script to change the first letter of a name to uppercase or to use a proprietary calculation to validate an account number used in a transaction.

For scripting, Capture uses the JavaScript script engine included with the Java Runtime Environment. Refer to the Oracle Java documentation for more information.

Scripts can be incorporated in the following Capture components:

- Client

  Client Scripts are JavaScript modules that enable you to customize the behavior of certain client events. To use one or more scripts in a client profile, a workspace manager selects and orders them in an extension profile.

- Recognition Processor

  Recognition Processor scripts allow you to customize the behavior of certain recognition job events.

- Import Processor

  Import Processor scripts allow you to customize the behavior of certain import job events.

  For more information on incorporating scripts in Capture, see *Managing Oracle WebCenter Enterprise Capture*.

Developing Scripts with WebCenter Enterprise Capture

The following are the main steps for developing and incorporating scripts in Capture components:

1. For each Capture component, write the JavaScript using the events and classes. For more information, refer to the following component's chapter:
   - Creating Client Scripts
   - Creating Recognition Processor Scripts
   - Creating Import Processor Scripts
2. On the Advanced tab of a selected workspace in the WebCenter Enterprise Capture Workspace Console, a workspace manager adds the script by identifying its component type and loading the script file.

   For more information, see *Managing Oracle WebCenter Enterprise Capture*.

3. In a client profile or an import or Recognition Processor job, the workspace manager then selects the script for use.

   Note that workspace managers can incorporate multiple client scripts in a client profile and specify the order in which they are executed. For more information, see *Managing Oracle WebCenter Enterprise Capture*.

2

Integrating the Client With Other Web Applications

This chapter discusses how a web application can launch and communicate with the Capture Client.
The web application invokes the Capture Client through a Uniform Resource Locator (URL). Parameters such as the workspace, capture source, client profile, document profile, and optional metadata values are passed within the URL.

For example, you might add a Scan button to a line of business web application. After completing business application entry fields, the user clicks **Scan**. The Capture Client window displays to the user and Capture immediately begins scanning a document using a specified scanner and settings in the client profile specified in the URL.
After scanning, the document is displayed in the document pane. Metadata fields are automatically populated with user entries, which were passed in the URL. The user reviews the document and completes other metadata fields, then releases the batch, scans additional batches, or closes the Capture Client.

When a web application launches the URL, the Capture Client starts and prompts the user to log in. After logged in, the Capture Client uses the parameters, accordingly. If the Capture Client is already running when a web application launches the URL, the parameters will be passed to the already running instance of the Capture Client.

Configuring a Client Integration

To configure an integration between a web application and the Capture Client:

1. In the web application, add a launching point, such as a Scan button, from which to activate the client.
2. Configure the URL and its parameters.

   Parameters are listed and described in Table 2-1.

   See the example integration configuration in Example Client Integration Web Address.

> **Note:**
>
> In 12*c*, client users always have to login to the Capture Client because the Client does not run within a browser.

Table 2-1 Client Integration URL Parameters

Parameter	Description
CaptureWorkspace	Specifies the workspace to which to capture documents.

Table 2-1 (Cont.) Client Integration URL Parameters

Parameter	Description
ClientProfile	Optionally specifies the client profile with which to capture documents. If you specify a profile, the **Client Profile** field does not display in the client's batch pane. If no profile is specified, the client uses the client profile that was last used by the user on the system.
CaptureDriver	Specifies the driver to use to capture documents. • For importing, specify `CAPTURE_IMPORT_DRIVER`. • For TWAIN scanning, specify `CAPTURE_TWAIN_DRIVER`. If neither driver ID or source name are specified, the last used driver and source are used.
CaptureSource	Specify the source to use to capture documents, based on the selected CaptureDriver. • For importing, specify `Import Source`. • For TWAIN scanning, specify the scanner name. This is the same scanner name as identified in the client's **Capture Source** options. If neither driver ID or source name are specified, the client uses the driver and source that were last used by the user on the system.
SignOutOnRelease	Specify whether the business user is signed out of Capture after releasing a batch. • If you specify `false` or `0` (default), the user remains signed in after releasing a batch by clicking the **Release** button. • If you specify `true` or `1`, the user is signed out after a batch is released.
SignOutOnClose	Specify whether the user is signed out of Capture after closing the window. This setting governs whether user will be signed out or not when the Capture client is closed. • If you specify `false` or `0`, the user remains signed in the browser application (from where the capture client was launched) after closing the Capture client window. • If you specify `true` or `1` (default), the user is signed out by redirecting the user to the logout URL in the browser application (from where the capture client was launched).
ShowAllBatches	Specifies if batches display in a list to client users in the batch pane. • If you specify `false` or `0` (default), the batch list is initially empty and only shows batches scanned during the session. • If you specify `true` or `1`, the batch list shows all the batches the user is allowed to see.

Table 2-1 (Cont.) Client Integration URL Parameters

Parameter	Description
DocumentProfile	Specifies the document profile for users to use to index documents. • If you specify a document profile, the **Document Profile** field does not display in the client's metadata pane. • If no profile is specified, the client uses the document profile that was last used by the user on the system.
Other	Any other characters included in the URL are assumed to be a metadata names and values. When specifying date values for Capture metadata fields having a Field Type as *Date*, the date must be in the Coordinated Universal Time (UTC) format, yyyy-MM-ddTHH:mm:ssZ.

Example Client Integration URL

Here is an example URL. (Note that this URL should be all on one line.)

```
oraclecapture://
CaptureWorkspace=Accounting&ClientProfile=Import%20Invoices&CaptureDriver=
CAPTURE_IMPORT_DRIVER&CaptureSource=Import%20Source&SignOutOnRelease=1&Com
pany=MyCompany&Dept=Accounting&Invoice%20Date=2015-08-04T12:00:00Z
```

This URL configures the client integration as follows:

- `CaptureWorkspace=Accounting` - Specifies Accounting as the workspace to which to capture documents.
- `ClientProfile=Import Invoices` - Specifies Import Invoices as the client profile to use.
- `CaptureDriver=CAPTURE_IMPORT_DRIVER` - Specifies CAPTURE_IMPORT_DRIVER as the capture source for importing.
- `CaptureSource=Import Source` - Specifies Import Source as the capture source.
- `SignOutOnRelease=1` - Specifies that the user is signed out after releasing a batch.
- `Company=MyCompany` - Passes a value of MyCompany for the Company metadata field.
- `Dept=Accounting` - Passes a value of Accounting for the Dept metadata field.
- `Invoice%20Date=2015-08-04T12:00:00Z` - Passes a date and time value as August 4, 2015 at 12 noon UTC time for the Invoice Date metadata field.

3
Creating Client Scripts

This chapter describes the various events and classes that can be used to create scripts for Capture client.
Capture enables you to create custom scripts to suit your business requirements. Scripts provide hooks into client events. You can create a client script that gets executed when certain Capture client events are triggered.

Extensions allow you to customize client scripts. You can write and incorporate JavaScript extensions to extend Capture functionality. For more information on JavaScript extensions, see *Managing Oracle WebCenter Enterprise Capture*.

This chapter includes the following sections:

Client Events

Client scripts are JavaScript modules that enable you to customize the behavior of certain client events.

This section describes the following events:

- FieldGotFocus
- FieldLostFocus
- FieldProcessKey
- InitializeFields
- PageCreated
- PreBatchDelete
- PreBatchScan
- PreCaptureImage
- PreDocumentProfileChange
- DocumentProfileChanged
- PreDownloadItem
- PrePageDelete
- PreReleaseBatch
- PreUploadItem
- PostCaptureImage
- PostDownloadItem
- PostUploadItem
- RegionSelected
- ScriptStart

AttachmentCreated

The AttachmentCreated event occurs after an attachment has been created.

Syntax	Parameter
`public void AttachmentCreated(AttachmentCreatedEvent event);`	AttachmentCreatedEvent event

AttachmentRemoved

The AttachmentRemoved event occurs after an attachment has been removed.

Syntax	Parameter
`public void AttachmentRemoved(AttachmentRemoveEvent event);`	AttachmentRemoveEvent event

AttachmentSelected

The AttachmentSelected event occurs when an attachment has been selected.

Syntax	Parameter
`public void AttachmentSelected(AttachmentSelectedEvent event);`	AttachmentSelectedEvent event

BatchScanBegin

The BatchScanBegin event occurs when scanning into a batch is about to begin.

Syntax	Parameter
`public void BatchScanBegin(BatchScanEvent event);`	BatchScanEvent event

BatchScanComplete

The BatchScanComplete event occurs when scanning into a batch is complete.

Syntax	Parameter
`public void BatchScanComplete(BatchScanEvent event);`	BatchScanEvent event

BatchSelected

The BatchSelected event occurs when a batch has been selected.

Syntax	Parameter
`public void BatchSelected(BatchSelectedEvent event);`	BatchScanEvent event

CaptureImage

The CaptureImage event occurs when an image is about to be captured from the scan source.

Syntax	Parameter
`public void CaptureImage(ImageCaptureEvent event);`	ImageCaptureEvent event

CaptureInitialize

The CaptureInitialize event occurs prior to images being captured, and it can be used to initialize properties of the Capture source.

Syntax	Parameter
`public void CaptureInitialize(ImageCaptureEvent event);`	ImageCaptureEvent event

DBSearchComplete

The DBSearchComplete event occurs when the database search is completed and before the results are being processed.

Syntax	Parameter
`public void DBSearchComplete(DBSearchEvent event);`	DBSearchEvent event

DBSearchResults

The DBSearchResults event occurs as database search results are being processed.

Syntax	Parameter
`public void DBSearchResults(DBSearchEvent event);`	DBSearchEvent event

DBSearchStart

The DBSearchStart event occurs just before a database search.

Syntax	Parameter
`public void DBSearchStart(DBSearchEvent event);`	DBSearchEvent event

DocumentCreated

The DocumentCreated event occurs after a document has been created.

Syntax	Parameter
`public void DocumentCreated(CaptureDocument document);`	CaptureDocument document

DocumentRemoved

The DocumentRemoved event occurs after a document has been removed.

Syntax	Parameter
`public void DocumentRemoved(DocumentRemoveEvent event);`	DocumentRemoveEvent event

DocumentSelected

The DocumentSelected event occurs when a document has been selected.

Syntax	Parameter
`public void DocumentSelected(DocumentSelectedEvent event);`	DocumentSelectedEvent event

FieldGotFocus

The FieldGotFocus event occurs when a metadata field receives the input focus.

Syntax	Parameter
`public void FieldGotFocus(FieldEvent event);`	FieldEvent event

FieldLostFocus

The FieldLostFocus event occurs when a field has lost the input focus.

Syntax	Parameter
`public void FieldLostFocus(FieldEvent event);`	FieldEvent event

FieldProcessKey

The FieldProcessKey event occurs when a key event happens while the focus is in a metadata field.

Syntax	Parameter
`public void FieldProcessKey(FieldEvent event);`	FieldEvent event

InitializeFields

The InitializeFields event occurs before the dataFields model is initialized.

Syntax	Parameter
`public void InitializeFields(InitializeFieldsEvent event)`	InitializeFieldsEvent event

PageCreated

The PageCreated event occurs when a page is being added to a document.

Syntax	Parameter
`public void PageCreated(PageCreatedEvent event);`	PageCreatedEvent event

PreBatchDelete

The PreBatchDelete event occurs when a batch is about to be deleted.

Syntax	Parameter
`public void PreBatchDelete(BatchDeleteEvent event);`	BatchDeleteEvent event

PreBatchScan

The PreBatchScan event occurs before a batch is about to be scanned.

Syntax	Parameter
`public void PreBatchScan(BatchScanEvent event);`	BatchScanEvent event

PreCaptureImage

The PreCaptureImage event occurs before an image has been captured from the scan source.

Syntax	Parameter
`public void PreCaptureImage(ImageCaptureEvent event);`	ImageCaptureEvent event

PreDocumentProfileChange

The PreDocumentProfileChange event occurs when a document profile is about to change.

Syntax	Parameter
`public void PreDocumentProfileChange(PreDocumentProfileChangeEvent event);`	PreDocumentProfileChangeEvent event

DocumentProfileChanged

The DocumentProfileChanged event occurs after a document profile is changed.

Syntax	Parameter
`public void DocumentProfileChanged(DocumentProfileChangedEvent event);`	DocumentProfileChangedEvent event

PreDownloadItem

The PreDownloadItem event occurs when a batch item is about to be downloaded.

Syntax	Parameter
`public void PreDownloadItem(DownloadItemEvent event);`	DownloadItemEvent event

PrePageDelete

The PrePageDelete event occurs when one or more pages are about to be deleted.

Syntax	Parameter
`public void PrePageDelete(PageDeleteEvent event);`	PageDeleteEvent event

PreReleaseBatch

The PreReleaseBatch event occurs when a batch is about to be released.

Syntax	Parameter
`public void PreReleaseBatch(ReleaseBatchEvent event);`	ReleaseBatchEvent event

PreUploadItem

The PreUploadItem event occurs when a batch item is about to be uploaded.

Syntax	Parameter
`public void PreUploadItem(UploadItemEvent event);`	UploadItemEvent event

PostCaptureImage

The PostCaptureImage event occurs after an image has been captured from the scan source.

Syntax	Parameter
`public void PostCaptureImage(ImageCaptureEvent event);`	ImageCaptureEvent event

PostDownloadItem

The PostDownloadItem event occurs after a batch item has been downloaded.

Syntax	Parameter
`public void PostDownloadItem(DownloadItemEvent event);`	DownloadItemEvent event

PostUploadItem

The PostUploadItem event occurs after a batch item has been uploaded.

Syntax	Parameter
`public void PostUploadItem(UploadItemEvent event);`	UploadItemEvent event

RegionSelected

The RegionSelected event occurs when a region has been selected on a document page.

Syntax	Parameter
`public void RegionSelected(RegionSelectedEvent event);`	RegionSelectedEvent event

ScriptStart

The ScriptStart event occurs when scripting is first initialized.

The syntax for this event is: `public void ScriptStart();`

Event Classes

An event class is used to define an event. This section describes the following event classes:

- AttachmentCreatedEvent
- AttachmentRemoveEvent
- AttachmentSelectedEvent
- BatchDeleteEvent
- BatchScanEvent
- BatchSelectedEvent
- DBSearchEvent
- DocumentRemoveEvent

- DocumentSelectedEvent
- PreDocumentProfileChangeEvent
- DocumentProfileChangedEvent
- DownloadItemEvent
- FieldEvent
- ImageCaptureEvent
- InitializeFieldsEvent
- PageCreatedEvent
- PageDeleteEvent
- RegionSelectedEvent
- ReleaseBatchEvent
- UploadItemEvent

AttachmentCreatedEvent

The AttachmentCreatedEvent class is used in events that occur when an attachment has been created.

Property	Type	Description
attachment	CaptureAttachment	The attachment that has been created.

AttachmentRemoveEvent

The AttachmentRemoveEvent class is used in events that occur when a user removes one or more attachments from a document.

Property	Type	Description
attachments	List<CaptureAttachment>	The attachments being removed from the document.

AttachmentSelectedEvent

The AttachmentSelectedEvent class is used in events that occur when a user selects an attachment.

Property	Type	Description
attachment	CaptureAttachment	The attachment that has been selected in the batch pane.

BatchDeleteEvent

The BatchDeleteEvent class is used in events that occur when a user deletes a batch.

Property	Type	Description
batches	List<CaptureBatch>	List of batches that will be deleted.
canceled	boolean	If set to *True*, the delete operation will be canceled.

BatchScanEvent

The BatchScanEvent class is used in events that occur when a user scans a batch.

Property	Type	Description
batch	CaptureBatch	The batch that new items will be added to during scan or import.
canceled	boolean	If set to *True*, the scan or import will be canceled.
sourceFiles	List<File>	When using the Import Source, this contains the list of files being imported.
operation	CaptureOperation	Indicates the operation that triggered this event.
allowConfiguration	boolean	Setting this property to *True* in the PreBatchScan event allows scanners to display their configuration dialog, before the first page is scanned.

BatchSelectedEvent

The BatchSelectedEvent class is used in events that occur when a user selects a batch.

Property	Type	Description
batch	CaptureBatch	Batch that has been selected in the batch pane.

DBSearchEvent

The DBSearchEvent class is used in events that occur when a user initiates a database lookup.

Property	Type	Description
displayHitlist	boolean	If set to *True*, displays the database lookup results.
exactMatch	boolean	If set to *True*, the search value must be an exact match.
metadataID	String	Unique identifier of the metadata field being searched.
metadataValue	String	Value of the metadata field being searched.

Property	Type	Description
rowResults	List<DbSearchResultRow>	List of row results returned from the search.
canceled	boolean	If set to *True*, cancels the search.

DocumentRemoveEvent

The DocumentRemoveEvent class is used in events that occur when a user removes a document from the batch.

Property	Type	Description
documents	List<CaptureDocument>	A list of documents being removed from the batch.
destinationDocument	CaptureDocument	When removing a document separation, this property contains the document to which all pages are to be moved.

The following table describes the syntax for isRemove() method:

Syntax	Description
`public boolean isRemove()`	Returns *True* if a document separation is being removed.

The following table describes the syntax for isDelete() method:

Syntax	Description
`public boolean isDelete()`	Returns *True* if the list of documents is being deleted.

DocumentSelectedEvent

The DocumentSelectedEvent class is used in events that occur when a user selects a document.

Property	Type	Description
document	CaptureDocument	The document that has been selected in the batch pane.

PreDocumentProfileChangeEvent

The PreDocumentProfileChangeEvent class is used in events that occur when a user is about to change a document profile.

Property	Type	Description
oldDocumentType	DocumentType	The previous document type or document profile before change.

Property	Type	Description
newDocumentType	DocumentType	The currently selected document type or document profile.
canceled	Boolean	If set to true, the change will be reverted back in the document type or profile drop-down.

DocumentProfileChangedEvent

The DocumentProfileChangedEvent class is used in events that occur when a user has changed a document profile.

Property	Type	Description
documentType	DocumentType	The currently selected document type or document profile.

DownloadItemEvent

The DownloadItemEvent class is used in events that occur when batch items are downloaded from the server.

Property	Type	Description
captureItem	CaptureItem	After a batch is opened, indicates the current item being downloaded from the server.

FieldEvent

The FieldEvent class is used in events that occur when a user enters a field, exits a field, or types into a field.

Property	Type	Description
cancel	Boolean	If set to *True*, cancels the event.
field	DataField	The field this event is acting upon.
keyEvent	KeyEvent	The keyboard event used to generate this event.
traversalDirection	Integer	Indicates which direction the field focus is moving. **TRAVERSAL Constants**: • TRAVERSAL_UNDETERMINED = 0 • TRAVERSAL_FORWARD = 1 • TRAVERSAL_BACKWARD = 2 • TRAVERSAL_FORWARD_COMPONENT = 3 • TRAVERSAL_BACKWARD_COMPONENT = 4

ImageCaptureEvent

The ImageCaptureEvent class is used in events that occur when the user is capturing an image.

Property	Type	Description
cancel	boolean	If set to *True*, the capture operation will be canceled.
imageCount	Integer	Indicates how many images have been captured.
imageFileName	String	Indicates the file name of image saved locally.
xdpi	Integer	For images, indicates the horizontal dots per inch.
ydpi	Integer	For images, indicates the vertical dots per inch.
brightness	Integer	The brightness value used to capture the image.
contrast	Integer	The contrast value used to capture the image.
logicalBreak	boolean	If set to *True*, indicates the start of a document.
sourceFiles	List<File>	When using the Import Source, contains the list of files being imported.
sourceFileName	String	When using the Import Source, contains the name of the source file being imported.
image	BufferedImage	For images files, contains a BufferedImage object.
source	ImageCaptureEngine	The ImageCaptureEngine that created this event.
imageFormat	ImageCaptureEngine.ImageFormat	Indicates the format that images will be saved as Available Formats: tiffG4, tiffJpegGray, tiffJpegColor, jpegGray, and jpegColor.
printerString	String [ ]	The imprinter or endorser strings printed on a page during capture.

InitializeFieldsEvent

The InitializeFieldsEvent class is used in events that occur before the fieldedit control is initialized on the client. You can use this class to modify or remove the DataFields from the list.

Property	Type	Description
dataFields	List <DataField>	The list of DataFields that will be displayed to the user.

PageCreatedEvent

The PageCreatedEvent class is used when a page is created.

Property	Type	Description
page	CaptureDocumentPage	The page that was created.

PageDeleteEvent

The PageDeleteEvent class is used when a page is being deleted.

Property	Type	Description
pages	List<CaptureDocumentPage>	The list of pages being deleted. You can modify this list to remove pages that you do not want to delete.
canceled	boolean	If set to *True*, no pages will be deleted.

RegionSelectedEvent

The RegionSelectedEvent class is used in events that occur when a user selects a region of the image in the viewer.

Property	Type	Description
mouseEvent	MouseEvent	The MouseEvent used to select the region within the image.
selectionRectangle	Rectangle	The rectangle selected within the image.
image	BufferedImage	The BufferedImage containing the selected portion of the image.

ReleaseBatchEvent

The ReleaseBatchEvent class is used in events that occur when a batch is about to get released or unlocked.

Property	Type	Description
batches	List<CaptureBatch>	The list of batches that are about to be released.
processorID	String	The unique identifier of the processor to which the batches will get released.
jobID	String	The unique identifier of the processor job.
canceled	boolean	If the flag is set to *True*, the release will be canceled.
releaseProcessName	String	The name of the process used to release the batch.

UploadItemEvent

The UploadItemEvent class is used in events that occur when batch items are uploaded to the server.

Property	Type	Description
captureItem	CaptureItem	After a batch is released, indicates the current item being uploaded to the server.

Capture Client Core Classes

This section describes the following Capture Client Core classes:

- AttachmentType
- AttachmentTypes
- BarcodeInfo
- CaptureAttachment
- CaptureAttachments
- CaptureBatch
- CaptureBatchStatus
- CaptureDataType
- CaptureDocument
- CaptureDocumentPage
- CaptureDocumentPages
- CaptureDocuments
- CaptureErrorManager
- CaptureField
- CaptureFields
- CaptureItem
- CaptureItems
- CaptureOperation
- CaptureStateManager
- CaptureWorkspace
- ClientProfile
- ClientReleaseProcess
- ClientReleaseProcesses
- ClientUI
- DBLookupProfile
- DBLookupResult

- DbSearchResultRow
- DbSearchFieldInfo
- DocumentType
- DocumentTypes
- FieldDefinition
- FieldDefinitions
- ImageCaptureEngine
- MicrInfo
- Source
- TWAINSource

AttachmentType

The AttachmentType class contains all properties of an attachment type.

Property	Type	Description
ID	String	The internal identifier for the attachment type.
name	String	The name for the attachment type.
description	String	The description for the attachment type.
required	Boolean	Indicates if the attachment type is required.

AttachmentTypes

The AttachmentTypes class is a map of attachment types. It is of type LinkedHashMap<String, AttachmentType> and the map key is the attachment type ID. Use the LinkedHashMap methods to retrieve the attachment types from instances of this class.

See the Java API documentation for more information on the LinkedHashMap class and its methods.

BarcodeInfo

The BarcodeInfo class contains all the properties associated with a bar code detected by a scanner.

Property	Type	Description
type	Integer	The type of the bar code. Valid bar code types are: • TWBT_3OF9 - 0 • TWBT_2OF5INTERLEAVED - 1 • TWBT_2OF5NONINTERLEAVED - 2 • TWBT_CODE93 - 3 • TWBT_CODE128 - 4 • TWBT_UCC128 - 5 • TWBT_CODABAR - 6 • TWBT_UPCA - 7 • TWBT_UPCE - 8 • TWBT_EAN8 - 9 • TWBT_EAN13 - 10 • TWBT_POSTNET - 11 • TWBT_PDF417 - 12 • TWBT_2OF5INDUSTRIAL - 13 • TWBT_2OF5MATRIX - 14 • TWBT_2OF5DATALOGIC - 15 • TWBT_2OF5IATA - 16 • TWBT_3OF9FULLASCII - 17 • TWBT_CODABARWITHSTARTSTOP - 18 • TWBT_MAXICODE - 19
text	String	The textual value of the bar code.
x	Integer	The horizontal location (in pixels) of the bar code on the page. This value may be null if the scanner does not supply this information.
y	Integer	The vertical location (in pixels) of the bar code on the page. This value may be null if the scanner does not supply this information.
confidence	Integer	The degree of certainty in the accuracy of the bar code information. The value ranges from 0 (no confidence) to 100 (supreme confidence) or may be null or (-)1 if the scanner does not supply this information.
rotation	Integer	The orientation of the bar code. Valid orientations are: • TWBCOR_ROT0 (Normal reading orientation) - 0 • TWBCOR_ROT90 (Rotated 90 degrees clockwise) - 1 • TWBCOR_ROT180 (Rotated 180 degrees clockwise) - 2 • TWBCOR_ROT270 (Rotated 270 degrees clockwise) - 3 • TWBCOR_ROTX (The orientation is not known) - 4 This value may be null if the scanner does not supply this information.

CaptureAttachment

The CaptureAttachment class contains all properties of an attachment.

Property	Type	Description
attachmentType	AttachmentType	The type of attachment.
id	String	The internal identifier for the attachment.
pages	CaptureDocumentPages	The pages contained in the attachment.
parentDocument	CaptureDocument	The parent document owning this attachment.
title	String	The title of the attachment.

The following table describes the syntax for persist() method:

Syntax	Description
`persist() throws BatchLockException, CaptureException`	Saves the attachment and related document pages to the server.

CaptureAttachments

The CaptureAttachments class is a collection of attachments and is of type Vector<CaptureAttachment>. Use the Vector methods to retrieve attachments from instances of this class.

See the Java API documentation for more information on the Vector class and its methods.

CaptureBatch

The CaptureBatch class contains all properties and operations for a batch.

Property	Type	Description
batchId	String	The internal identifier for the batch.
batchName	String	The name of the batch.
batchPath	String	The local path where information for this batch is stored.
createdBy	String	The user that created the batch.
createdDate	Date	The date that the batch was created.
currentPriority	Integer	The priority assigned to the batch.
currentStatus	CaptureBatchStatus	The status assigned to the batch.
documents	CaptureDocuments	The documents contained in the batch.
items	CaptureItems	The items associated with the batch.
jobID	String	ID of the processor job. This property is populated just before the batch is released.
lastModifiedDate	Date	The date that the batch was last modified.

Property	Type	Description
note	String	The note assigned to the batch.
processorID	String	ID of the processor to which the batch will get released. This property is populated just before the batch is released.
workspace	CaptureWorkspace	The workspace used to create the batch.

The following table describes the syntax for persist() method:

Syntax	Description
`persist() throws BatchLockException, CaptureException`	Saves the batch record to the server.

CaptureBatchStatus

The CaptureBatchStatus class contains the properties of a batch status.

Property	Type	Description
value	String	The description of the status.
id	String	The internal identifier of the status.

CaptureDataType

The CaptureDataType is an enumeration that defines the data types for metadata field definitions. The following are valid Capture data types:

- NUMERIC
- ALPHA
- ALPHANUMERIC
- DATE
- FLOAT

CaptureDocument

The CaptureDocument class contains all properties of a document.

Property	Type	Description
documentType	DocumentType	The document profile assigned to the document.
fields	CaptureFields	The metadata assigned to the document.
id	String	The internal identifier for the document.
pages	CaptureDocumentPages	The pages contained in the document.
parentBatch	CaptureBatch	The batch that contains this document.
title	String	The title of the document.

Property	Type	Description
attachments	CaptureAttachments	The attachments associated with this document.

The following table describes the syntax for persist() method:

Syntax	Description
`persist() throws BatchLockException, CaptureException`	Saves the document, related document pages, and metadata to the server.

CaptureDocumentPage

The CaptureDocumentPage class contains the properties of a document page.

Property	Type	Description
imageFilenameKey	String	The local filename for this page.
item	CaptureItem	The item associated with the page.
pageID	String	The internal identifier for the page.
pageNumber	Integer	The number of the page within the document.
parentDocument	CaptureDocument	The document that contains this page.

CaptureDocumentPages

The CaptureDocumentPages class is a collection of document pages and is of type Vector<CaptureDocumentPage>. Use the Vector methods to retrieve document pages from instances of this class.

See the Java API documentation for more information on the Vector class and its methods.

CaptureDocuments

The CaptureDocuments class is a collection of documents and is of type Vector<CaptureDocument>. Use the Vector methods to retrieve documents from instances of this class.

See the Java API documentation for more information on the Vector class and its methods.

CaptureErrorManager

The CaptureErrorManager class manages what error messages are logged.

Property	Type	Description
logLevel	Level	The minimum level used to log messages.

The following table describes the syntax for logMessage() method:

Syntax	Description
`logMessage(Level level, String message)` `logMessage(Level level, String message, Throwable errorException)`	Logs a message with the specified log level. or Logs a message and error with the specified log level.

The following table describes the parameters for logMessage() method:

Parameter	Type	Description
level	Level	The severity level for this log entry.
message	String	The message you wish to log.
errorException	Throwable	If logging an error, the exception that is the cause of the error.

CaptureField

The CaptureField class contains the properties of a document metadata field.

Property	Type	Description
displayValue	String	The value to display for this field.
fieldName	String	The name of the field.
length	Integer	The maximum length of the field.
required	boolean	If *True*, this field is required to have a value.
value	String	The value for this field.

The following table describes the syntax for setDate() method:

Syntax	Description
`setDate(Date date)`	Sets the value of the metadata field to a date.

The following table describes the parameters for setDate() method:

Parameter	Type	Description
date	Date	The date value to be set.

CaptureFields

The CaptureFields class is a map of metadata field definitions. It is of type LinkedHashMap<String, CaptureField> and the map key is the field name. Use the LinkedHashMap methods to retrieve the fields from instances of this class.

See the Java API documentation for more information on the LinkedHashMap class and its methods.

CaptureItem

The CaptureItem class contains properties of an item (single image or non-image file) associated with a document page.

Property	Type	Description
filename	String	The name of the file for this item.
parentBatch	CaptureBatch	The batch containing this item.
sourceFilename	String	If this item was imported, contains the name of the file it was imported from.
sourceFormat	String	Non-image documents contains the format of the document, which is typically the file extension.
patchCode	Integer	The patch code for this item.
barcodes	List<String>	The bar codes for this item.
micrValue	String	The scanner's MICR text for this item.
endorsement	String	The scanner's endorsement text for this item.

The following table describes the syntax for persist() method:

Syntax	Description
`persist() throws BatchLockException, CaptureException`	Saves the item to the server.

CaptureItems

The CaptureItems class is a map of Capture items. It is of type TreeMap<String, CaptureItem> and the map key is the item filename. Use the TreeMap methods to retrieve the items from instances of this class.

See the Java API documentation for more information on the TreeMap class and its methods.

CaptureOperation

This is an enumeration that defines the capture operation being performed on the batch. The following are valid Capture operation values:

- Create
- Append
- Insert
- Replace

CaptureStateManager

The CaptureStateManager class contains properties related to the current state of the client. The instance of this class is available to all scripting events through the Capture property.

Property	Type	Description
activeAttachment	CaptureAttachment	The active document attachment.
activeAttachmentPage	CaptureDocumentPage	The active document attachment page.
activeAttachmentType	AttachmentType	The active document attachment type.
activeBatch	CaptureBatch	The active batch.
activeDocument	CaptureDocument	The active document.
activeDocumentPage	CaptureDocumentPage	The active document page.
activeDocumentType	DocumentType	The active document profile.
activeProfile	ClientProfile	The active client profile.
applicationUserPath	String	The path the client uses for its application data.
batchesPath	String	The path the client uses to cache batch data.
captureSystemId	String	The internal identifier of the Capture system that the client is connected to.
computerName	String	The name of the computer the client is running on.
currentUser	String	The user currently logged into the client.
errorManager	CaptureErrorManager	The Error Manager object used for logging information.

CaptureWorkspace

The CaptureWorkspace class contains all properties and operations for a workspace.

Property	Type	Description
id	String	The internal identifier associated with the workspace.
fieldDefinitions	FieldDefinitions	The metadata defined for this workspace.
name	String	The name of the workspace.
statuses	List<CaptureBatchStatus>	A list of batch statuses available to this workspace.

The following table describes the syntax for getDBLookupProfile() method:

Syntax	Description
`public DBLookupProfile getDBLookupProfile(String profileId) throws CaptureException`	Retrieves the database lookup profile for the given database lookup profile ID.

The following table describes the parameter for getDBLookupProfile() method:

Parameter	Type	Description
profileId	String	The identifier of the database lookup profile.

ClientProfile

The ClientProfile class contains the properties of a client profile as defined in the Capture Workspace Console.

Property	Type	Description
alwaysDisplayHitList	boolean	If *True*, after a database lookup is executed, the results are displayed regardless of the number of results returned.
applyBrightness	boolean	If *True*, applies the brightness and contrast settings to the selected Capture source.
batchFilterDaysOldFrom	Integer	A batch filter setting that specifies the minimum days old a batch can be.
batchFilterDaysOldTo	Integer	A batch filter setting that specifies the maximum days old a batch can be.
batchFilterPrimarySortField	String	The batch property used for the primary sort of the batches tree.
batchFilterPrimarySortOrder	SortOrder	The primary sort order for the batches tree.
batchFilterSecondarySortField	String	The batch property used for the secondary sort of the batches tree.
batchFilterSecondarySortOrder	SortOrder	The secondary sort order for the batches tree.
batchFilterState	Integer	The batch states to include in the batch filter.
batchPrefix	String	The batch prefix to use for batches created using this profile.
batchVisibility	BatchVisibility	A batch filter setting that specifies when a user sees the batch in the batches tree.
blankByteThreshold	long	If the number of bytes in the file size of an image is less than the blankByteThreshold, the page is considered to be a blank page.
brightness	Integer	The brightness to apply to the selected Capture source.
captureType	CaptureType	Indicates whether the profile is capture-only, capture and index, or index-only.
contrast	Integer	The contrast to apply to the selected Capture source.
dBLookupMaxRecords	Integer	The maximum number of records to return from a database lookup.
dBLookupProfileId	String	The unique identifier of the database lookup profile used by this profile.
defaultColor	ColorType	The default color type used during capture.

Property	Type	Description
defaultDpi	Integer	The default DPI to use during capture.
defaultPriority	Integer	A batch created from this profile will be assigned this priority.
defaultStatusId	String	A batch created from this profile will be assigned this status.
description	String	The description for the profile.
documentCreationType	DocumentCreationType	Specifies how many pages are created per document at capture time.
documentTypes	DocumentTypes	An object containing the document profiles that this profile can assign to a batch.
id	String	The identifier associated with this profile.
maxPages	Integer	The non-image file preview page limit.
name	String	The name of the profile.
nonImageAction	NonImageAction	The action to take for non-image files.
nonImageDpi	int	For client profiles having a nonImageAction property of type *CONVERT*, this property contains the image DPI for non-image files.
nonImageFormat	String	For client profiles having a nonImageAction property of type *CONVERT*, this property contains the image format for non-image files. This value will be either *TIFF_BW* or *JPEG*.
nonImageJpegQuality	int	For client profiles having a nonImageAction property of type *CONVERT* and a nonImageFormat value of *JPEG*, this property contains the JPEG image quality for non-image files.
picklistRelationshipProfile	String	The dependent choice list used by this profile.
prefixes	List<String>	The batch prefixes used in the batch filter.
preventDefaultColorOverride	boolean	If *True*, the color cannot be overridden.
preventDefaultDpiOverride	boolean	If *True*, the dpi cannot be overridden.
priorities	List<Integer>	The batch priorities used in the batch filter.
sepByteThreshold	Integer	If the number of bytes in the file size of an image is less than the sepByteThreshold, the page is considered to be a separator sheet.
statuses	List<String>	The batch statuses used in the batch filter.
supportedDocumentTypes	List<String>	A list of document profile identifiers which represent the document profiles that this profile can assign to a batch.
workspaceId	String	The unique identifier of the workspace in which this profile is associated.

Property	Type	Description
workspaceName	String	The name of the workspace in which this profile is associated.
releaseProcesses	ClientReleaseProcesses	A list of the available client release processes.

The following table describes the enumeration and values for the ClientProfile class:

Enumeration	Value
AutoPopulateType	NONE, SCANDATE, INDEXDATE, DEFAULTVALUE, BATCHNAME, USERID, COMPUTERNAME, CLIENTPROFILENAME, BATCHSTATUS, BATCHPRIORITY
BatchVisibility	USER_AND_COMPUTER, USER, ALL
DocumentCreationType	ONE_PAGE, TWO_PAGES, VARIABLE_PAGES, PROMPT_USER
CaptureType	CAPTURE_ONLY, CAPTURE_AND_INDEX, INDEX_ONLY
NonImageAction	DISALLOW, ALLOW, CONVERT
SortOrder	ASCENDING, DESCENDING
ColorType	NotSpecified, BlackAndWhite, Gray, Color

ClientReleaseProcess

The ClientReleaseProcess class contains all properties of a client release process.

Property	Type	Description
name	String	The name of the release process.
description	String	The description of the release process.
processorID	String	The processor ID of the release process.
jobID	String	The processor job ID of the release process.
defaultRelease	boolean	Indicates if this is the default release process.

ClientReleaseProcesses

The ClientReleaseProcesses class is a list of client release processes. It is of type ArrayList<ClientReleaseProcess>. Use the ArrayList methods to retrieve the client release processes from instances of this class.

See the Java API documentation for more information on the ArrayList class and its methods.

ClientUI

The ClientUI class allows the user to invoke user interface related actions and can be accessed through client scripts.

Property	Type	Description
batchEditForm	BatchEditForm	An instance of the current batch edit form.

This class includes the following methods:

- releaseBatch()
- setActiveMetadataFieldByName()
- setActiveMetadataFieldByID()
- execDBSearch()
- execDBSearch()
- selectDocument()
- refreshDocumentMetadata()

releaseBatch()

The following table describes the syntax for releaseBatch() method:

Syntax	Description
`public void releaseBatch(List<CaptureBatch> batches, String processorID, String jobID)`	Unlocks or releases a list of batches for further processing.

The following table describes the parameters for releaseBatch() method:

Parameter	Type	Description
batches	List<CaptureBatch>	The list of batches to be released.
processorID	String	The unique identifier of the processor to which the batches will get released.
jobID	String	The unique identifier of the processor job.

setActiveMetadataFieldByName()

The following table describes the syntax for setActiveMetadataFieldByName() method:

Syntax	Description
`public void setActiveMetadataFieldByName(String fieldName)`	Moves the focus in the metadata pane to the metadata field specified by the field name.

The following table describes the parameter for setActiveMetadataFieldByName() method:

Parameter	Type	Description
fieldName	String	The name of the metadata field to which the focus is to be moved.

setActiveMetadataFieldByID()

The following table describes the syntax for setActiveMetadataFieldByID() method:

Syntax	Description
`public void setActiveMetadataFieldByID(String fieldID)`	Moves the focus in the metadata pane to the metadata field name specified by the field ID.

The following table describes the parameter for setActiveMetadataFieldByID() method:

Parameter	Type	Description
fieldID	String	The unique identifier of the metadata field to which the focus is to be moved.

execDBSearch()

The following table describes the syntax for execDBSearch() method:

Syntax	Description
`public void execDBSearch(String metadataName, String metadataValue, Boolean alwaysDisplayHitList)`	Performs a database lookup for the given metadata field name and value. This method then updates the metadata fields in the metadata pane with the results.

The following table describes the parameters for execDBSearch() method:

Parameter	Type	Description
metadataName	String	The name of the metadata field that is to be looked up.
metadataValue	String	The value that is used to perform the lookup.
alwaysDisplayHitList	Boolean	If set to *True*, displays the lookup results irrespective of the number of results.

execDBSearch()

The following table describes the syntax for execDBSearch() method:

Syntax	Description
`public void execDBSearch(String metadataName, String metadataValue, Boolean alwaysDisplayHitList, Boolean exactSearch, Integer maximumRecords)`	Performs a database lookup for the given metadata field name and value. This method then updates the metadata fields in the metadata pane with the results.

The following table describes the parameters for execDBSearch() method:

Parameter	Type	Description
metadataName	String	The name of the metadata field that is to be looked up.
metadataValue	String	The value that is used to perform the lookup.
alwaysDisplayHitList	Boolean	If set to *True*, displays the lookup results irrespective of the number of results.
exactSearch	Boolean	If set to *True*, the result must exactly match the value being searched. If set to *False*, partial matches are also returned.
maximumRecords	Integer	Indicates the maximum number of records the search returns.

selectDocument()

The following table describes the syntax for selectDocument() method:

Syntax	Description
`public void selectDocument(CaptureDocument document)`	Selects the given document in the batches tree.

The following table describes the parameters for selectDocument() method:

Parameter	Type	Description
document	CaptureDocument	The document to select.

refreshDocumentMetadata()

The following table describes the syntax for refreshDocumentMetadata() method:

Syntax	Description
`public void refreshDocumentMetadata()`	Reloads the metadata from the current document and displays it in the metadata pane.

DBLookupProfile

The DBLookupProfile class represents a profile for database lookup. This class includes the following method:

execDBLookup()

The following table describes the syntax for execDBLookup() method:

Syntax	Description

Database Lookup without sorting
`DBLookupResult execDBLookup(String searchID, String fieldID, String value, boolean exactMatch, Integer maxRows) throws CaptureException`
Database Lookup with sorting
`DBLookupResult execDBLookup(String searchID, String fieldID, String value, Boolean exactMatch, String primarySortField, Integer primarySortOrder, String secondarySortField, Integer secondarySortOrder, Integer maxRows) throws CaptureException`
	Executes a database lookup with or without sorting.
	Sort Constants:
	• SORT_ASC = 0
	• SORT_DESC = 1

The following table describes the parameters for the execDBLookup() method:

Parameter	Type	Description
searchID	String	The unique identifier of the database search defined for the database lookup profile.
fieldID	String	The unique identifier of the metadata field being searched.
value	String	The value being searched upon.
exactMatch	Boolean	If *True*, the value must be an exact match.
primarySortField	String	The field identifier used for the primary sort.
primarySortOrder	Integer	The sort order of the primary sort.
secondarySortField	String	The field identifier used for the secondary sort.
secondarySortOrder	Integer	The sort order of the secondary sort.
maxRows	Integer	The maximum number of rows to be returned.

DBLookupResult

The DBLookupResult class represents the result of a database lookup.

Property	Type	Description
searchFieldInfoList	List<DbSearchFieldInfo>	A list of search field information describing the results returned by the database lookup.
searchResultRows	List<DbSearchResultRow>	A list of search result rows returned by the database lookup.

DbSearchResultRow

The DbSearchResultRow class represents one row result returned from a database lookup.

Property	Type	Description
results	List<String>	A list of string values associated with one search result. The values in the list will be in the same order in which the return fields are defined.

DbSearchFieldInfo

The DbSearchFieldInfo class represents the field information describing the results of a database lookup.

Property	Type	Description
captureIndexDefID	String	The metadata field identifier.
dbColumnName	String	The name of the database column.
dbColumnType	Integer	The type of the database column.
captureFieldType	Integer	The data type of the metadata field.

DocumentType

The DocumentType class represents a document profile. A document profile dictates what metadata fields are available to documents created from this type.

Property	Type	Description
fieldDefinitions	FieldDefinitions	The metadata applicable to the document profile.
id	String	The internal identifier associated with the document profile.
name	String	The name for the document profile.
description	String	The description for the document profile.
attachmentTypes	AttachmentTypes	The attachment types available to the document profile.

DocumentTypes

The DocumentTypes class is a map of document profiles. It is of type TreeMap<String, DocumentType> and the map key is the document profile ID. You can use the TreeMap methods to retrieve the document profiles from instances of this class.

See the Java API documentation for more information on the TreeMap class and its methods.

FieldDefinition

The FieldDefinition class represents a metadata field's definition.

Property	Type	Description
autoPopulateType	ClientProfile.AutoPopulateType	Specifies how the field should be auto-populated.
dataType	CaptureDataType	The data type of this field.
autoPopulateDefault	String	The default value for this field.
displayable	boolean	Indicates whether this field will be displayed in the client.
length	Integer	The maximum length for this field.
id	String	The internal identifier for this field.
name	String	The name of this field.
inputMask	String	The input mask.
locked	boolean	Set to *True*, if the field is locked.
maxValue	Float	The maximum value for this field.
minValue	Float	The minimum value for this field.
pickListCaseInsensitive	boolean	If *True*, the choice list is case insensitive.
pickListID	String	The unique identifier of the choice list associated with this field.
pickListParentFieldID	String	The unique identifier of the choice list parent field.
pickListSourceID	String	The source identifier of the dependent choice list for this field.
profileDisplayFormat	String	The format used when displaying the field.
required	boolean	Indicates whether this field is required to have a value.
validationExpression	String	A regular expression used to validate the values entered for this field.

FieldDefinitions

The FieldDefinitions class is a map of metadata field definitions. It is of type LinkedHashMap<String, FieldDefinition> and the map key is the metadata field definition ID. You can use the LinkedHashMap methods to retrieve the metadata field definitions from instances of this class.

See the Java API documentation for more information on the LinkedHashMap class and its methods.

ImageCaptureEngine

The ImageCaptureEngine class contains all the properties and operations associated with image capture.

Property	Type	Description
activeSource	ImageCaptureSource	The active source of the Image Capture Engine from which to capture images.
activeDriver	ImageCaptureDriver	The active driver of the Image Capture Engine from which to capture images.

MicrInfo

The MicrInfo class contains all the properties associated with the magnetic data detected by a scanner.

Property	Type	Description
string	String	The textual information read from the MICR data.

Source

The Source class is a low-level class used to interact with a TWAIN scanner.

This class includes the following methods:

- getCurrentPrinter()
- getPrinterIndex()
- getPrinterMode()
- getPrinterString()
- getPrinterSuffix()
- getSupportedPrinters()
- isPrinterEnabled()
- isBarcodeDetectionEnabled()
- getBarcodeMaxRetries()
- getBarcodeMaxSearchPriorities()
- getBarcodeSearchMode()
- getBarcodeTimeout()
- getBarcodeSearchPriorities()
- getSupportedBarcodeTypes()
- getBarcodesDetected()
- isPatchCodeDetectionEnabled()
- getPatchCodeMaxRetries()
- getPatchCodeMaxSearchPriorities()
- getPatchCodeSearchMode()
- getPatchCodeTimeout()

- getPatchCodeSearchPriorities()
- getSupportedPatchCodeTypes()
- getPatchCodesDetected()
- isMicrEnabled()
- getMicrDetected()
- setPrinterIndex(int printerIndex)
- setPrinterMode(int printerMode)
- setPrinterString(String printerString)
- setPrinterString(String[ ] printerString)
- setPrinterSuffix(String printerSuffix)
- setBarcodeDetectionEnabled(boolean enabled)
- setBarcodeMaxRetries(int maxRetries)
- setBarcodeMaxSearchPriorities(int maxSearchPriorities)
- setBarcodeSearchMode(int searchMode)
- setBarcodeTimeout(int timeout)
- setBarcodeSearchPriorities(int[] value)
- setPatchCodeDetectionEnabled(boolean enabled)
- setPatchCodeMaxRetries(int maxRetries)
- setPatchCodeMaxSearchPriorities(int maxSearchPriorities)
- setPatchCodeSearchMode(int searchMode)
- setPatchCodeTimeout(int timeout)
- setPatchCodeSearchPriorities(int[] value)
- setMicrEnabled(boolean enabled)

getCurrentPrinter()

The following table describes the syntax for getCurrentPrinter() method:

Syntax	Description
`public int getCurrentPrinter() throws InvalidStateException, OperationException;`	Returns the current imprinter or endorser. See the setPrinter(int printer) method for valid values.

getPrinterIndex()

The following table describes the syntax for getPrinterIndex() method:

Syntax	Description
`public int getPrinterIndex() throws OperationException, InvalidStateException;`	Returns the counter value of the current imprinter or endorser.

getPrinterMode()

The following table describes the syntax for getPrinterMode() method:

Syntax	Description
`public int getPrinterMode() throws OperationException, InvalidStateException;`	Returns the mode of the current imprinter or endorser. See the method setPrinterMode(int printerMode) for the meaning of the return value.

getPrinterString()

The following table describes the syntax for getPrinterString() method:

Syntax	Description
`public String[] getPrinterString() throws OperationException, InvalidStateException;`	Returns the string or format of the current imprinter or endorser.

getPrinterSuffix()

The following table describes the syntax for getPrinterSuffix() method:

Syntax	Description
`public String getPrinterSuffix() throws OperationException, InvalidStateException;`	Returns the suffix string of the current imprinter or endorser.

getSupportedPrinters()

The following table describes the syntax for getSupportedPrinters() method:

Syntax	Description
`public int[] getSupportedPrinters() throws InvalidStateException, OperationException;`	Returns an array of the supported imprinters or endorsers. See the setPrinter(int printer) method for the meaning of the return values.

isPrinterEnabled()

The following table describes the syntax for isPrinterEnabled() method:

Syntax	Description
`public boolean isPrinterEnabled() throws OperationException, InvalidStateException;`	Returns whether the current imprinter or endorser is enabled.

isBarcodeDetectionEnabled()

The following table describes the syntax for isBarcodeDetectionEnabled() method:

Syntax	Description
`public boolean isBarcodeDetectionEnabled() throws OperationException, InvalidStateException;`	Returns whether bar code detection is enabled for the scanner.

getBarcodeMaxRetries()

The following table describes the syntax for getBarcodeMaxRetries() method:

Syntax	Description
`public int getBarcodeMaxRetries() throws OperationException, InvalidStateException;`	Returns the number of times a search will be retried if no bar codes are found.

getBarcodeMaxSearchPriorities()

The following table describes the syntax for getBarcodeMaxSearchPriorities() method:

Syntax	Description
`public int getBarcodeMaxSearchPriorities() throws OperationException, InvalidStateException;`	Returns the maximum number of supported bar code search priorities.

getBarcodeSearchMode()

The following table describes the syntax for getBarcodeSearchMode() method:

Syntax	Description
`public int getBarcodeSearchMode() throws OperationException, InvalidStateException;`	Returns the orientation or orientation priority used for bar code searching. Possible return values are: • TWBD_HORZ - 0 • TWBD_VERT - 1 • TWBD_HORZVERT - 2 • TWBD_VERTHORZ - 3

getBarcodeTimeout()

The following table describes the syntax for getBarcodeTimeout() method:

Syntax	Description
`public int getBarcodeTimeout() throws OperationException, InvalidStateException;`	Returns the maximum amount of time to spend searching for bar codes on a page.

getBarcodeSearchPriorities()

The following table describes the syntax for getBarcodeSearchPriorities() method:

Syntax	Description
`public int[] getBarcodeSearchPriorities() throws OperationException, InvalidStateException;`	Returns a prioritized array of bar code types dictating the order in which they will be sought. Valid bar code types are: TWBT_3OF9, TWBT_2OF5INTERLEAVED, TWBT_2OF5NONINTERLEAVED, TWBT_CODE93, TWBT_CODE128, TWBT_UCC128, TWBT_CODABAR, TWBT_UPCA, TWBT_UPCE, TWBT_EAN8, TWBT_EAN13, TWBT_POSTNET, TWBT_PDF417, TWBT_2OF5INDUSTRIAL, TWBT_2OF5MATRIX, TWBT_2OF5DATALOGIC, TWBT_2OF5IATA, TWBT_3OF9FULLASCII, TWBT_CODABARWITHSTARTSTOP, TWBT_MAXICODE For definitions of valid bar code types, see BarcodeInfo.

getSupportedBarcodeTypes()

The following table describes the syntax for getSupportedBarcodeTypes() method:

Syntax	Description
`public int[] getSupportedBarcodeTypes() throws OperationException, InvalidStateException;`	Returns an array of the bar code types supported by the scanner. Valid bar code types are: TWBT_3OF9, TWBT_2OF5INTERLEAVED, TWBT_2OF5NONINTERLEAVED, TWBT_CODE93, TWBT_CODE128, TWBT_UCC128, TWBT_CODABAR, TWBT_UPCA, TWBT_UPCE, TWBT_EAN8, TWBT_EAN13, TWBT_POSTNET, TWBT_PDF417, TWBT_2OF5INDUSTRIAL, TWBT_2OF5MATRIX, TWBT_2OF5DATALOGIC, TWBT_2OF5IATA, TWBT_3OF9FULLASCII, TWBT_CODABARWITHSTARTSTOP, TWBT_MAXICODE For definitions of valid bar code types, see BarcodeInfo.

getBarcodesDetected()

The following table describes the syntax for getBarcodesDetected() method:

Syntax	Description
`public BarcodeInfo[] getBarcodesDetected();`	Returns an array of BarcodeInfo objects describing the bar codes found on the current page by the scanner.

isPatchCodeDetectionEnabled()

The following table describes the syntax for isPatchCodeDetectionEnabled() method:

Syntax	Description
`public boolean isPatchCodeDetectionEnabled() throws OperationException, InvalidStateException;`	Returns whether the patch code detection is enabled for the scanner.

getPatchCodeMaxRetries()

The following table describes the syntax for getPatchCodeMaxRetries() method:

Syntax	Description
`public int getPatchCodeMaxRetries() throws OperationException, InvalidStateException;`	Returns the number of times a search will be retried if no patch codes are found.

getPatchCodeMaxSearchPriorities()

The following table describes the syntax for getPatchCodeMaxSearchPriorities() method:

Syntax	Description
`public int getPatchCodeMaxSearchPriorities() throws OperationException, InvalidStateException;`	Returns the maximum number of supported patch code search priorities.

getPatchCodeSearchMode()

The following table describes the syntax for getPatchCodeSearchMode() method:

Syntax	Description
`public int getPatchCodeSearchMode() throws OperationException, InvalidStateException;`	Returns the orientation or orientation priority used for patch code searching. Possible return values are: TWBD_HORZ, TWBD_VERT, TWBD_HORZVERT, TWBD_VERTHORZ For definitions of possible return values, see getBarcodeSearchMode().

getPatchCodeTimeout()

The following table describes the syntax for getPatchCodeTimeout() method:

Syntax	Description
`public int getPatchCodeTimeout() throws OperationException, InvalidStateException;`	Returns the maximum amount of time to spend searching for patch codes on a page.

getPatchCodeSearchPriorities()

The following table describes the syntax for getPatchCodeSearchPriorities() method:

Syntax	Description
`public int[] getPatchCodeSearchPriorities() throws OperationException, InvalidStateException;`	Returns a prioritized array of patch code types dictating the order in which they will be sought. Valid patch code types are: • TWPCH_PATCH1 - 0 • TWPCH_PATCH2 - 1 • TWPCH_PATCH3 - 2 • TWPCH_PATCH4 - 3 • TWPCH_PATCH6 - 4 • TWPCH_PATCHT - 5

getSupportedPatchCodeTypes()

The following table describes the syntax for getSupportedPatchCodeTypes() method:

Syntax	Description
`public int[] getSupportedPatchCodeTypes() throws OperationException, InvalidStateException;`	Returns an array of the patch code types supported by the scanner. Valid patch code types are: TWPCH_PATCH1, TWPCH_PATCH2, TWPCH_PATCH3, TWPCH_PATCH4, TWPCH_PATCH6, TWPCH_PATCHT For definitions of valid patch code types, see getPatchCodeSearchPriorities().

getPatchCodesDetected()

The following table describes the syntax for getPatchCodesDetected() method:

Syntax	Description
`public int[] getPatchCodesDetected();`	Returns an array containing the first patch code type found on the current page by the scanner. The return value may be null or an empty array if no patch codes were detected. Valid patch code types are: TWPCH_PATCH1, TWPCH_PATCH2, TWPCH_PATCH3, TWPCH_PATCH4, TWPCH_PATCH6, TWPCH_PATCHT For definitions of valid patch code types, see getPatchCodeSearchPriorities().

isMicrEnabled()

The following table describes the syntax for isMicrEnabled() method:

Syntax	Description
`public boolean isMicrEnabled() throws OperationException, InvalidStateException;`	Returns whether MICR detection is enabled for the scanner.

getMicrDetected()

The following table describes the syntax for getMicrDetected() method:

Syntax	Description
`public MicrInfo getMicrDetected();`	Returns a MicrInfo object describing the magnetic data found on the current page by the scanner.

setPrinter(int printer)

The following table describes the syntax for setPrinter(int printer) method:

Syntax	Description
`public void setPrinter(int printer) throws InvalidStateException, OperationException;`	Sets the current imprinter or endorser. Valid values are: • 0 - Top Pre-Imprinter • 1 - Top Post-Imprinter • 2 - Bottom Pre-Imprinter • 3 - Bottom Post-Imprinter • 4 - Top Pre-Endorser • 5 - Top Post-Endorser • 6 - Bottom Pre-Endorser • 7 - Bottom Post-Endorser

setPrinterIndex(int printerIndex)

The following table describes the syntax for setPrinterIndex(int printerIndex) method:

Syntax	Description
`public void setPrinterIndex(int printerIndex) throws InvalidStateException, OperationException`	Sets the counter value of the current imprinter or endorser.

setPrinterMode(int printerMode)

The following table describes the syntax for setPrinterMode(int printerMode) method:

Syntax	Description
`public void setPrinterMode(int printerMode) throws InvalidStateException, OperationException;`	Sets the mode of the current imprinter or endorser. Valid values are: • 0 - Single string • 1 - Multi-line string • 2 - Compound string

setPrinterString(String printerString)

The following table describes the syntax for setPrinterString(String printerString) method:

Syntax	Description
`public void setPrinterString(String printerString) throws InvalidStateException, OperationException;`	Sets the string or format of a single line to be printed to the current imprinter or endorser.

setPrinterString(String[ ] printerString)

The following table describes the syntax for setPrinterString(String[ ] printerString) method:

Syntax	Description
`public void setPrinterString(String[] printerString) throws InvalidStateException, OperationException;`	Sets the format of one or more lines (if capable) to be printed to the current imprinter or endorser.

setPrinterSuffix(String printerSuffix)

The following table describes the syntax for setPrinterSuffix(String printerSuffix) method:

Syntax	Description
`public void setPrinterSuffix(String printerSuffix) throws InvalidStateException, OperationException;`	Sets the suffix for the current imprinter or endorser. (Compound string mode only).

setBarcodeDetectionEnabled(boolean enabled)

The following table describes the syntax for setBarcodeDetectionEnabled(boolean enabled) method:

Syntax	Description
`public void setBarcodeDetectionEnabled(boolean enabled) throws OperationException, InvalidStateException;`	Enables or disables bar code detection for the scanner.

setBarcodeMaxRetries(int maxRetries)

The following table describes the syntax for setBarcodeMaxRetries(int maxRetries) method:

Syntax	Description
`public void setBarcodeMaxRetries(int maxRetries) throws InvalidStateException, OperationException;`	Restricts the number of times a search will be retried if no bar codes are found.

setBarcodeMaxSearchPriorities(int maxSearchPriorities)

The following table describes the syntax for setBarcodeMaxSearchPriorities(int maxSearchPriorities) method:

Syntax	Description
`public void setBarcodeMaxSearchPriorities(int maxSearchPriorities) throws InvalidStateException, OperationException;`	Specifies the maximum number of supported bar code search priorities.

setBarcodeSearchMode(int searchMode)

The following table describes the syntax for setBarcodeSearchMode(int searchMode) method:

Syntax	Description
`public void setBarcodeSearchMode(int searchMode) throws InvalidStateException, OperationException;`	Restricts bar code searching to certain orientations, or prioritizes one orientation over another. Valid values are: TWBD_HORZ, TWBD_VERT, TWBD_HORZVERT, TWBD_VERTHORZ For definitions of valid values, see getBarcodeSearchMode().

setBarcodeTimeout(int timeout)

The following table describes the syntax for setBarcodeTimeout(int timeout) method:

Syntax	Description
`public void setBarcodeTimeout(int timeout) throws InvalidStateException, OperationException;`	Restricts the total time spent on searching for bar codes on a page.

setBarcodeSearchPriorities(int[] value)

The following table describes the syntax for setBarcodeSearchPriorities(int[] value) method:

Syntax	Description
`public void setBarcodeSearchPriorities(int[] value) throws OperationException, InvalidStateException;`	Sets a prioritized array of bar code types dictating the order in which they will be sought. Valid bar code types are: TWBT_3OF9, TWBT_2OF5INTERLEAVED, TWBT_2OF5NONINTERLEAVED, TWBT_CODE93, TWBT_CODE128, TWBT_UCC128, TWBT_CODABAR, TWBT_UPCA, TWBT_UPCE, TWBT_EAN8, TWBT_EAN13, TWBT_POSTNET, TWBT_PDF417, TWBT_2OF5INDUSTRIAL, TWBT_2OF5MATRIX, TWBT_2OF5DATALOGIC, TWBT_2OF5IATA, TWBT_3OF9FULLASCII, TWBT_CODABARWITHSTARTSTOP, TWBT_MAXICODE For definitions of valid bar code types, see BarcodeInfo.

setPatchCodeDetectionEnabled(boolean enabled)

The following table describes the syntax for setPatchCodeDetectionEnabled(boolean enabled) method:

Syntax	Description
`public void setPatchCodeDetectionEnabled(boolean enabled) throws OperationException, InvalidStateException;`	Enables or disables patch code detection for the scanner.

setPatchCodeMaxRetries(int maxRetries)

The following table describes the syntax for setPatchCodeMaxRetries(int maxRetries) method:

Syntax	Description
`public void setPatchCodeMaxRetries(int maxRetries) throws InvalidStateException, OperationException;`	Restricts the number of times a search will be retried if no patch codes are found.

setPatchCodeMaxSearchPriorities(int maxSearchPriorities)

The following table describes the syntax for setPatchCodeMaxSearchPriorities(int maxSearchPriorities) method:

Syntax	Description
`public void setPatchCodeMaxSearchPriorities(int maxSearchPriorities) throws InvalidStateException, OperationException;`	Specifies the maximum number of supported patch code search priorities.

setPatchCodeSearchMode(int searchMode)

The following table describes the syntax for setPatchCodeSearchMode(int searchMode) method:

Syntax	Description
`public void setPatchCodeSearchMode(int searchMode) throws InvalidStateException, OperationException;`	Restricts patch code searching to certain orientations, or prioritizes one orientation over another. Valid values are: TWBD_HORZ, TWBD_VERT, TWBD_HORZVERT, TWBD_VERTHORZ For definitions of valid values, see getBarcodeSearchMode().

setPatchCodeTimeout(int timeout)

The following table describes the syntax for setPatchCodeTimeout(int timeout) method:

Syntax	Description
`public void setPatchCodeTimeout(int timeout) throws InvalidStateException, OperationException;`	Restricts the total time spent on searching for patch codes on a page.

setPatchCodeSearchPriorities(int[] value)

The following table describes the syntax for setPatchCodeSearchPriorities(int[] value) method:

Syntax	Description
`public void setPatchCodeSearchPriorities(int[] value) throws OperationException, InvalidStateException;`	Sets a prioritized array of patch code types dictating the order in which they will be sought. Valid patch code types are: TWPCH_PATCH1, TWPCH_PATCH2, TWPCH_PATCH3, TWPCH_PATCH4, TWPCH_PATCH6, TWPCH_PATCHT For definitions of valid patch code types, see getPatchCodeSearchPriorities().

setMicrEnabled(boolean enabled)

The following table describes the syntax for setMicrEnabled(boolean enabled) method:

Syntax	Description
`public void setMicrEnabled(boolean enabled) throws OperationException, InvalidStateException;`	Enables or disables MICR detection for the scanner.

TWAINSource

The TWAINSource class is an instance of ImageCaptureSource, and it represents the source used by TWAIN scanning.

Property	Type	Description
deviceSource	Source	The low-level object used to interact with a TWAIN scanner.

Capture Client FieldEdit Classes

The FieldEdit class is the user interface component for entering metadata values. This section describes the following Capture Client FieldEdit classes:

- DataField
- FloatField
- IntegerField
- PicklistEntry
- PicklistField
- TextField

DataField

The DataField class represents the data for a single field of the FieldEdit component and is the base class for the various field types.

Property	Type	Description
caption	String	The field caption.
fieldName	String	The field name.
displayFormat	Format	The display format for the field.
inputMask	String	The input mask.
fieldLock	boolean	If *True*, the field is locked.
maxLength	Long	The maximum length of the field.
required	boolean	If *True*, this field is required to be entered.
uncommittedText	String	The current text in the field.

DateField

The DataField class extends from DataField and represents a date field.

Property	Type	Description
value	Date	The value, represented by a Date object.

FloatField

This FloatField class extends from DataField and represents a float field.

Property	Type	Description
value	Float	The value, represented by a Float object.

IntegerField

The IntegerField class extends from DataField and represents an integer field.

Property	Type	Description
value	Integer	The value, represented by an Integer object.

PicklistEntry

The PicklistEntry class is used in picklist-type fields to determine the display and commit values. This class includes the following methods:

- getCommitValue()
- getDisplayValue()

getCommitValue()

The following table describes the syntax for getCommitValue() method:

Syntax	Description
`public String getCommitValue()`	Returns the commit value of the choice list entry.

getDisplayValue()

The following table describes the syntax for getDisplayValue() method:

Syntax	Description
`public String getDisplayValue()`	Returns the display value of the choice list entry.

PicklistField

The PicklistField class extends from DataField and represents a Pick-list (Choice List) field.

Property	Type	Description
pickListCaseSensitive	boolean	If *True*, the contents of this list are sensitive to case.
pickListEntries	List<PicklistEntry>	The list of entries in the choice list.
value	PicklistEntry	The current value of the choice list.

TextField

This TextField class extends from DataField and represents an alphanumeric field.

Property	Type	Description
value	String	The alphanumeric value of the field.

Sample Client Scripts

The section describes the following sample client scripts:

- Sample Client Script 1
- Sample Client Script 2
- Sample Client Script 3

Sample Client Script 1

This sample script customizes client behavior in the following ways:

- Prevents the client user from leaving a metadata field if the entry contains the word *test*.
- Prevents the user from entering an asterisk in any metadata field.

- Outputs event information to the java console, such as coordinates after a user right-mouse-drags a selection on an image.
- Stops releasing the batches to predefined processors and unlocks the batches.

Note that this script also writes out a line (printIn) to the java console for each script event, for verification or debugging purposes.

```
function ScriptStart() {
    java.lang.System.out.println("ScriptStart");
}

function PreBatchScan(event) { // BatchScanEvent
    java.lang.System.out.println("PreBatchScan");
}

function BatchScanBegin(event) { // BatchScanEvent
    java.lang.System.out.println("BatchScanBegin");
}

function BatchScanComplete(event) { // BatchScanEvent
    java.lang.System.out.println("BatchScanComplete");
    java.lang.System.out.println(event.getBatch().getBatchName() + " finished
Scanning.");
}

function BatchSelected(event) { // BatchSelectedEvent
    java.lang.System.out.println("BatchSelected: " +
event.getBatch().getBatchName());
}

function PreBatchDelete(event) { // BatchDeleteEvent
    java.lang.System.out.println("PreBatchDelete");
}

function CaptureImage(event) { // ImageCaptureEvent
    java.lang.System.out.println("CaptureImage");
}

function DocumentCreated(event) { // CaptureDocument
    java.lang.System.out.println("DocumentCreated");
}

function DocumentSelected(event) { // DocumentSelectedEvent
    java.lang.System.out.println("DocumentSelected: " +
event.getDocument().getTitle());
}

function FieldGotFocus(event) { // FieldEvent
    java.lang.System.out.println("FieldGotFocus");
}

function FieldLostFocus(event) { // FieldEvent
    var dataField;
    java.lang.System.out.println("FieldLostFocus");
    dataField = event.getField();
    if (dataField.getUncommittedText().equalsIgnoreCase("test")) {
        event.setCancel(true);
        java.lang.System.out.println("invalid value. script will not allow
leaving focus.");
    }
```

```
}

function FieldProcessKey(event) { // FieldEvent
    var keyEvent;
    // java.lang.System.out.println("FieldProcessKey");
    keyEvent = event.getKeyEvent();
    if (keyEvent.getID() == java.awt.event.KeyEvent.KEY_TYPED) {
        //java.lang.System.out.println(keyEvent.getKeyChar());
        if (new java.lang.Character(keyEvent.getKeyChar()).equals(new
java.lang.Character('*'))) {
            java.lang.System.out.println("Asterisk not allowed in any field.");
            keyEvent.consume();
        }
    }
}

function PostCaptureImage(event) { // ImageCaptureEvent
    java.lang.System.out.println("PostCaptureImage");
}

function PreCaptureImage(event) { // ImageCaptureEvent
    java.lang.System.out.println("PreCaptureImage");
}


function PreUploadItem(event) { // UploadItemEvent
    java.lang.System.out.println("PreUploadItem: " +
event.getCaptureItem().getFilename());
}

function PostUploadItem(event) { // UploadItemEvent
    java.lang.System.out.println("PostUploadItem: " +
event.getCaptureItem().getFilename());
}

function DBSearchComplete(searchEvent) { // DBSearchEvent
    java.lang.System.out.println("DBSearchComplete.");
}

function DBSearchResults(searchEvent) { // DBSearchEvent
    var results;
    var resultRow;
    var searchParameters;

    java.lang.System.out.println("DBSearchResult");

    results = searchEvent.getRowResults();
    java.lang.System.out.println("Found " + results.size() + " results.");
}

function DBSearchStart(searchEvent) {  // DBSearchEvent
    java.lang.System.out.println("DBSearchStart");
    java.lang.System.out.println("Metadata value being sought: " +
searchEvent.getMetadataValue());
}

function DocumentRemoved(event) { // DocumentRemovedEvent
    java.lang.System.out.println("DocumentRemoved");
}

function PostDownloadItem(event) { // DownloadItemEvent
```

```
    java.lang.System.out.println("PostDownloadItem: " +
event.getCaptureItem().getFilename());
}

function PreDownloadItem(event) { // DownloadItemEvent
    java.lang.System.out.println("PreDownloadItem: " +
event.getCaptureItem().getFilename());
}

function RegionSelected(event) { // RegionSelectedEvent
    java.lang.System.out.println("RegionSelected");
    var rect = event.getSelectionRectangle();
    java.lang.System.out.println("Rectangle (X,Y): (" + rect.getX() + "," +
rect.getY() +
        "); (W,H): (" + rect.getWidth() + "," + rect.getHeight() + ")");
}

function PreReleaseBatch(event) { // ReleaseBatchEvent
    // Skips the postProcess setting specified in the Capture profile and only
unlocks the batch.
    event.setProcessorID(null);
    event.setJobID(null);
}
```

Sample Client Script 2

This sample script customizes client behavior in the following ways:

- Uses the BatchScanBegin function to restrict files that can be imported to those with a .TIF extension only.
- Uses the DBSearchResults function to modify the results of a database lookup so that only the first result is used, and prevents the results list from displaying.

```
function BatchScanBegin(event) { // BatchScanEvent
    // Check if there are files being imported.
    var sourceFilesList = event.getSourceFiles();
    if (sourceFilesList != null) {
        // Create a list to hold the filtered results.
        var filteredList = new java.util.ArrayList();

        // Loop through each of the files.
        var iterator = sourceFilesList.iterator();
        while (iterator.hasNext()) {
            // If the file name ends with ".TIF", add it to the list.
            var file = iterator.next();
            var filename = file.getName().toUpperCase();
            if (filename.endsWith(".TIF")) {
                filteredList.add(file);
            }
        }

        // Replace the original list with the filtered list.
        event.setSourceFiles(filteredList);
    }
}

function DBSearchResults(searchEvent) { // DBSearchEvent
    var results;
    var resultRow;
    var searchParameters;
```

```
    // Return only the first search result.
    results = searchEvent.getRowResults();
    if (results.size() > 0) {
        resultRow = results.get(0);
        results.clear();
        results.add(resultRow);
        // Do not display the list of results to the user.
        searchEvent.setDisplayHitlist(false);
    }
}
```

Sample Client Script 3

This sample script customizes client behavior in the following ways:

- Uses the *PreReleaseBatch* event to copy metadata from the first document in a batch to the remaining documents in the batch. Copying of metadata takes place when the batch is released. The variable, *fieldsToCopy*, specifies which metadata fields are to be copied. For each document to which fields are being copied, if a field already contains a value, that value is not overwritten.
- Iterates through the documents in a batch.
- Obtains the metadata fields from a document.
- Sets the metadata field values.
- Saves the changes to a document.

```
function PreReleaseBatch(event) {
    // Metadata fields to copy to remaining documents
    var fieldsToCopy = new Array("Account Name", "Account Number");

    // Loop through each batch being released
    var batches = event.getBatches();
    for (var batchIdx = 0; batchIdx < batches.size(); batchIdx++) {
        var batch = batches.get(batchIdx);
        var documents = batch.getDocuments();
        if (documents.size() == 0)
            continue;
        // Get the first document from which we'll be copying
        var firstDocument = documents.get(0);
        var captureFields = [];
        // Create an array of the first document's fields
        var fieldDefs = batch.getWorkspace().getFieldDefinitions();
        for (var fieldIdx = 0; fieldIdx < fieldsToCopy.length; fieldIdx++) {
            var fieldID = fieldDefs.findByName(fieldsToCopy[fieldIdx]).getId();
            captureFields[fieldIdx] = firstDocument.getFields().get(fieldID);
        }
        // Loop through the remaining documents in the batch
        for (var documentIdx = 1; documentIdx < documents.size(); documentIdx++)
{
            var destDocument = documents.get(documentIdx);
            var fields = destDocument.getFields();
            // Copy the source document's field values to the destination
document
            for (var fieldIdx = 0; fieldIdx < captureFields.length; fieldIdx++) {
                var sourceField = captureFields[fieldIdx];
                // If the field in the source document was never set, skip it.
                if (sourceField == null)
                    continue;
```

```
                // Look up the field in the document
                var destField = fields.get(sourceField.getFieldName());
                // If it doesn't exist yet, create it
                if (destField == null)
                    destField = fields.add(sourceField.getFieldName());
                // If the value hasn't been set yet, set it
                if ((destField.getValue() == null) ||
(destField.getValue().isEmpty())) {
                    destField.setValue(sourceField.getValue());
                    destField.setDisplayValue(sourceField.getDisplayValue());
                }
            }
            // Save the document.
            destDocument.persist();
        }
    }
}
```

4

Creating Recognition Processor Scripts

This chapter describes creating Recognition Processor scripts.
The following are common uses for Recognition Processor scripts:

- Splitting a single bar code value into multiple field values.
- Assigning bar code value(s) to proper fields.
- Using custom logic to determine which pages constitute document separation.
- Performing custom auditing of server activity.
- Canceling the committing of a batch due to invalid data.

Capture enables you to create Recognition Processor scripts to customize recognition processing. For more information, see *Managing Oracle WebCenter Enterprise Capture*.

This chapter covers the following topics:

Recognition Processor Methods

This section provides a description of the Recognition Processor methods. Methods are executed in the following order in Recognition Processor batch jobs:

1. initialize
2. processBatch
3. restoreCaptureBatch
4. beginPhase
5. endPhase
6. extractBatchItem
7. barcodesFoundOnItem
8. batchItemAllValidBarcodes
9. determineSeparatorPage
10. batchItemValidBarcode
11. determineDocType
12. beginDatabaseLookup
13. determineIndexValues
14. renameOrigCaptureDocTitle
15. createCaptureDoc

16. postProcess
17. endBatchProcess

> **Note:**
>
> Some methods are only executed under certain job configurations.

initialize

This is the very first call the Recognition Processor makes to the script. There is no batch identified yet.

The following are the properties populated in the Recognition Processor class (rpc):

- phaseID: 0
- logger: Logger can be used to log additional entries. This property remains during the entire process, and does not repeat for every method.
- job: current Recognition Job. This property remains during the entire process, and does not repeat for every method.
- workspaceEntity: Current workspace entity. This property remains during the entire process, and does not repeat for every method.
- batchManager: BatchManager can be used to audit and manipulate batches, documents, and batch items. Use this property with caution when calling methods within BatchManager. If this property is not used properly, batch can get corrupted.

The following table describes the syntax and parameter for this method:

Syntax	Parameter
`public void initialize(RecognitionProcessorContext rpc);`	RecognitionProcessorContext rpc

processBatch

The processBatch method is called before the Recognition Processor processes the batch. The following are the properties populated in the rpc:

- phaseID: 0
- ble: At this point, the Recognition Processor has refreshed the document list for the batch. This property will remain during the remainder of the process, and will not repeat for the rest of the methods.
- cancelAction: You can set the flag to true to skip processing of a batch.
- processorBase: Represents a Dynamic Monitoring Service (DMS) Noun object that can be used to collect your own set of metrics. This property will stay through the remainder of the process and does not repeat for each method.
- DMS_Literals: Resource bundles that are being used by DMS. This property will stay through the remainder of the process and does not repeat for each method.

The following table describes the syntax and parameter for this method:

Syntax	Parameter
`public void processBatch(RecognitionProcessorContext rpc);`	RecognitionProcessorContext rpc

restoreCaptureBatch

The restoreCaptureBatch method is invoked when a batch that was processed earlier was aborted due to an error or other reasons during document creation phase. Recognition Processor must first clean up the batch to restore the batch to its original state, before initiating processing.

The restoreCaptureBatch method is invoked when all the following conditions are met:

- Batch state indicates that the Recognition Processor last failed at the document creation phase.
- Batch has not been modified since last process.
- Recognition job has not been modified since last process.

The Recognition Processor makes sure that both batch and job have not been modified since the last process. In such cases, the Recognition Processor attempts to restore the batch to its original state by removing previous documents created by the recognition process.

The following are the properties populated in the rpc:

- phaseID: 0
- cancelAction: You can set the flag to true to skip restoring of the batch, and the process skips processing this batch.

The following table describes the syntax and parameter for this method:

Syntax	Parameter
`public void restoreCaptureBatch(RecognitionProcessorContext rpc);`	RecognitionProcessorContext rpc

beginPhase

The beginPhase method indicates the beginning of a phase. The following are the properties populated in the rpc:

- phaseID: Identification of the phase. There are six different phases (see RecognitionProcessorContext for details on RecognitionProcessorContext phaseID).
- cancelAction: You can set the flag to true to skip certain phases. For phases that cannot be skipped, this flag is ignored.
  - Phases that can be canceled are: bar code recognition, document classification, and indexing.

- Phases that cannot be canceled are: document organization, document creation, and post-processing.

The following table describes the syntax and parameter for this method:

Syntax	Parameter
`public void beginPhase(RecognitionProcessorContext rpc);`	RecognitionProcessorContext rpc

endPhase

The endPhase method indicates the end of a phase. The following are the properties populated in the rpc:

- phaseID: Identification of the phase. There are six different phases (see RecognitionProcessorContext for details on RecognitionProcessorContext phaseID).

The following table describes the syntax and parameter for this method:

Syntax	Parameter
`public void endPhase(RecognitionProcessorContext rpc);`	RecognitionProcessorContext rpc

extractBatchItem

The extractBatchItem method is executed during the bar code recognition phase. The Recognition Processor extracts batch items, one at a time, into a directory right before the Recognition Processor performs bar code recognition on the page. Then the Recognition Processor informs you where the items are located.

The following are the properties populated in the rpc:

- phaseID: 1.
- extractPath: The directory where the batch item is located.

The following table describes the syntax and parameter for this method:

Syntax	Parameter
`public void extractBatchItem(RecognitionProcessorContext rpc);`	RecognitionProcessorContext rpc

barcodesFoundOnItem

The barcodesFoundOnItem method is invoked after the Recognition Processor processed the batch item, collected and recognized bar codes on this item.

The following are the properties populated in the rpc:

- phaseID: 1.
- batchItem: Current batch item that is used to perform bar code recognition.

- patchCodeRead: Patch code value found on the batch item.
- barCodesRead: A combination of bar codes read on the page and existing bar codes on the batch item.

The following table describes the syntax and parameter for this method:

Syntax	Parameter
`public void barcodesFoundOnItem(RecognitionProcessorContext rpc);`	RecognitionProcessorContext rpc

batchItemAllValidBarcodes

The batchItemAllValidBarcodes method is called after the Recognition Processor has finished validating bar codes on a specific batch item.

The following are the properties populated in the rpc:

- phaseID: 2.
- batchItem: Current batch item that is used to perform bar code validation.
- validBarCodes: A list of name and value pairs of the valid bar codes found on the batch item. This list includes all bar codes definitions in the recognition job. You can change the value, but you must not change the name, or add or remove items from the list.

The following table describes the syntax and parameter for this method:

Syntax	Parameter
`public void batchItemAllValidBarcodes(RecognitionProcessorContext rpc);`	RecognitionProcessorContext rpc

determineSeparatorPage

The determineSeparatorPage method is called after the Recognition Processor has validated a page as a separator. This method is only invoked if a separator page is defined for a recognition job.

The following are the properties populated in the rpc:

- phaseID: 2.
- batchItem: Current batch item that is to determine whether the page is a separator or not.
- validBarCodes: A list of name and value pairs for the valid bar codes found on the batch item. This list includes all bar code definitions in the recognition job.
- separator: This object is null unless this batch item is a valid separator page. If you want to make changes, you must either set the separator to null or to a valid object of class ProcessSeparatorPage.

Recognition Processor's hierarchical separator feature processes and organizes documents within a hierarchy of levels. You can change the level determined by the Recognition Processor. However, if the level does not fit into a recognition job

definition, the Recognition Processor uses either the lowest level (level<=0) or highest level (level>=max defined level). The level property of the separator object is used for the hierarchy separator page type only. For any other document organization type, this value is ignored. Level should always begin with 1.

The following table describes the syntax and parameter for this method:

Syntax	Parameter
`public void determineSeparatorPage(RecognitionProcessorContext rpc);`	RecognitionProcessorContext rpc

batchItemValidBarcode

The batchItemValidBarcode method passes in one valid bar code recognized on a specific batch item. This method call will only happen when the document organization type is *Same bar code value on each page* and *Optimize Bar Code Recognition* is turned on.

When the Recognition Processor cannot find a bar code on a page, it will try to determine the separator bar code value on the next page. validBarcode is populated with the bar code found on the next page. If bar code is not found, validBarcode is set to null. In such cases, this method is called right after the Recognition Processor has determined the bar code value.

The following are the properties populated in the rpc:

- phaseID: 2.
- batchItem: Next page batch item that is to determine the separator bar code value.
- validBarcode: Name and value pair for the separator bar code. You can change the value if required.

The following table describes the syntax and parameter for this method:

Syntax	Parameter
`public void batchItemValidBarcode(RecognitionProcessorContext rpc);`	RecognitionProcessorContext rpc

determineDocType

The determineDocType method is called after the Recognition Processor has identified a document type as either the default document type or one of the dynamic document type mappings. docTypeID can be null if the Recognition Processor is unable to identify it.

The following are the properties populated in the rpc:

- phaseID: 3.
- document: Contains the current document information. Some properties are specific to certain document organization type. You can modify the document's metadata values by using the indexValues property of the document object.

The following table describes the syntax and parameter for this method:

Syntax	Parameter
`public void determineDocType(RecognitionProcessorContext rpc);`	RecognitionProcessorContext rpc

beginDatabaseLookup

The beginDatabaseLookup method is called after the Recognition Processor has determined the lookup value, and before the actual execution of the lookup is called.

The following are the properties populated in the rpc:

- phaseID: 4.
- dbLookupValue: You can modify the lookup value.
- cancelAction: You can cancel lookup.

The following table describes the syntax and parameter for this method:

Syntax	Parameter
`public void beginDatabaseLookup(RecognitionProcessorContext rpc);`	RecognitionProcessorContext rpc

determineIndexValues

The determineIndexValues method is called after the Recognition Processor has determined all metadata values for a particular document. You can modify the metadata values.

The following are the properties populated in the rpc:

- phaseID: 4.
- document: Contains the current document information. Some properties are specific to certain document organization types. You can modify the metadata values of the document by using the indexValues property of the document object.

The following table describes the syntax and parameter for this method:

Syntax	Parameter
`public void determineIndexValues(RecognitionProcessorContext rpc);`	RecognitionProcessorContext rpc

renameOrigCaptureDocTitle

The renameOrigCaptureDocTitle method is called before the Recognition Processor renames the original document as *unindexed*. This applies to all document organization types except the *Do not perform document organization* type.

The following are the properties populated in the rpc:

- phaseID: 5.
- unIndexedDocTitle: You can change the title.

The following table describes the syntax and parameter for this method:

Syntax	Parameter
`public void renameOrigCaptureDocTitle(RecognitionProcessorContext rpc);`	RecognitionProcessorContext rpc

createCaptureDoc

Before the Recognition Processor creates the Capture document, it is possible to customize the document title, document type id, metadata values, and document comments. You can also change the batch items associated with this document, although in the case of the *Do not perform document organization* type, changing batch items does not affect the outcome.

> **Note:**
>
> You must be careful while changing batch items as it may possibly leave orphan items in the batch, that are not associated with any documents.

The following are the properties populated in the rpc:

- phaseID: 5.
- document: Capture document that the Recognition Processor is about to create.

The following table describes the syntax and parameter for this method:

Syntax	Parameter
`public void createCaptureDoc(RecognitionProcessorContext rpc);`	RecognitionProcessorContext rpc

postProcess

The postProcess method is invoked after the Recognition Processor has determined all post-process settings, but before any actual changes take place.

The following is the property populated in the rpc:

- phaseID: 6.

The following table describes the syntax and parameter for this method:

Syntax	Parameter
`public void postProcess(RecognitionProcessorContext rpc, PostProcessContext ppc);`	RecognitionProcessorContext rpc PostProcessContext ppc

endBatchProcess

The endBatchProcess method indicates that the Recognition Processor has finished processing the batch.

The following is the property populated in the rpc:

- phaseID: 0

The syntax for this method is: `public void endBatchProcess(RecognitionProcessorContext rpc);`

Recognition Processor Classes

The Recognition Processor classes can be used to design Recognition Processor scripts. This section describes the following classes:

- BarcodeDefinition
- DocumentDefinition
- PostProcessContext
- ProcessorAttachment
- ProcessorDocument
- ProcessorItem
- ProcessSeparatorPage
- RecognitionJob
- RecognitionJobField
- RecognitionProcessorContext
- SeparatorDefinition
- SeparatorRuleDefinition

In addition to the following classes that can be used to design Recognition Processor scripts, there are some common classes that pertain to the Recognition Processor and the Import Processor. For more information on the common classes, see Common Capture Classes.

BarcodeDefinition

The BarcodeDefinition class contains the constants for the bar code validation rule type. This class represents one bar code definition specified in the third train stop of Recognition Processor Job.

Property	Type	Description
barcodeName	String	Name of the bar code definition .
validationRule	Integer	Bar code validation rule; values are 0-4, as defined in the constants. The following are the constants for the bar code validation rule type: • 0 – Does not have a validation rule specified. • 1 – Uses the bar code length to validate. • 2 – Uses the mask to validate. • 3 – Uses a regular expression to validate. • 4 – Uses a choice list to validate.
validationLength	Integer	Validation length.
validationMask	String	Validation mask.
validationRegularExpression	String	Validation regular expression.
pickListSourceID	String	Validation choice list source identifier.
pickListID	String	Validation choice list identifier.

DocumentDefinition

When a document profile is set to *Determine dynamically using bar code*, you can define *Document Profile and Bar Code Value Mappings*. Each mapping is represented by a DocumentDefinition class.

Property	Type	Description
docTypeID	String	Unique identifier of the Document Type.
mappingType	Integer	This sets whether to determine document type based on a literal value or a choice list. The values are 0 and 1, as defined in the constants: • 0 – To compare bar code value detected with a literal value specified. • 1 – The document type mapping option that determines document type based on values in the choice list.
value	String	Literal string specified.
pickListSourceID	String	Choice list source ID specified.
pickListID	String	Choice list ID specified.

PostProcessContext

The PostProcessContext class represents all the settings needed to apply to a batch after processing is completed. If there is any error during processing, PostProcessContext data is populated from the setting of Post Process train stop of Recognition Processor Job.

Property	Type	Description
renameBatch	String	Name that the batch will be renamed to during post process. If null, the batch will not be renamed.
priority	int	Priority that the batch will be changed to during post process. If the priority is not valid (<0 or >10), the batch priority will remain the same.
status	BatchStatusEntity	Status entity object that the batch will be associated with during post process. If null, the batch status will remain the same.
batchState	int	If there were some errors during the recognition process, the batch state will be preset to 16; otherwise, the batch state will be preset to 1.
emailRecipients	List<String>	A list of email recipients that email notification will be sent to. If empty, email will not be sent.
emailSubject	String	Subject line of the email notification.
emailMessage	String	Main message body of email notification. If empty, email will not be sent.
processorID	String	The Recognition Processor ID to which the current batch will be released.
processorJobID	String	The Recognition Processor job ID to which the current batch will be released.
comment	String	Represents the batch note.
errorMessage	String	Error that occurred during the batch processing.

ProcessorAttachment

The ProcessorAttachment class is a representation of a logical capture document attachment that Recognition Processor has identified. In the last phase of the document creation phase, the Recognition Processor attempts to create document attachment to the associated capture documents, based on a collection of ProcessorAttachments.

Property	Type	Description
attachmentTypeID	String	Unique identifier of the attachment type.
batchItems	List<String>	Batch items associated with this attachment.
separator	ProcessSeparatorPage	Separator page for this attachment. This property only applies to hierarchy separator organization type.
title	String	Title for this attachment.

ProcessorDocument

The ProcessorDocument class is a representation of a logical capture document that Recognition Processor has identified. In the last phase of the document creation phase, the Recognition Processor attempts to create documents within the batch, based on a collection of ProcessorDocument.

Property	Type	Description
title	String	Title of the document, which is populated during the document creation phase.
batchItems	List<String>	All batch items associated with this document. This is populated during the document organization phase.
validBarcodes	List<ProcessorItem>	Valid bar codes associated with this document. This is a combination of all valid bar codes found for all batch items associated with this document. This is populated during the document organization phase.
failureStatus	int	Status of the current document: • 0 – No error • 1 – Failed to validate bar code. This is the case when the Recognition Processor finds duplicate bar codes in a document that matches the bar code validation rule, and the job setting is to clear the value. • 2 – Document exceeded maximum page rule. • 3 – Unable to determine document type. • 4 – No database search result found, and job setting is to prevent commit when no record is found.
docTypeID	String	Document type ID associated with the document. If null, the document type has not been determined.
comment	String	Comments for the document. It is usually error detail for 'failureStatus,' which you can customize through script.
captureDocID	String	This is only used in the *Do not perform document organization* type, where the Recognition Processor does not organize documents, and does not create any Capture documents. This ID is the Capture document ID.
separator	ProcessSeparatorPage	Separator page of this document. This applies to the *Do not perform document organization* and *multiple page document with separator* organization types.

Property	Type	Description
hierarchySeparators	List<ProcessSeparatorPage>	Separator pages for this document. This applies to the *multiple pages with hierarchy separator* organization type.
indexValues	List<IndexValue>	List of metadata names and values.
attachments	List<ProcessorAttachment>	A list of attachments associated with this document.

ProcessorItem

The ProcessorItem class is a representation of an item identified by Name and Value properties. This class holds the name and value pair for a metadata field. In this case, this class holds a particular bar code's name and value.

Property	Type	Description
name	String	Indicates the bar code name for the ProcessorItem.
value	String	Specifies the value for the ProcessorItem.

ProcessSeparatorPage

The ProcessSeparatorPage class represents a separator page that has been identified by Recognition Processor.

Property	Type	Description
include	boolean	Indicates whether this separator page will be deleted after commit.
level	int	This is only used in the hierarchy separator pages organization type. Level always starts with 1.
name	String	Separator page name.
batchItemID	String	The batch item with which this separator page is associated.
validBarcodes	List<ProcessorItem>	Used only in hierarchical separator pages that holds all the valid bar codes for this batch item.
attachmentTypeID	String	This is the attachment type ID that is associated with the separator page that has been detected by the Recognition Processor.

RecognitionJob

The RecognitionJob class represents a Recognition Process Job and contains the constants for the bar code symbologies.

Property	Type	Description
workspaceName	String	Name of the workspace with which this job is associated.
workspaceID	String	The unique identifier of the workspace with which this job is associated.
jobID	String	The unique identifier of the job .
lastModifiedDateTime	Date	Date and time the job was last modified.
lastModifiedUserID	String	ID of the user that last modified the job.
jobName	String	The name of the job.
description	String	The description of the job.
scriptID	String	The identifier of the script with which this job is associated.
barcodes	List<BarcodeDefinition>	List of bar code definitions.
autoDetectBarcodes	Boolean	Determines whether Enable Auto-detect Bar Codes is turned on.
validateCheckSum	Boolean	Determines whether Validate Optional Checksum is turned on.
symbologies	List<Integer>	A list of selected bar code symbologies for recognition: values are from 0 - 21, as defined in the constants for bar code symbologies earlier in this section. The constants for the bar code symbologies are as follows: • 0 – codabar • 1 – code 128 • 2 – code 39 • 3 – code 93 • 4 – EAN-13 • 5 – EAN-8 • 6 – interleaved 2/5 • 7 – UCC/EAN 128 • 8 – UPC-A • 9 – UPC-E • 10 – Airline(IATA) 2/5 • 11 – Code 32 • 12 – Datalogic 2/5 • 13 – Industrial 2/5 • 14 – ISBN Addon 2 • 15 – ISBN Addon 5 • 16 – Matrix 2/5 • 17 – Postnet/Planet • 18 – Patch Code • 19 – Data Matrix • 20 – PDF417 • 21 – QR code

Property	Type	Description
batchOrganization	Integer	Document organization type; values ranges from 0 - 4. The following are constants for the document organization type: • 0 – Fixed number of pages per document. • 1 – (None) Do not perform document organization. • 2 – Same bar code value on each page. • 3 – Separator pages • 4 – Hierarchical separator pages.
documentPageCount	Integer	For the *Fixed number of pages per document* document organization type, this property refers to the maximum number of pages per document.
pagesPerDoc2ReadBarcodes	Integer	For the *None: Do not perform document organization* document organization type, this property refers to the number of pages per document to read bar codes.
maxPageCountPerDoc	Integer	For the *Same bar code value on each page, or Separator pages* document organization type, this property refers to the maximum number of pages per document.
multiPageDocBarcode	BarcodeDefinition	For the *Same bar code value on each page* document organization type, this property refers to the bar code that determines document separation.
optimizeBarcodeDetection	Boolean	For the *Same bar code value on each page* document organization type, this property determines whether to optimize bar code detection.
coverPages	List<SeparatorDefinition>	For the *Separator pages, Hierarchical separator pages, None: Do not perform document organization* document organization type, this property holds the data that defines the separator page. When the hierarchical separator page is used, the list may contain more than one separator page definition, while in the other two scenarios, the list will only contain one separator page definition.
multiBarcodeValuesOption	Integer	Actions to take if more than one value is found for a bar code within a document; values are 0-2 as defined in the constants. The following are actions to take when multiple bar code values are found for a bar code definition: • 0 – Use the first bar code value found. • 1 – Use the last bar code value found. • 2 – Do not use the bar code values.

Property	Type	Description
dynamicDocType	Integer	Options on how the Dynamic Document Profile is determined; values are 0-2 as defined in the constants. The following values show how the document type is dynamically determined: • 0 – The document type is not dynamically determined. • 1 – The document type is dynamically determined based on a bar code value. • 2 – The document type is dynamically determined based on a separator page.
defaultDocTypeID	String	The identifier for the Default Document Profile.
docTypeBarCode	BarcodeDefinition	When the Document Profile is being dynamically determined using the bar code, this property represents the selected bar code.
docTypeMappings	List<DocumentDefinition>	When the Document Profile is being dynamically determined using the bar code, this mapping represents the Document Profile and Bar Code Value Mappings.
jobFields	List<RecognitionJobField>	Field mappings information.
dblookupUsing	Integer	Type of value the database lookup will be using; values are 0-2 as defined in the constants. The following are values used by database lookup: • 0 – No database lookup is configured. • 1 – Use a bar code value to perform database lookup. • 2 – Use the index field value to perform database lookup.
dblookupBarcodeField	BarcodeDefinition	Bar code definition that is selected for database lookup.
dblookupIndexDefID	String	Metadata field ID that is selected for database lookup.
dblookupProfile	String	Database lookup profile ID.
dblookupSearchField	String	Database lookup search field ID.
dblookupMultipleRecordAction	Integer	Actions to take when more than one record is found during database lookup; values are 0-1 as defined in the constants. The following show actions to take when a database lookup finds multiple records: • 0 – Use the first record found during database lookup. • 1 – Do not populate the database lookup result.

Property	Type	Description
dblookupNoMatchAction	Integer	Actions to take when no record is found during database lookup; values are 0-1 as defined in the constants. The following show what action to take when a database lookup finds no match: • 0 – Permit the batch to be committed even when no database record is found. • 1 – Do not allow the batch to be committed when no match is found.
renamePrefix	String	Part of post-process setting. When there is no system error, this is the batch prefix to rename, if required.
renameEmail	String	Part of post-process setting. When there is no system error, this is the email address to send email notification to rename, if required.
renameStatus	String	Part of post-process setting. When there is no system error, this is the batch status to change, if required.
renamePriority	Integer	Part of post-process setting. When there is no system error, this is the batch priority to change, if required.
processorID	String	Part of post-process setting. When there is no system error, this is the batch processor ID to which the batch will be released.
processorJobID	String	Part of post-process setting. When there is no system error, this is the batch processor job ID to which the batch will be released.
failureRenamePrefix	String	Part of post-process setting. When there is a system error, this is the batch prefix to rename, if required.
failureRenameEmail	String	Part of post-process setting. When there is a system error, this is the email address to which notification should be sent, if required.
failureRenameStatus	String	Part of post-process setting. When there is a system error, this is the batch status to change, if required.
failureRenamePriority	Integer	Part of post-process setting. When there is a system error, this is the batch priority to change, if required.
failureProcessorID	String	Part of post-process setting. When there is a system error, this is the batch processor ID to which the batch will be released.
failureProcessorJobID	String	Part of post-process setting. When there is a system error, this is the batch processor job ID to which the batch will be released.
online	boolean	Indicates whether this recognition job is active or not.

Property	Type	Description
sourceDocAttachments	Integer	Options for source document attachments. The following are the possible values: • 0 – Include all attachments to create documents. • 1– Include attachments with matching Document Profile attachment types. • 2 – Do not include attachments.

RecognitionJobField

The RecognitionJobField class represents each field in the *Fields* train stop.

Property	Type	Description
indexDefID	String	Metadata ID to populate with property values.
autoPopulate	Integer	Auto-populate type; values are 0-5, as defined in the constants. The following are the constants for the auto-populate type: • 0 – Does not auto-populate the index value. • 1 – Auto-populates the index value with the bar code value. • 2 – Auto-populates the index value with the batch name. • 3 – Auto-populates the index value with a default value. • 4 – Auto-populates the index value with the index date. • 5 – Auto-populates the index value with the scan date.
populateValue	String	For the bar code type, this represents the bar code definition name; for the default type, this represents a default value.

RecognitionProcessorContext

The RecognitionProcessorContext class is a context object that contains relevant attributes that relates to the recognition processing.

property	Type	Description
logger	Logger	An instance of java.util.logging.Logger that can be used to log additional entries.
job	RecognitionJob	Current job being used.
ble	BatchLockEntity	A lock entity which contains the batch currently being processed.
workspaceEntity	CaptureWorkspaceEntity	Current workspace that is being used.

property	Type	Description
phaseID	int	An integer that identifies the current phase: • 0 – pre batch process. In this step, Recognition Processor performs resource initialization, batch validation, and clean up if required. • 1 – bar code recognition. In this step, Recognition Processor goes through all batch items for all documents, extracts batch items one at a time, and performs bar code recognition based on recognition settings. • 2 – document organization. In this step, Recognition Processor finds valid bar codes based on barcode definition configuration, and creates logical documents based on document processing settings. • 3 – document classification. In this step, Recognition Processor determines property document type for each logical document created in previous step based on Document Profile settings. • 4 – indexing. In this step, Recognition Processor performs database lookup based on database lookup configuration, and determines index values for all logical documents based on fields settings. • 5 – document creation. In this step, Recognition Processor creates actual capture documents based on the logical documents determined, populates document indexes, and assigns document type. If any warnings or errors occurred during process, document comments are updated. • 6 – post processing. In this step, Recognition Processor releases a batch according to post processing configuration. Batch may also get renamed, batch status and priority changed, and email message sent if required.
cancelAction	boolean	In certain calls, the user is allowed to cancel the action (for example, bar code recognition or database lookup).
batchItem	BatchItemEntity	Current batch item being processed. This is specifically used during bar code recognition and bar code validation (part of the document organization phase).
patchCodeRead	Integer	Patch code found on a batch item. This is only used during the bar code recognition phase.

property	Type	Description
barcodesRead	List<String>	All bar codes associated with a batch item, which includes original bar codes associated with the batch item, and bar codes read through the bar code recognition engine. This is only used during the bar code recognition phase.
validBarcodes	List<ProcessorItem>	List of valid bar codes found for a specific batch item. This only applies to the bar code validation step (part of the document organization phase). ProcessorDocument also contains a list of valid bar codes, which is associated with a specific document. It is a collection of all valid bar codes found on all batch items associated with the document.
validBarcode	ProcessorItem	Specific to the bar code that determines document separation and optimized bar code recognition setting. If batch organization type is bar code on every page, optimized recognition is turned on, and the barcode on a given page is null (barcode not found), then validBarcode contains the barcode for the following page.
separator	ProcessSeparatorPage	Specific for organization types that involve a separator page. If the separator is null, then this batch item is not a separator page.
document	ProcessorDocument	Used for the document classification, indexing, and document creation phase. It contains everything you must know about the document.
dbLookupValue	String	Used only before database lookup is executed. You can change the lookup value.
unIndexedDocTitle	String	Specific to the Document Creation phase. The first capture document holds all batch items for which the Recognition Processor is unable to determine the document they belong to. This property allows you to customize the first Capture document title. The default title is *unindexed*; if this value is null, then the first document title will remain unchanged.
extractPath	String	Path to which batch items were extracted. This is specific during the bar code recognition phase. You should not modify this property.
processorBase	Noun	DMS Noun that holds the Recognition Processor metrics data.
DMS_Literals	ResourceBundle	Resource bundle that is being used by DMS.

property	Type	Description
batchManager	BatchManagerSession	An instance of oracle.odc.batchmanager.BatchManagerSession that can be used to perform batch related operations.

SeparatorDefinition

The SeparatorDefinition class represents the definition on what is considered a separator page.

Property	Type	Description
name	String	Name of the separator page.
deleteUponCommit	Boolean	Determines whether to delete the separator page after commit.
operator	Integer	Operator used for rules; values are 0 and 1: • 0 – The OR operator, used in cover page definition rules. For rules separated using this operator, any one rule must match the rule condition. • 1 – The AND operator, used in cover page definition rules. For rules separated using this operator, all rules must match the rule condition.
docTypeID	String	If the document type is dynamically determined based on a separator page, this is the ID of the document type for this separator page.
rules	List<SeparatorRuleDefinition>	Collection of rules associated with this separator page.

SeparatorRuleDefinition

The SeparatorRuleDefinition class represents one rule that applies to a separator definition.

Property	Type	Description
name	String	Name of the rule.
operator	Integer	Operator used for patch code and bar codes selected; values are 0 and 1. For more information, see SeparatorDefinition.
patchCode	String	Patch code selected for this rule.
barcodes	List<String>	Bar codes selected for this rule.

Sample Recognition Processor Script

The following steps are involved in configuring a batch job:

- Set the job to detect PDF417 bar codes.
- Set the PDF417 bar code on the page to be `| delimited`, and has 10 fields concatenated together.
- Define three bar code definitions: processorDate, Title, and Amount (with no validation rules).
- Map the three bar code definitions to three index fields.

The Recognition Processor script parses a PDF417 bar code found on a batch item, parses the value, and applies appropriate parsed text to the three bar code definitions. This sample script allows you to modify the processing behavior based on the job configuration steps:

```
function batchItemAllValidBarcodes (rpc) {
        // Obtain current batch item
        var batchItem = rpc.getBatchItem();

        // obtain bar code count.
        var count = batchItem.getBarcodeCount();

        // All barcodes on a batch item.
    var allBarcodes;

        // bar code of interest.
    var barcodeValue;

        // after parsed barcode value.
        var parsed;

        // Obtain bar code value if there is a bar code found.
        if (count > 0) {
                allBarcodes = batchItem.getBarcodes();
                barcodeValue = allBarcodes[0];

                // Parse the bar code value by | character.
                var regex = "|";
                  parsed = barcodeValue.split(regex);
                var len = parsed.length;


                // It should get splitted into 10 strings.
                if (len == 10) {
                        // This is the barcode we want, populate valid bar codes.
                        populateValues(rpc, parsed);
                }

        }
}

function populateValues(rpc, parsed) {
        var valid = rpc.getValidBarcodes();
        var i;

        for (i=0; i<valid.size(); i++) {
                var bar = valid.get(i);

                if (bar.getName() == "processDate") {
                        bar.setValue(parsed[5]);
                } else if (bar.getName() == "Title") {
                        bar.setValue(parsed[6]);
```

```
                } else if (bar.getName() == "Amount") {
                        bar.setValue(parsed[4]);
                }
        }
}
```

5

Creating Import Processor Scripts

This chapter describes creating Import Processor scripts.
You can develop scripts for the Import Processor to perform a wide variety of functions. Some common tasks include:

- Skipping the importing of certain image files
- Changing Capture batch properties
- Skipping the importing of a batch
- Adding page level metadata values during importing
- After importing, moving images to a different folder

Capture enables you to create Import Processor scripts to customize the importing process. For more information, see *Managing Oracle WebCenter Enterprise Capture*.

This chapter contains the following sections:

Import Processor Events

Import Processor scripts are JavaScript modules that enable you to customize the behavior of certain Import Processor events.

This section describes the following Import Processor events:

- postRelease
- preDatabaseSearch
- processDatabaseSearchResults

preProcess

The preProcess event occurs prior to the pre-processing of the import source. Initialization code can be performed here. The processing can be canceled by setting the cancel property to True in the *ctx* parameter.

Syntax	Parameter
`public void preProcess(ImportProcessorContext ctx);`	ImportProcessorContext ctx

process

The process event signals the start of the import process.

Syntax	Parameter
`public process(ImportProcessorContext ctx);`	ImportProcessorContext ctx

postProcess

The postProcess event occurs after the import source has been processed.

Syntax	Parameter
`public void postProcess(ImportProcessorContext ctx);`	ImportProcessorContext ctx

preCreateBatch

The preCreateBatch event occurs prior to a new batch being created. The batch creation can be canceled by setting the cancel property to True in the *ctx* parameter.

Syntax	Parameter
`public void preCreateBatch(ImportProcessorContext ctx);`	ImportProcessorContext ctx

postCreateBatch

The postCreateBatch event occurs immediately after a batch is created, but before any documents have been created.

Syntax	Parameter
`public void postCreateBatch(ImportProcessorContext ctx);`	ImportProcessorContext ctx

preCreateDocument

The preCreateDocument event occurs prior to a new document being created. The document creation can be canceled by setting the cancel property to True in the *ctx* parameter.

Syntax	Parameter
`public void preCreateDocument(ImportProcessorContext ctx);`	ImportProcessorContext ctx

postCreateDocument

The postCreateDocument event occurs after a new document has been created.

Syntax	Parameter
`public void postCreateDocument(ImportProcessorContext ctx);`	ImportProcessorContext ctx

preImportFile

The preImportFile event occurs prior to a file being imported. The importing of files can be canceled by setting the cancel property to True in the *ctx* parameter.

Syntax	Parameter
`public void preImportFile(ImportProcessorContext ctx);`	ImportProcessorContext ctx

postImportFile

The postImportFile event occurs after a file is imported.

Syntax	Parameter
`public void postImportFile(ImportProcessorContext ctx);`	ImportProcessorContext ctx

preRelease

The preRelease event occurs prior to a batch being released. The releasing of a batch can be canceled by setting the cancel property to True in the *ctx* parameter.

Syntax	Parameter
`public void preRelease(ImportProcessorContext ctx);`	ImportProcessorContext ctx

postRelease

The postRelease event occurs after a batch has been released.

Syntax	Parameter
`public void postRelease(ImportProcessorContext ctx);`	ImportProcessorContext ctx

preDatabaseSearch

The preDatabaseSearch event occurs prior to a database lookup. A database search can be canceled by setting the cancelDBSearch property to True in the *ctx* parameter.

Syntax	Parameter
`public void preDatabaseSearch(ImportProcessorContext ctx);`	ImportProcessorContext ctx

processDatabaseSearchResults

The processDatabaseSearchResults event occurs after the database lookup has returned the search results.

Syntax	Parameter
`public void processDatabaseSearchResults(ImportProcessorContext ctx);`	ImportProcessorContext ctx

Email Source Events

This section describes the following email source events:

- deleteMessage
- moveMessage
- newAttachment
- newMessage

> **Note:**
>
> If you select the import source for emails as Microsoft Exchange Web Service, then you should invoke corresponding (getExchange) methods in the script. See ImportProcessorContext and EmailSourceContext classes for information on new methods that have been introduced.

deleteMessage

The deleteMessage event occurs in the email message post-processing step when an email message is about to be deleted. To prevent the email message from being deleted, set the cancel property to True in the *ctx* parameter.

Syntax	Parameter
`public void deleteMessage(ImportProcessorContext ctx, EmailSourceContext emailCtx);`	ImportProcessorContext ctx EmailSourceContext emailCtx

moveMessage

The moveMessage event occurs in the email message post-processing step when an email message is about to be moved to an email folder. To prevent the email message from being moved, set the cancel property to True in the *ctx* parameter.

Syntax	Parameter
`public void moveMessage(ImportProcessorContext ctx, EmailSourceContext emailCtx);`	ImportProcessorContext ctx EmailSourceContext emailCtx

newAttachment

The newAttachment event occurs when a new email attachment is about to be processed. To prevent the attachment from being imported, set the cancel property to True in the *ctx* parameter.

Syntax	Parameter
`public void newAttachment(ImportProcessorContext ctx, EmailSourceContext emailCtx);`	ImportProcessorContext ctx EmailSourceContext emailCtx

newMessage

The newMessage event occurs when a new email message is about to be processed. To prevent the email message from being imported, set the cancel property to True in the *ctx* parameter.

Syntax	Parameter
`public void newMessage(ImportProcessorContext ctx, EmailSourceContext emailCtx);`	ImportProcessorContext ctx EmailSourceContext emailCtx

Folder Source Events

This section describes the following folder source events:

- deleteDocumentFile
- newFolder
- renameDocumentFile

deleteDocumentFile

The deleteDocumentFile event occurs in the folder post-processing step when a file from the folder is about to be deleted. To prevent the document file from being deleted, set the cancel property to True in the *ctx* parameter.

Syntax	Parameter
`public void deleteDocumentFile(ImportProcessorContext ctx, FolderSourceContext folderCtx);`	ImportProcessorContext ctx FolderSourceContext folderCtx

newFolder

The newFolder event occurs when a new folder is about to be processed. To exclude this folder from being processed, set the cancel property to True in the *ctx* parameter.

Syntax	Parameter
`public void newFolder(ImportProcessorContext ctx, FolderSourceContext folderCtx);`	ImportProcessorContext ctx FolderSourceContext folderCtx

renameDocumentFile

The renameDocumentFile event occurs in the folder post-processing step when a file from the folder is about to be renamed. To prevent the document file from being renamed, set the cancel property to True in the *ctx* parameter.

Syntax	Parameter
`public void renameDocumentFile(ImportProcessorContext ctx, FolderSourceContext folderCtx);`	ImportProcessorContext ctx FolderSourceContext folderCtx

List File Source Events

This section describes the following list file source events:

- deleteListFile
- newFolder
- newListFile
- newListFileLine
- renameListFile

deleteListFile

The deleteListFile event occurs in the list file post-processing step when a list file is about to be deleted. To prevent the list file from being deleted, set the cancel property to True in the *ctx* parameter.

Syntax	Parameter
`public void deleteListFile(ImportProcessorContext ctx, ListFileSourceContext listFileCtx);`	ImportProcessorContext ctx ListFileSourceContext listFileCtx

newFolder

The newFolder event occurs when a new folder containing list files is about to be processed. To exclude the folder from being processed, set the cancel property to True in the *ctx* parameter.

Syntax	Parameter
`public void newFolder(ImportProcessorContext ctx, ListFileSourceContext listFileCtx);`	ImportProcessorContext ctx ListFileSourceContext listFileCtx

newListFile

The newListFile event occurs when a new list file is about to be processed. To prevent the list file from being processed, set the cancel property to True in the *ctx* parameter.

Syntax	Parameter
`public void newListFile(ImportProcessorContext ctx, ListFileSourceContext listFileCtx);`	ImportProcessorContext ctx ListFileSourceContext listFileCtx

newListFileLine

The newListFileLine event occurs when a new line in the list file is about to be processed. To prevent the list file line from being processed, set the cancel property to True in the *ctx* parameter.

Syntax	Parameter
`public void newListFileLine(ImportProcessorContext ctx, ListFileSourceContext listFileCtx);`	ImportProcessorContext ctx ListFileSourceContext listFileCtx

renameListFile

The renameListFile event occurs in the list file post-processing step when a list file is about to be renamed. To prevent the list file from being renamed, set the cancel property to True in the *ctx* parameter.

Syntax	Parameter
`public void renameListFile(ImportProcessorContext ctx, ListFileSourceContext listFileCtx);`	ImportProcessorContext ctx ListFileSourceContext listFileCtx

Import Processor Classes

This section describes the following Import Processor classes:

- EmailSourceContext
- FolderSourceContext
- ImportJob
- ImportProcessorContext
- ListFileSourceContext

In addition to the following event classes that can be used to design Import Processor scripts, there are some common classes that pertain to the Recognition Processor and the Import Processor. For more information on the common classes, see Common Capture Classes.

EmailSourceContext

The EmailSourceContext class contains all classes used in the processing of an email source.

Property	Type	Description
account	String	Name of the email account currently being processed.
attachmentFilename	String	File name of the email message attachment currently being processed.
Standard IMAP connection uses the following properties:		
folder	Folder	Email folder currently being processed.
message	Message	Email message currently being processed.
EWS connection uses the following properties:		
getExchangeMessage	microsoft.exchange.webservices.data.core.service.item.EmailMessage	Email message currently being processed.
getExchangeFolder	microsoft.exchange.webservices.data.core.service.folder.Folder	Email folder currently being processed.

For more information on the Folder and Message class definitions, see the Javamail API documentation.

FolderSourceContext

The FolderSourceContext class contains all classes used in the processing of a folder source.

Property	Type	Description
folderName	String	Name of the directory currently being processed.
documentFilename	String	Name of the file currently being processed.
renamedDocumentFilename	String	If the post-processing step indicates the file should have a prefix added to it or the extension changed, this property indicates the changed file name.

ImportJob

Import jobs are configured within a Capture Workspace to import batches from import sources such as a file system folder, a delimited list file, or an inbox/folder of an email server.

Property	Type	Description
jobID	String	A value that uniquely identifies the job in the system.
workspaceID	String	Identifier of the workspace to which the job belongs.
jobName	String	A human-readable name for the job.
dbSearchID	String	Identifier of the database search to use when processing the job.
dbSearchFieldID	String	Identifier of the database search field to use when processing the job.
imageDownsample	Integer	Determines how to sample an image: • 0 – None (retain image format). • 1 – Down-sample color to 8 bit grayscale. • 2 – Down-sample color or grayscale to black and white.
jpegQuality	Integer	The JPEG quality ratio 0 to 99.
batchPrefix	String	Batch prefix to use when creating batch names.
defaultBatchStatusID	String	Identifier of the batch status to associate with batches created by this job.
defaultPriority	Integer	Default priority assigned to batches ranging from 0 to 10.
defaultDocumentTypeID	String	Default document profile for documents created by this job.

Property	Type	Description
searchResultOption	Integer	Determines how to handle database lookups that return more than one result. • 0 – Use the first record. • 1 – Ignore results (do not populate fields).
scriptID	String	Unique identifier of a script to use for this job.
importFrequency	Integer	A value, specified in seconds, that determines how often a job should be polled for work to process. The following values are possible: • 0 – Inactive • 30 – Every 30 seconds • 60 – Every 1 minute • 300 – Every 5 minutes • 900 – Every 15 minutes • 1800 – Every 30 minutes • 3600 – Every 1 hour • -1 – Daily (Specify Time)
hour	Integer	If the importFrequency is set to Daily, this specifies the hour of the day.
minute	Integer	If the importFrequency is set to Daily, this specifies the minute of the day.
lastCheck	Date	Date or time the job was last checked for processing. This will be updated by the Import Job Scheduler after a job is polled for work to process.
fieldMappings	Map<String, FieldMappingInfo>	A set of values that map Capture fields to import source metadata fields.
importSourceClassName	String	Name of the Java class that provides the implementation of the import source for this job.
batchProcessorClassName	String	Name of the class that will be used to process the batch when it is released. If this value is null, the batch lock will be discarded and the batch will be put in a READY state.
batchProcessorJobID	String	A unique identifier for a batch processor job. If this value is null, either the processor does not support jobs or the batch is going to be put in a READY state.
imageFailureAction	Integer	Specifies the action to be taken if an invalid image is encountered: • 0 – Abort the batch • 1 – Skip the item
locale	Locale	Specifies the locale of the list file source.
defaultDateFormat	String	Specifies the default date format of dates in the list file source.
description	String	Description of this job.

Property	Type	Description
encoding	String	Specifies the file encoding of the list file source.
isJobOnline	Boolean	Indicates whether this job should be processed.
preserveImageFiles	Boolean	If *True*, prevents image files from being altered during import.

ImportProcessorContext

The ImportProcessorContext class contains properties relevant to the job being processed. An instance of this class is created before processing is started and is passed to an import source at various stages throughout processing.

Property	Type	Description
cancel	Boolean	When this boolean value is set to *True*, it will cancel the operation being performed.
cancelDBSearch	Boolean	When this boolean value is set to *True*, it will cancel the database lookup.
dBSearchResults	DBSearchResults	Contains the results from a database lookup.
sourceName	String	Name of the import source that the current Import Job is configured to use.
logger	Logger	An instance of java.util.logging.Logger that can be used to log additional entries.
importCancelAction	Integer	This property specifies the action to be taken if a script sets the cancel property to *True* in the preImportFile event. The value may be set to one of the following constants: • CANCEL_ACTION_SKIP = 0 — Skips importing the file into the batch • CANCEL_ACTION_ABORT = 1 — Aborts the entire batch
importJob	ImportJob	Current Import Job being processed.
batchLock	BatchLockEntity	Contains the batch lock entity for the batch, after a new batch has been created.
importSourceFile	String	Name of the file currently being processed.
documentEntity	DocumentEntity	Document entity associated with the file currently being processed.
documentPageEntity	DocumentPageEntity	Document page entity associated with the file currently being processed.
lastMultiPageTiffNumber	Integer	Contains the current page number of a multi-page TIFF file being processed.
workspaceEntity	CaptureWorkspaceEntity	Workspace entity associated with the current batch.
batchManager	BatchManagerSession	Batch manager object used for batch operations.

Property	Type	Description
isExchangeMail	Boolean	Checks whether the current email import job is using exchange web service APIs.

ListFileSourceContext

The ListFileSourceContext class contains all classes used in the processing of a list file source.

Property	Type	Description
folderName	String	Name of the folder currently being processed.
listFilename	String	Name of the list file currently being processed.
listFileLine	String	Contents of the line currently being processed in the list file.
documentFilename	String	Name of the file currently being processed from the current line in the list file.
renamedListFilename	String	If the post-processing step indicates the list file should have a prefix added to it or the extension changed, this property indicates the changed list file name.

Sample Import Processor Scripts

The section describes the following sample Import Processor scripts:

- Sample Import Processor Script 1
- Sample Import Processor Script 2

Sample Import Processor Script 1

The following sample script sets each document's title to the name of the file being imported. When the documents are later committed, their document title can be mapped to an output field.

```
importClass(java.io.File);

function preCreateDocument(event) { // ImportProcessorContext
    var document;   // DocumentEntity
    var sourceFile; // File

    sourceFile = new File(event.getImportSourceFile());
    document = event.getDocumentEntity();

    // Set the document title to be the name of the source file
    document.setDocumentTitle(sourceFile.getName());
}
```

Sample Import Processor Script 2

The following sample script demonstrates using the preCreateDocument event to obtain the base file name of the file being imported and assign that name to a metadata field. In addition, this script shows how to look up the definition of a metadata field by name, locate and create an IndexValue, and set the value of an IndexValue.

```
function preCreateDocument(ctx) {
    // Get the base name of the file being imported.
    var sourceFile = new java.io.File(ctx.getImportSourceFile());
    var baseFileName = sourceFile.getName();

    // Strip off any file extension.
    var dotPos = baseFileName.lastIndexOf('.');
    if (dotPos > -1)
        baseFileName = baseFileName.substring(0, dotPos);

    // Update the "File Name" metadata field with the base name of the file.
    updateIndex(ctx, "File Name", baseFileName);
}

// Update a metadata field
function updateIndex(ctx, indexName, commitValue) {
    var doc = ctx.getDocumentEntity();
    var workspace = ctx.getWorkspaceEntity();

    // Locate the index definition object by the index name.
    var indexDef = findIndexDefinitionByName(workspace, indexName);
    if (indexDef != null) {
        // Get the ID for the index field.
        var indexID = indexDef.getIndexFieldID();
        // Get the index value object for the given document and index ID.
        var indexValue = getIndexValue(doc, indexID);
        // Set the commit value for the index field.
        indexValue.setFieldValue(commitValue);
    }
}

// Search the workspace to find the index definition by name
function findIndexDefinitionByName(workspace, indexName) {
    var indexDefs = workspace.getIndexDefinitions();
    var size = indexDefs.size();
    var foundIndexDef = null;
    for (var i = 0; i < size; i++) {
        var indexDef = indexDefs.get(i);
        if (indexName.equals(indexDef.getFieldName())) {
            // An index by this name was found.
            foundIndexDef = indexDef;
            break;
        }
    }
    return foundIndexDef;
}

// Search the index values of the document for an IndexValue object with the
given ID.
// If one is found, return it; Otherwise, create one and return it.
function getIndexValue(doc, indexDefID) {
```

```
    // Look through all existing document indexes to see if our index is present.
    var indexes = doc.getIndexes();
    var size = indexes.size();
    var foundIndexValue = null;
    for (var i = 0; i < size; i++) {
        var indexValue = indexes.get(i);
        if (indexDefID.equals(indexValue.getFieldID())) {
            // An index by this ID was found.
            foundIndexValue = indexValue;
            break;
        }
    }

    if (foundIndexValue == null) {
        // The index value wasn't found, so create one with blank values.
        foundIndexValue = new Packages.oracle.odc.data.IndexValue(indexDefID,
"", "");
        // Add it to the document's index collection.
        indexes.add(foundIndexValue);
    }

    // Return the IndexValue object.
    return foundIndexValue;
}
```

6

Creating Document Conversion Processor Scripts

Similar to other batch processors, the Document Conversion Processor allows customization of document conversion jobs using JavaScript (Nashorn). Here you learn how to create Document Conversion Processor scripts.

This chapter covers the following topics:

DocumentConverterContext Class

The DocumentConverterContext class contains properties relevant to the job being processed. An instance of this class is created before processing starts, and this instance is passed on to document conversion events at various stages of processing.

The following table lists the DocumentConverterContext fields. When a document conversion event is invoked, the corresponding DocumentConverterContext field is exposed to the event.

Property	Type	Description
cancel	Boolean	When the boolean value is set to true, it will cancel the operation being performed.
logger	Logger	An instance of `java.util.logging.Logger` that can be used to log additional entries.
docConverterJob	DocConverterJob	The current document conversion job being processed.
ble	BatchLockEntity	After a new batch is created, this property contains the Batch Lock entity for the batch.
activeDocument	DocumentEntity	The current active document being processed.
activeAttachment	DocumentEntity	The current active attachment being processed.
activePage	DocumentPageEntity	The current active page being processed.
externalProgramPath	String	The fully-qualified path to an external application that will be used to convert documents.
externalProgramCommandLine	String	The fully-qualified path to an external application that will be used to convert documents.

Property	Type	Description
externalProgramSuccessCode	int	The value returned from an external program that indicates a successful conversion.
externalProgramMonitoringMethod	int	The monitoring method used to monitor the external program. The valid values are: 0 - Process duration 1 - Output file
externalProgramTimeout	int	When `externalProgramMonitoriingMethod` is 1 (Process duration), this value contains the number of minutes to allow the program to run before the program is considered to have timed-out due to a fault or hang. When this value is reached, the external program is terminated and an exception is thrown.
externalProgramDestFile	File	A File object representing the destination file that the external program will generate. Create and pass a File object in the script.

Document Conversion Processor Events

Document Conversion Processor scripts are JavaScript modules that enable you to customize the behavior of certain Document Conversion Processor events.

This section describes the following Document Conversion Processor events:

- Initialize
- preProcessBatch
- postProcessBatch
- preProcessDocument
- postProcessDocument
- preProcessAttachment
- postProcessAttachment
- preProcessPage
- postProcessPage
- preInvokeExternalProcess
- postInvokeExternalProcess

Initialize

The Initialize event signals that the document conversion processor job is in the initialization phase. The `initialize` method is invoked when the job starts. The `DocumentConversionContext` instance for the entire job is initialized at this point and

passed into the method. An implementer can use this method to perform initialization tasks, such as creating database connections or creating temporary paths.

Syntax	Parameter
`public initialize(DocumentConverterContext ctx);`	DocumentConverterContext ctx

preProcessBatch

The preProcessBatch event occurs before a new batch is processed.

Syntax	Parameter
`public preProcessBatch(DocumentConverterContext ctx);`	DocumentConverterContext ctx

Associated DocumentConversionContext Properties

- cancel: If set to *true*, the batch is not processed.
- ble: At this point, this property will be initialized.

postProcessBatch

The postProcessBatch event occurs after the document conversion process is complete. An implementer can close database connections as well as cleanup temporary files and directories.

Syntax	Parameter
`public postProcessBatch(DocumentConverterContext ctx);`	DocumentConverterContext ctx

preProcessDocument

The preProcessDocument event occurs when the document, which is a part of the batch, is active for the conversion job. If there are multiple documents, this event is signaled for each document.

Syntax	Parameter
`public preProcessDocument(DocumentConverterContext ctx);`	DocumentConverterContext ctx

Associated DocumentConversionContext Properties

- cancel: If set to *true*, the document is not processed.
- ble: A reference to the associated BatchLockEntity.
- activeDocument: A reference to the document that is about to be processed.

postProcessDocument

The postProcessDocument event occurs after the document, which is a part of the batch, has completed the conversion job. If there are multiple documents, this event is signaled for each document.

Syntax	Parameter
`public postProcessDocument(DocumentConverterContext ctx);`	DocumentConverterContext ctx

preProcessAttachment

The preProcessAttachment event occurs when the attachment, which is a part of the batch, is active for the conversion job. If there are multiple attachments, this event is signaled for each attachment.

Syntax	Parameter
`public preProcessAttachment(DocumentConverterContext ctx);`	DocumentConverterContext ctx

Associated DocumentConversionContext Properties

- cancel: If set to *true*, the attachment is not processed.
- ble: A reference to the associated BatchLockEntity.
- activeAttachment: A reference to the attachment that is about to be processed.

postProcessAttachment

The postProcessAttachment event occurs after the attachment, which is a part of the batch, has completed the conversion job. If there are multiple attachments, this event is signaled for each attachment.

Syntax	Parameter
`public postProcessAttachment(DocumentConverterContext ctx);`	DocumentConverterContext ctx

preProcessPage

The preProcessPage event occurs when the page, which is a part of the batch or document, is about to be sent to a conversion job. If there are multiple pages, this event is signaled for each page.

Syntax	Parameter
`public preProcessPage(DocumentConverterContext ctx);`	DocumentConverterContext ctx

Associated DocumentConversionContext Properties

- cancel: If set to *true*, the page is not processed.
- ble: A reference to the associated BatchLockEntity.
- activeAttachment: A reference to the page that is about to be processed.

postProcessPage

The postProcessPage event occurs after the page, which is a part of the batch or document, has completed the conversion job. If there are multiple pages, this event is signaled for each page.

Syntax	Parameter
`public postProcessPage(DocumentConverterContext ctx);`	DocumentConverterContext ctx

preInvokeExternalProcess

The preInvokeExternalProcess method is invoked right before an external conversion program is invoked.

Syntax	Parameter
`public preInvokeExternalProcess(DocumentConverterContext ctx);`	DocumentConverterContext ctx

Associated DocumentConversionContext Properties

- cancel: If set to *true*, the page is not processed.
- ble: A reference to the associated BatchLockEntity.
- externalProgramPath: The path to the external program to execute.
- externalProgramCommandLine: The command line to pass to the external program.
- externalProgramSuccessCode: The integer value that represents a successful run when the external program terminates.
- externalProgramMonitoringMethod: The method used to monitor the external program for hangs.
- externalProgramTimeout: The duration in minutes that the external program is allowed to run before the program is terminated.

- externalProgramDestFile: The output file to be generated by the external program; create and pass this File object in the script.

postInvokeExternalProcess

The postInvokeExternalProcess method is invoked soon after an external conversion program is completed.

Syntax	Parameter
`public postInvokeExternalProcess(DocumentConverterContext ctx);`	DocumentConverterContext ctx

Sample Document Conversion Processor Scripts

The section contains the following sample Document Conversion Processor scripts:

- Sample Document Conversion Processor Script 1
- Sample Document Conversion Processor Script 2
- Sample Document Conversion Processor Script 3

Sample Document Conversion Processor Script 1

The following script prints all events when a batch goes through conversion:

```
//doc conversion job script to print at the specific events
function initialize(event) {
    java.lang.System.out.println("initialize");
}
function preProcessBatch(event){
     java.lang.System.out.println("In preProcessBatch");
     java.lang.System.out.println("Batch name preProcessBatch:
"+event.getBle().getBatch().getBatchName());
}
function postProcessBatch(event){
     java.lang.System.out.println("In postProcessBatch");
     java.lang.System.out.println("Batch name postProcessBatch:
"+event.getBle().getBatch().getBatchName());
}
function preProcessDocument(event){
     java.lang.System.out.println("In preProcessDocument");
     java.lang.System.out.println("Title name preProcessDocument:
"+event.getActiveDocument().getDocumentTitle());
}
function postProcessDocument(event){
     java.lang.System.out.println("In postProcessDocument");
}
function preProcessAttachment(event){
     java.lang.System.out.println("In preProcessAttachment");
     java.lang.System.out.println("Attachment name:
"+event.getActiveAttachment().getDocumentTitle());
}
function postProcessAttachment(event){
     java.lang.System.out.println("In postProcessAttachment");
}
```

```
function preProcessPage(event){
     java.lang.System.out.println("In preProcessPage");
     java.lang.System.out.println("Page name:
"+event.getActivePage().getDocumentEntity().getDocumentTitle());
}
function postProcessPage(event){
     java.lang.System.out.println("In postProcessPage");
}
function preInvokeExternalProcess(event){
     java.lang.System.out.println("In preInvokeExternalProcess");
}
function postInvokeExternalProcess(event){
     java.lang.System.out.println("In postInvokeExternalProcess");
}
```

Sample Document Conversion Processor Script 2

The following script cancels operation at the preProcessBatch event:

```
function initialize(event) {
    java.lang.System.out.println("initialize");
}
function preProcessBatch(event){
    var isCancel = true;
    event.setCancel(isCancel);
    java.lang.System.out.println("preProcessBatch and about to cancel
operation.");
    java.lang.System.out.println("Batch name preProcessBatch:
"+event.getBle().getBatch().getBatchName());
}
function postProcessBatch(event){
    java.lang.System.out.println("postProcessBatch - this line will not be
printed.");
    java.lang.System.out.println("Batch name postProcessBatch:
"+event.getBle().getBatch().getBatchName());
}
```

Sample Document Conversion Processor Script 3

The following script changes the batch name in the preProcessDocument event:

```
function preProcessDocument(event){
     java.lang.System.out.println("preProcessDocument
batch-"+event.getBle().getBatch().getBatchName());
     event.getBle().getBatch().setBatchName("BatchInPreDocumentConv");
}

function postProcessDocument(event){
    java.lang.System.out.println("postProcessDocument
batch-"+event.getBle().getBatch().getBatchName());
}
```

7

Creating Commit Processor Scripts

This chapter describes creating Commit Processor scripts. The Commit Processor scripting allows customization of commit processor jobs using user-defined JavaScript.

This chapter covers the following topics:

CommitEventObject Class

The CommitEventObject class contains the properties relevant to the batches or documents being processed. An instance of this class is created before the processing begins and is passed to commit drivers at various stages throughout the processing.

The following table lists the CommitEventObject fields:

Property	Type	Description
cancel	java.lang.Boolean	If set to *True*, the processing operation is canceled.
logger	Logger	An instance of java.util.logging.Logger that can be used to log data to log files.
workingDirectory	java.io.File	The current working directory from where the commit driver processes documents.
batch	BatchEntity	Contains the batch information.
document	DocumentEntity	The current active document which is being processed.
attachmentFileNames	DocumentEntity	The current active attachments which is being processed.
exportDriverInformation	ExportDriverInformation	The export driver information for the attachments.
dateCommitted	java.util.Date	The date on which the document has been committed.
commitProfile	CommitProfileEntity	The current commit profile used by the Commit Processor.
documentFileName	java.lang.String	The file name of the document.

Commit Processor Events

Commit Processor scripts are JavaScript modules that enable you to customize the behavior of certain Commit Processor events.

This section describes the following Commit Processor events:

- preCommit
- preReleaseDocument
- postReleaseDocument
- postCommit

preCommit

The preCommit event occurs prior to a document being committed to a repository. The user-defined method can use this functionality to take action for the commit profile being processed by the commit driver. Setting the cancel attribute to true allows all the documents to skip this commit profile and move onto next active commit profile.

Syntax	Parameter
`public preCommit (CommitEventObject commitEventObject)`	CommitEventObject commitEventObject

Associated CommitEventObject Properties

- cancel: If set to *true*, this commit profile will be skipped and the system will try to commit the documents with the next active commit profile defined in the capture console.
- batch: At this point, this property has been initialized.
- commitProfile: The commit profile has been set.

preReleaseDocument

The preReleaseDocument event occurs prior to a document being released. It allows the user-defined script to take action for the document being released. The cancel property also allows to cancel further processing of the document by canceling and moving onto the next document.

Syntax	Parameter
`public preReleaseDocument (CommitEventObject commitEventObject)`	CommitEventObject commitEventObject

Associated CommitEventObject Properties

- cancel: If set to *true*, this document will not be processed further.
- batch: This property has been initialized.
- commitProfile: The commit profile has been set.
- document: This property has been set to the current document being committed.
- documentFileName: This property is also initialized.

postReleaseDocument

The postReleaseDocument event occurs after each document has been released or committed to the repository. It will allow the user-defined script to take action for

the document after release. Setting the cancel property to true at this point will not have any effect. This method might not be triggered if there is some problem while committing the document. The Commit Profile "Error Handling Policy" will override the behavior.

For example: **Cancel to next commit profile** defined in Commit Profile's "Error Handling Policy" will skip to the next active commit profile in case of an error and then this method will not be called.

Syntax	Parameter
`public postReleaseDocument (CommitEventObject commitEventObject)`	CommitEventObject commitEventObject

Associated CommitEventObject Properties

- cancel: If set to *true*, it will not have any effect.
- batch: This property has been initialized.
- commitProfile: The commit profile has been set.
- document: This property has been set to the current document being committed.
- documentFileName: This property is also initialized at this point.

postCommit

The postCommit event occurs after a batch has been processed for a given commit profile. It will allow the user-defined script to later on take some cleanup or logging action. This method execution does not mean that the documents have been committed successfully to the repository. Setting the cancel property to true at this point will not have an effect.

Syntax	Parameter
`public postCommit (CommitEventObject commitEventObject)`	CommitEventObject commitEventObject

Associated CommitEventObject Properties

- cancel: If set to *true*, it will have no effect.
- batch: This property has been initialized.
- commitProfile: The commit profile has been set.

Sample Commit Processor Scripts

The section contains the following sample Commit Processor scripts:

- Sample Commit Processor Script 1
- Sample Commit Processor Script 2
- Sample Commit Processor Script 3

Sample Commit Processor Script 1

The following script prints all events when a batch goes through commit:

```
//commit processor javascript to print at the specific events
function preCommit(event){
     java.lang.System.out.println("In preCommit");
     java.lang.System.out.println("Batch name preCommit:
"+event.getBatch().getBatchName());
}
function preReleaseDocument(event){
     java.lang.System.out.println("In preReleaseDocument");
     java.lang.System.out.println("Batch name preReleaseDocument:
"+event.getBatch().getBatchName());
}
function postReleaseDocument(event){
     java.lang.System.out.println("In postReleaseDocument");
     java.lang.System.out.println("Batch name postReleaseDocument:
"+event.getBatch().getBatchName());
}
function postCommit(event){
     java.lang.System.out.println("In postCommit");
     java.lang.System.out.println("Batch name postCommit:
"+event.getBatch().getBatchName());
}
```

Sample Commit Processor Script 2

The following script cancels operation at the preCommit event:

```
function preProcessBatch(event){
     event.setCancel(true);
     java.lang.System.out.println("preCommit about to cancel operation commit
profile operation.");
     java.lang.System.out.println("preCommit: Batch name
"+event.getBatch().getBatchName());
}
function preReleaseDocument(event){
     java.lang.System.out.println("postProcessBatch - this method will not be
called.");
     java.lang.System.out.println("postProcessBatch: Batch name
"+event.getBatch().getBatchName());
}
```

Sample Commit Processor Script 3

The following script cancels the document commit at the preReleaseDocument event:

```
function preReleaseDocument(event){
     event.setCancel(true);
     java.lang.System.out.println("preReleaseDocument
batch-"+event.getBatch().getBatchName());
}
```

8
Working with Common Capture Classes

This chapter describes the common Capture classes that pertain to the Recognition Processor and the Import Processor.
This is in addition to the classes that you can use to design the Recognition Processor scripts and the Import Processor scripts. For more information on the Recognition Processor and Import Processor classes, see Recognition Processor Classes and Import Processor Classes.

Common Capture Classes

The following are the classes that pertain to the Recognition Processor and the Import Processor:

- BatchEntity
- BatchItemEntity
- BatchLockEntity
- BatchManagerSession
- BatchStatusEntity
- CaptureWorkspaceEntity
- DBSearchResults
- DBSearchResultRow
- DBSearchFieldInfo
- DocumentEntity
- DocumentPageEntity
- DocumentTypeEntity
- IndexDefinitionEntity
- IndexValue

BatchEntity

The BatchEntity class represents a batch within a Capture Workspace. A batch is a collection of batch items and documents.

Property	Type	Description
itemID	String	The unique batch item identifier.
id	Integer	The unique batch ID.

Property	Type	Description
state	Integer	The current state of the batch, which will be one of the following values: • 1 – READY • 2 – LOCKED • 16 – ERROR • 32 – PROCESSING
errorMessage	String	An error message related to processing failure.
status	BatchStatusEntity	A reference to a BatchStatusEntity that represents the current status of the batch.
priority	Integer	The current priority value of the batch.
itemCount	Integer	The number of items in the batch.
userID	String	The user id of the user that created the batch.
workstationID	String	The host name of the system that created the batch.
comment	byte[]	A comment or note regarding the batch.
dateTime	Date	The date and time the batch was created.
workspace	CaptureWorkspaceEntity	A reference to the workspace to which the batch belongs.
documents	List<DocumentEntity>	A list of DocumentEntity references that exist in the batch.
items	List<BatchItemEntity>	A list of batch items associated with the batch.
lastModifiedDateTime	Date	The date and time the batch was last modified.
lastModifiedUserID	String	The ID of the user that last modified the batch.
batchName	String	The name of the batch.

BatchItemEntity

The BatchItemEntity class represents a batch item within a batch. BatchItemEntities are associated with DocumentPageEntities that are used to form documents within a batch.

Property	Type	Description
itemID	String	The unique batch item identifier.
sourceFileName	String	The original file name of the item. Useful if the item is an imported file.
sourceFormat	String	For non–image files, this is generally the file extension (DOC, XLS, PDF). For image files, the value will be empty.
patchCode	Integer	A patch code value if a patch code was read.

Property	Type	Description
barcodeCount	Integer	The number of barcodes that were read during barcode detection.
linkCount	Integer	The number of documents the item is linked to.
fileLength	Long	The size of the item in bytes.
barcodes	String[]	An array of strings that represent barcode values that were read during barcode recognition.

BatchLockEntity

The BatchLockEntity class represents a lock on a batch. The lock is used to prevent users and processors from accessing the same batch simultaneously.

Property	Type	Description
id	String	The unique batch lock ID.
batch	BatchEntity	A reference to the locked batch.
batchName	String	The name of the batch to which the batch lock is applied.
workspace	CaptureWorkspaceEntity	A reference to the workspace to which the batch belongs.
workspaceName	String	The name of the workspace to which the batch belongs.
lockDate	Date	The date that the batch lock was created.
userID	String	The ID of the user who locked the batch.
computerName	String	The name of the computer used to lock the batch.
processID	String	The process ID used to lock the batch.

BatchManagerSession

The BatchManagerSession class provides methods to audit actions and manipulate batch, document, document page, and batch item objects. This includes create, read, update, delete, move, and so on.

This class includes the following methods:

- auditActivity()
- calculateTotalDocumentPagesInBatch()
- createDocument()
- createDocumentAttachments()
- deleteDocument()
- findDocumentByID()
- findBatchItemByID()

- linkItemToDocument()
- loadBatchItems()
- loadDocumentPages()
- loadDocuments()
- loadDocumentAttachments()
- persistBatch()
- persistBatchItem()
- persistDocument()
- unlinkDocumentPage()
- deleteDocumentPages()
- insertItemsIntoDocument()
- splitDocument()
- mergeDocuments()

auditActivity()

The following table describes the syntax for auditActivity() method:

Syntax	Description
`public void auditActivity(BatchLockEntity ble, Integer actionID, int dataInt, float dataFloat, String dataText1, String dataText2, String dataText3, String dataText4, String dataText5) throws CaptureException`	Writes an audit record to the database for a process defined action.

The following table describes the parameters for auditActivity() method:

Parameter	Type	Description
ble	BatchLockEntity	The batch lock for the batch being acted on.
actionID	Integer	The identifier of the action to log. This is specific to the process.
dataInt	int	A process and action specific integer.
dataFloat	Float	A process and action specific float.
dataText1	String	A process and action specific text value.
dataText2	String	A process and action specific text value.
dataText3	String	A process and action specific text value.
dataText4	String	A process and action specific text value.
dataText5	String	A process and action specific text value.

calculateTotalDocumentPagesInBatch()

The following table describes the syntax for calculateTotalDocumentPagesInBatch() method:

Syntax	Description
`public Integer calculateTotalDocumentPagesInBatch(String batchID) throws CaptureException)`	Calculates the total number of batch items that are linked to documents.

The following table describes the parameters for calculateTotalDocumentPagesInBatch() method:

Parameter	Type	Description
batchID	String	The identifier of the batch.

The following table describes the values that are returned for calculateTotalDocumentPagesInBatch() method:

Value	Type	Description
return	Integer	An integer representing the total number of document pages in the batch.

createDocument()

The following table describes the syntax for createDocument() method:

Syntax	Description
`public String createDocument(String batchLockID, DocumentEntity doc, Integer insertionPoint) throws BatchLockException, CaptureException`	Associates a document object with the specified locked batch.

The following table describes the parameters for createDocument() method:

Parameter	Type	Description
batchLockID	String	The unique identifier of the batch lock.
doc	DocumentEntity	The document entity that is to be associated with the batch.
insertionPoint	Integer	The document number where this doc should be inserted.

The following table describes the values that are returned for createDocument() method:

Value	Type	Description
return	String	The unique document identifier.

createDocumentAttachments()

The following table describes the syntax for createDocumentAttachments() method:

Syntax	Description
`public void createDocumentAttachments(String batchLockID, List<DocumentEntity> documents, List<BatchItemEntity> items, Integer insertionPoint, Boolean reNumberFirstDocument, String parentDocID) throws BatchLockException, CaptureException`	Creates a collection of DocumentEntities as attachments to a specified document and BatchItemEntities to the database. A client might call this method to create a series of batch items and document entities rather than making repeated requests to the server.

The following table describes the parameters for createDocumentAttachments() method:

Parameter	Type	Description
batchLockID	String	The unique identifier of the batch lock.
documents	List<DocumentEntity>	A collection of document objects that require database persistence.
items	List<BatchItemEntity>	A collection of batch item objects that require database persistence.
insertionPoint	Integer	An integer that represents where the document will be inserted.
reNumberFirstDocument	Boolean	A Boolean indicating whether the first document should be renumbered. You should specify *False*, if the first document already exists in the database.
parentDocID	String	An identifier of a parent document to associate the attachments with.

deleteDocument()

The following table describes the syntax for deleteDocument() method:

Syntax	Description
`public void deleteDocument(String batchLockID, String documentID) throws BatchLockException, CaptureException`	Deletes a document in the specified locked batch.

The following table describes the parameters for deleteDocument() method:

Parameter	Type	Description
batchLockID	String	The unique identifier of the batch lock.
documentID	String	The unique identifier of the document to delete.

findDocumentByID()

The following table describes the syntax for findDocumentByID() method:

Syntax	Description
`public DocumentEntity findDocumentByID(String docID) throws CaptureException`	Locates a DocumentEntity by its identifier.

The following table describes the parameters for findDocumentByID() method:

Parameter	Type	Description
docID	String	The unique identifier of the document to locate.

The following table describes the values that are returned for findDocumentByID() method:

Value	Type	Description
return	DocumentEntity	If a matching document is found, the associated entity of the document will be returned. If a match could not be found, then null is returned.

findBatchItemByID()

The following table describes the syntax for findBatchItemByID() method:

Syntax	Description
`public BatchItemEntity findBatchItemByID(String docPageID) throws CaptureException`	Locates the BatchItemEntity for the specified docPageID.

The following table describes the parameters for findBatchItemByID() method:

Parameter	Type	Description
docPageID	String	The unique identifier of the batch item to locate.

The following table describes the values that are returned for findBatchItemByID() method:

Value	Type	Description
return	BatchItemEntity	If a matching batch item is found, the associated entity of the batch item will be returned. If a match could not be found, then null is returned.

linkItemToDocument()

The following table describes the syntax for linkItemToDocument() method:

Syntax	Description
`public DocumentPageEntity linkItemToDocument(String batchLockID, String imageID, String documentID, Integer documentPage) throws CaptureException, BatchLockException`	Associates a batch item with a document.

The following table describes the parameters for linkItemToDocument() method:

Parameter	Type	Description
batchLockID	String	The identifier of the batch lock.
imageID	String	The identifier of the batch item.
documentID	String	The document identifier.
documentPage	Integer	The page number for this item.

The following table describes the values that are returned for linkItemToDocument() method:

Value	Type	Description
return	DocumentPageEntity	A DocumentPageEntity reference containing the linked page.

loadBatchItems()

The following table describes the syntax for loadBatchItems() method:

Syntax	Description
`public List<BatchItemEntity> loadBatchItems(String batchID) throws CaptureException`	Returns a list of batch items belonging to the specified batch.

The following table describes the parameters for loadBatchItems() method:

Parameter	Type	Description
batchID	String	The unique identifier of the batch.

The following table describes the values that are returned for loadBatchItems() method:

Value	Type	Description
return	List<BatchItemEntity>	A collection of items belonging to the batch.

loadDocumentPages()

The following table describes the syntax for loadDocumentPages() method:

Syntax	Description
`public List<DocumentPageEntity> loadDocumentPages(String documentID) throws CaptureException`	Returns a collection of document objects associated with a batch.

The following table describes the parameters for loadDocumentPages() method:

Parameter	Type	Description
documentID	String	The document identifier.

The following table describes the values that are returned for loadDocumentPages() method:

Value	Type	Description
return	List<DocumentPageEntity>	A collection of document pages associated with the document.

loadDocuments()

The following table describes the syntax for loadDocuments() method:

Syntax	Description
`public List<DocumentEntity> loadDocuments(String batchID) throws CaptureException`	Returns a collection of document objects associated with a batch.

The following table describes the parameters for loadDocuments() method:

Parameter	Type	Description
batchID	String	The unique identifier of the batch.

The following table describes the values that are returned for loadDocuments() method:

Value	Type	Description
return	List<DocumentEntity>	A collection of document pages associated with the document.

loadDocumentAttachments()

The following table describes the syntax for loadDocumentAttachments() method:

Syntax	Description
`public List<DocumentEntity> loadDocumentAttachments(String docID) throws CaptureException`	Returns a collection of document objects attached to a specified document.

The following table describes the parameters for loadDocumentAttachments() method:

Parameter	Type	Description
docID	String	The unique identifier of the source document.

The following table describes the values that are returned for loadDocumentAttachments() method:

Value	Type	Description
return	List<DocumentEntity>	A collection of documents attached to the source document.

persistBatch()

The following table describes the syntax for persistBatch() method:

Syntax	Description
`public void persistBatch(String batchLockID, BatchEntity be) throws BatchLockException, CaptureException`	Persists the specified batch to the database.

The following table describes the parameters for persistBatch() method:

Parameter	Type	Description
batchLockID	String	The unique identifier of the batch lock.
be	BatchEntity	The batch entity that is to be updated.

persistBatchItem()

The following table describes the syntax for persistBatchItem() method:

Syntax	Description
`public BatchItemEntity persistBatchItem(String batchLockID, BatchItemEntity entity) throws BatchLockException, CaptureException`	Persists an item to the specified locked batch.

The following table describes the parameters for persistBatchItem() method:

Parameter	Type	Description
batchLockID	String	The identifier of the batch lock.
entity	BatchItemEntity	A BatchItemEntity to persist.

The following table describes the values that are returned for persistBatchItem() method:

Value	Type	Description
return	BatchItemEntity	A reference to the BatchItemEntity object that was persisted.

persistDocument()

The following table describes the syntax for persistDocument() method:

Syntax	Description
`public void persistDocument(String batchLockID, DocumentEntity document) throws CaptureException`	Persists a specific document entity within a batch.

The following table describes the parameters for persistDocument() method:

Parameter	Type	Description
batchLockID	String	The identifier of the batch lock.
document	DocumentEntity	A DocumentEntity to persist.

unlinkDocumentPage()

The following table describes the syntax for unlinkDocumentPage() method:

Syntax	Description
`public void unlinkDocumentPage(String batchLockID, String documentPageID) throws BatchLockException, CaptureException`	Unlinks a document page in the specified pages collection from their associated document.

The following table describes the parameters for unlinkDocumentPage() method:

Parameter	Type	Description
batchLockID	String	The unique identifier of the batch lock.
documentPageID	String	The identifier of the page to unlink.

deleteDocumentPages()

The following table describes the syntax for deleteDocumentPages() method:

Syntax	Description
`public void deleteDocumentPages(String batchLockID, List<DocumentPageEntity> pages) throws BatchLockException, CaptureException`	Unlinks the specified document pages in the pages collection from their associated document.

The following table describes the parameters for deleteDocumentPages() method:

Parameter	Type	Description
batchLockID	String	The unique identifier of the batch lock.
pages	List<DocumentPageEntity>	A collection of pages that require unlinking from their respective documents.

insertItemsIntoDocument()

The following table describes the syntax for insertItemsIntoDocument() method:

Syntax	Description
`public void insertItemsIntoDocument(String batchLockID, String docID, Integer insertionPoint, List<BatchItemEntity> items) throws BatchLockException, CaptureException`	Persists batch item entities in the items collection to the database and creates links to the items in the specified document.

The following table describes the parameters for insertItemsIntoDocument() method:

Parameter	Type	Description
batchLockID	String	The unique identifier of the batch lock.
docID	String	The identifier of the document to which items will be inserted.
insertionPoint	Integer	The point at which the items must be inserted.
items	List<BatchItemEntity>	A collection of items that need persistence.

splitDocument()

The following table describes the syntax for splitDocument() method:

Syntax	Description
`public DocumentEntity splitDocument(String batchLockID, String sourceDocID, Integer docBreakPoint, String newDocTitle) throws BatchLockException, CaptureException`	Breaks a source document into a new document at the specified break point. A new document is created in the database and all pages from the source document starting at the specified breakpoint will be moved into the new document and removed from the source.

The following table describes the parameters for splitDocument() method:

Parameter	Type	Description
batchLockID	String	A valid batch lock identifier for the batch being manipulated.
sourceDocID	String	The source document containing the pages to branch.

Parameter	Type	Description
docBreakPoint	Integer	The starting page number of pages to move.
newDocTitle	String	The title of the new document to be created.

The following table describes the values that are returned for splitDocument() method:

Value	Type	Description
return	DocumentEntity	A new document entity that contains the pages from the source branch.

mergeDocuments()

The following table describes the syntax for mergeDocuments() method:

Syntax	Description
`public void mergeDocuments(String batchLockID, MetadataMergeOption mergeOption,DocAttachmentIncludeOption attachmentIncludeOption, DocumentEntity sourceDocument, List<DocumentEntity> destinationDocuments, boolean addToBeginning) throws CaptureException, BatchLockException`	Merges two documents into one.

The following table describes the parameters for mergeDocuments() method:

Parameter	Type	Description
batchLockID	String	The identifier of the batch lock.
mergeOption	MetadataMergeOption	Indicates how to handle document indexes during merge. Following are the possible values: • discardSourceValues • applySourceValuesOverwrite • applySourceValuesDoNOTOverwrite
attachmentIncludeOption	DocAttachmentIncludeOption	Indicates how to handle document attachments during merge. Following are the possible values: • doNotInclude • includeAll • includeOnlyMatches (Include only those attachments with matching attachment types.)
sourceDocument	DocumentEntity	Source document entity.
destinationDocuments	List<DocumentEntity>	The destination documents collection.
addToBeginning	Boolean	Indicates whether the source document is to be merged to the beginning of the destination documents collection or not.

BatchStatusEntity

The BatchStatusEntity class defines a batch status within a Capture Workspace. Batch statuses may be associated with batches within a Capture Workspace.

Property	Type	Description
statusID	String	The unique identifier of the status.
value	String	The text value of the status.
workspaceEntity	CaptureWorkspaceEntity	A reference to the workspace where the status is defined.

CaptureWorkspaceEntity

The CaptureWorkspaceEntity class represents a workspace in the Capture system. A workspace defines metadata, document profiles, and batch statuses.

Property	Type	Description
workspaceID	String	The unique workspace identifier.
workspaceName	String	The name of the workspace.
description	String	A description of the workspace.
dateCreated	Date	The date the workspace was created.
dateLastModified	Date	The date the workspace was last modified.
createdBy	String	The user ID of the user that created the workspace.
lastModifiedBy	String	The user ID of the user that last modified the workspace.
indexDefinitions	List<IndexDefinitionEntity>	A list of index definition entities that have been defined in the workspace.
statuses	List<BatchStatusEntity>	A list of batch statuses defined in the workspace.
documentTypes	List<DocumentTypeEntity>	A list of document profiles that have been defined in the workspace.

DBSearchResults

The DBSearchResults class contains information returned from executing a database lookup. It contains a list of the rows returned as well as a list of the search field information describing the columns of the rows.

Property	Type	Description
resultsList	List<DbSearchResultRow>	A list of rows from the database lookup.
fieldInfoList	List<DbSearchFieldInfo>	A list of search field information describing the columns used in the database lookup.

DBSearchResultRow

The DbSearchResultRow class represents one row result returned from a database lookup.

Property	Type	Description
results	List<String>	A list of string values associated with one search result. The values in the list will be in the same order in which the return fields are defined.

DBSearchFieldInfo

The DbSearchFieldInfo class represents the field information describing the results of a database lookup.

Property	Type	Description
captureIndexDefID	String	The metadata field ID.
dbColumnName	String	The name of the database column.
dbColumnType	Integer	The type of the database column.
captureFieldType	Integer	The data type of the metadata field.

DocumentEntity

The DocumentEntity class represents a document within a batch. A document consists of a collection of DocumentPageEntity references which refer to BatchItemEntity references.

Property	Type	Description
documentID	String	A value that uniquely identifies the document.
documentTitle	String	The document title.
documentNumber	Integer	The document's position within the batch.
batchEntity	BatchEntity	A reference to the batch to which the document belongs.
documentPages	List<DocumentPageEntity>	A list of document page entity references that make up the document.
documentType	DocumentTypeEntity	A reference to the documentType associated with the document.
indexes	List<IndexValue>	A list of index values for the document.
documentState	Integer	The current state of the document which will be one of the following values: • 1 (READY) – Document is ready to be committed. • 2 (ON HOLD) – Document will not be committed by the Commit Processor.

Property	Type	Description
lastModifiedDateTime	Date	The date and time the document was last modified.
lastModifiedUserID	String	The ID of the user that last modified the document.

DocumentPageEntity

The DocumentPageEntity class represents a page within a document. It refers to a BatchItemEntity within a batch and contains a page number that represents the page's position within the parent document.

Property	Type	Description
docPageID	String	A value that uniquely identifies the document page.
batchItemEntity	BatchItemEntity	A reference to the BatchItemEntity associated with the document page.
documentEntity	DocumentEntity	A reference to the DocumentEntity to which the page belongs.
pageNumber	int	The position of the page within a document.

DocumentTypeEntity

The DocumentTypeEntity class defines a document profile within a Capture Workspace. A DocumentTypeEntity consists of a name, description, and list of index definition fields that pertain to the document profile.

Property	Type	Description
docTypeID	String	The unique identifier of the document profile.
docTypeName	String	The name of the document profile.
description	String	The description of the document type.
workspaceEntity	CaptureWorkspaceEntity	The parent workspace to which the document profile belongs.
fields	List<IndexDefinitionEntity>	A list of IndexFieldDefinitionEntity object references that are associated with the document profile.

IndexDefinitionEntity

The IndexDefinitionEntity represents an index definition defined in a workspace. An index definition defines a metadata field that can be used for input.

Property	Type	Description
indexFieldID	String	The unique identifier of the index definition.

Property	Type	Description
fieldName	String	The name of the field.
workspaceEntity	CaptureWorkspaceEntity	A reference to the parent workspace where the field is defined.
dataType	Integer	The data type of the field. • 0 – NUMERIC • 1 – ALPHA NUMERIC • 3 – DATE • 4 – FLOAT
maxLength	Integer	The maximum number of characters the field will hold.
minValue	Float	The minimum value allowed.
maxValue	Float	The maximum value allowed.
required	Boolean	Set to *True*, if the field is required to have a value at commit time; set to *False* if it is not. The default value is *False*.
defaultValue	String	A default value for the field.
inputMask	String	The input mask defined for the field.
displayFormat	String	The field's display format.
locked	Boolean	Indicates if the field is locked for input.
autoPopulate	Integer	Includes the following: • 0 – None • 1 – Default Value • 2 – Scan Date • 3 – Current Date • 4 – Batch Name • 5 – User ID • 6 – Host Name • 7 – Profile Name • 8 – Batch Status • 9 – Batch Priority • 10 – Document Type
validationExpression	String	The regular expression string used to validate the field value.

IndexValue

The IndexValue class represents the value of a metadata field in a document. It contains a display value that is presented to the user as well as a fieldValue which will be used at commit time.

Property	Type	Description
fieldID	String	The unique identifier of the IndexDefinitionEntity that is associated with the index value.
fieldValue	String	The value of the field that will be used when the document is committed.

Property	Type	Description
displayValue	String	A value that is presented to the user. This value will not be used at commit time.

A
Keycodes

If you need to specify a keycode in a JavaScript, refer to the following location:

http://docs.oracle.com/javase/7/docs/api/java/awt/event/KeyEvent.html