Oracle® Fusion Middleware

Developing Scripts for Oracle WebCenter Capture

11*g* Release 1 (11.1.1)

E28275-01

July 2013

Documentation for Oracle WebCenter Capture developers that describes how to develop scripts for customizing Capture functionality.

Oracle Fusion Middleware Developing Scripts for Oracle WebCenter Capture, 11*g* Release 1 (11.1.1)

E28275-01

Primary Author: Mike Manier

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

Contents

Preface

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

Audience

This guide is intended for developers responsible for customizing Oracle WebCenter 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 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

For more information, see the following documents in the Oracle WebCenter Capture 11*g* Release 1 (11.1.1) documentation set:

- *Oracle Fusion Middleware Administering Oracle WebCenter Capture*
- *Oracle Fusion Middleware Using Oracle WebCenter Capture*
- *Oracle Fusion Middleware Managing Oracle WebCenter Capture*

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.

Convention	Meaning
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1 Introduction to Oracle WebCenter Capture Scripts

This chapter provides an introduction to developing scripts for Oracle WebCenter Capture.

A script is a custom piece of code consumed by the Capture client or 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.

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

1. For the component, write the JavaScript, using the events, classes, and information provided in the component's chapter:
   - Chapter 4, "Creating Client Scripts"
   - Chapter 5, "Creating Recognition Processor Scripts"
   - Chapter 6, "Creating Import Processor Scripts"

   Refer to Chapter 2, "Capture Scripting Samples" for reference.
2. On the Advanced tab of a selected workspace in the WebCenter Capture Workspace Console, a workspace manager adds the script, identifying its component type and loading the script file.
3. In a client profile or an import or recognition processor job, the workspace manager 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 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, as referenced in the guide Oracle Fusion Middleware Managing Oracle WebCenter Capture:

- 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, as described in the guide *Oracle Fusion Middleware Managing Oracle WebCenter Capture*.
- Import Processor

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

- Recognition Processor

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

2

Capture Scripting Samples

This chapter includes JavaScript samples that provide examples of scripts you might create for client profiles, import jobs, or recognition jobs.

This chapter contains the following sections:

- Section 2.1, "Sample Client Scripts"
- Section 2.2, "Sample Import Processor Script"
- Section 2.3, "Sample Recognition Processor Script"

2.1 Sample Client Scripts

This section includes the following sample scripts:

Section 2.1.1, "Sample Client Script 1"

Section 2.1.2, "Sample Client Script 2"

2.1.1 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.

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

```
importClass(java.awt.event.KeyEvent);

function ScriptStart() {
    println("ScriptStart");
}

function BatchScanBegin(event) { // BatchScanEvent
    println("BatchScanBegin");
}

function BatchScanComplete(event) { // BatchScanEvent
    println("BatchScanComplete");
    println(event.getBatch().getBatchName() + " finished Scanning.");
}
```

```
function BatchSelected(event) { // BatchSelectedEvent
    println("BatchSelected: " + event.getBatch().getBatchName());
}

function PreBatchDelete(event) { // BatchDeleteEvent
    println("BatchDelete");
}

function CaptureImage(event) { // ImageCaptureEvent
    println("CaptureImage");
}

function DocumentCreated(event) { // DocumentCreatedEvent
    println("DocumentCreated");
}

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

function FieldGotFocus(event) { // FieldEvent
    println("FieldGotFocus");
}

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

function FieldProcessKey(event) { // FieldEvent
    var keyEvent;
    // println("FieldProcessKey");
    keyEvent = event.getKeyEvent();
    if (keyEvent.getID() == KeyEvent.KEY_TYPED) {
        //println(keyEvent.getKeyChar());
        if (String.fromCharCode(keyEvent.getKeyChar()) == '*') {
            println("Asterisk not allowed in any field.");
            keyEvent.consume();
        }
    }
}

function PostCaptureImage(event) { // ImageCaptureEvent
    println("PostCaptureImage");
}

function PreCaptureImage(event) { // ImageCaptureEvent
    println("PreCaptureImage");
}


function PreUploadItem(event) { // UploadItemEvent
    println("PreUploadItem: " + event.getCaptureItem().getFilename());
}
```

```
function PostUploadItem(event) { // UploadItemEvent
    println("PostUploadItem: " + event.getCaptureItem().getFilename());
}

function DBSearchComplete(searchEvent) { // DBSearchEvent
    println("DBSearchComplete.");
}

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

    println("DBSearchResult");

    results = searchEvent.getRowResults();
    println("Found " + results.size() + " results.");
}

function DBSearchStart(searchEvent) {  // DBSearchEvent
    println("DBSearchStart");
    println("Metadata value was " + searchEvent.getMetadataValue());
    //searchEvent.setMetadataValue("c");
}

function DocumentRemoved(event) { // DocumentRemovedEvent
    println("DocumentRemoved");
}

function ImportFilesSelected(files, cancel) { // ImportFilesSelectedEvent
    println("ImportFilesSelected");
}

function PostDownloadItem(event) { // DownloadItemEvent
    println("PostDownloadItem: " + event.getCaptureItem().getFilename());
}

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

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

2.1.2 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.

```
importClass(java.util.ArrayList);
```

```
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 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);
    }
}
```

2.2 Sample Import Processor Script

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());
}
```

2.3 Sample Recognition Processor Script

This sample script customizes the recognition job's behavior in the following ways:

- Sets the document organization type to a fixed number of pages per batch.
- Sets the maximum pages per document to two.
- Sets the job to detect pdf417 bar codes only.
- Defines three bar codes named processorDate, Title, and Amount (with no validation rules).
- Map the bar codes to three metadata fields.

```
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.
                parsed = barcodeValue.split('\\|');
                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]);
                    }
                }
            }
```

3

Integrating the Client With Other Web Applications

This chapter discusses how to configure the client to launch from another web application.

The web application invokes the client via a web address. Parameters such as the workspace, capture source, client profile, document profile, and optional metadata values are passed within the address.

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 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 web address. Once scanned, the document is displayed in the document pane. Metadata fields are automatically populated with user entries, which were passed in the web address. The user reviews the document and completes other metadata fields, then releases the batch, scans additional batches, or closes the browser. Upon close, the user returns to the line of business web application.

3.1 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 web address and its parameters.

   Parameters are listed and described in Table 3–1.

   See the example integration configuration in Section 3.1.1.
3. If needed, configure Single Sign-on (SSO) to prevent users from having to sign in to Capture before beginning scanning.

Table 3–1 Client Integration URL Parameters

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

Table 3–1 (Cont.) Client Integration URL Parameters

Parameter	Description
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** icon. ■ If you specify `true` or `1`, the user is signed out after a batch is released and the SignOutAction is performed, if specified.
SignOutAction	Specifies the action that occurs after the user is signed out. ■ If you specify `CloseBrowser`, the browser is closed and the user returns to the web application. Note that **SignOutAction** and **CloseBrowser** options function in Internet Explorer. In Firefox, they function if the client integration was launched via JavaScript. ■ If you specify a URL, the window is redirected to the specified URL.
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.
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.

3.1.1 Example Client Integration Web Address

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

```
http://svr-capture:16400/dc-client/faces/dc-client?CaptureWorkspace=Accoun
ting&ClientProfile=Import%20Invoices&CaptureDriver=CAPTURE_IMPORT_
DRIVER&CaptureSource=Import%20Source&SignOutOnRelease=1&SignOutAction=Clos
eBrowser&Company=MyCompany&Dept=Accounting
```

This web address 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 importing CAPTURE_IMPORT_DRIVER as the capture source rather than scanning.
- `CaptureSource=Import Source` - Specifies Import Source as the capture source.
- `SignOutOnRelease=1&SignOutAction=CloseBrowser` - 1&SignOutAction=CloseBrowser specifies that the user is signed out after releasing a batch and the browser closes.
- `Company=MyCompany` - Passes a value of MyCompany for the Company metadata field.
- `Dept=Accounting` - Passes a value of Accounting for the Dept metadata field.

4 Creating Client Scripts

This chapter discusses creating scripts for the client. Scripts provide hooks into client events.

User-written client scripts are executed when certain Capture client events are triggered. The framework defines a factory interface for script creation as well as an extension provider interface for extension configuration and invocation. This implementation of the extension provider allows for customization using JavaScript.

This chapter includes the following sections:

- Section 4.1, "Client Events"
- Section 4.2, "Client Event Classes"
- Section 4.3, "Capture Client Core Classes"
- Section 4.4, "Capture Client FieldEdit Classes"

4.1 Client Events

This section defines client events.

- **BatchScanBegin**

  Occurs when scanning into a batch is about to begin.

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

- **BatchScanComplete**

  Occurs when scanning into a batch is complete.

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

- **BatchSelected**

  Occurs when a batch has been selected.

  ```
  public void BatchSelect(BatchSelectedEvent event);
  ```

- **CaptureImage**

  Occurs when an image is about to be captured from the scan source.

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

- **DBSearchComplete**

  Occurs when the database search has completed, just before the results are to be processed.

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

- **DBSearchResults**

  Occurs as database search results are being processed.

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

- **DBSearchStart**

  Occurs just before a database search.

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

- **DocumentCreated**

  Occurs after a document has been created.

  ```
  public void DocumentCreated(DocumentCreatedEvent event);
  ```

- **DocumentRemoved**

  Occurs after a document has been removed.

  ```
  public void DocumentRemoved(DocumentRemovedEvent event);
  ```

- **DocumentSelected**

  Occurs when a document has been selected.

  ```
  public void DocumentSelect(DocumentSelectedEvent event);
  ```

- **FieldGotFocus**

  Occurs when a metadata field receives the input focus.

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

- **FieldLostFocus**

  Occurs when a field has lost the input focus.

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

- **FieldProcessKey**

  Occurs when a key event happens while the focus is in a metadata field.

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

- **PreBatchDelete**

  Occurs when a batch is about to be deleted.

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

- **PreCaptureImage**

  Occurs before an image has been captured from the scan source.

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

- **PreDownloadItem**

  Occurs when a batch item is about to be downloaded.

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

- **PreUploadItem**

  Occurs when a batch item is about to be uploaded.

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

- **PostCaptureImage**

  Occurs after an image has been captured from the scan source.

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

- **PostDownloadItem**

  Occurs after a batch item has been downloaded.

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

- **PostUploadItem**

  Occurs after a batch item has been uploaded.

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

- **RegionSelected**

  Occurs when a region has been selected on a document page.

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

- **ScriptStart**

  Occurs when scripting is first initialized.

  ```
  public void ScriptStart();
  ```

4.2 Client Event Classes

This section defines client event classes.

Table 4–1 Client Event Classes

Class	Description
oracle.oddc.clientextension.BatchDeleteEvent	**List<CaptureBatch> batches** - The list of batches that will be deleted. **boolean canceled** - If this is set to True, the delete operation will be canceled.
oracle.oddc.clientextension.BatchScanEvent	**CaptureBatch batch** - The batch that new items will be added to during scan/import. **boolean canceled** - If this is set to True, the scan/import will be canceled. **List<File> sourceFiles** - When using the Import Source, this contains the list of files being imported. **CaptureOperation operation** - Indicates the operation that triggered this event.
oracle.oddc.clientextension.BatchSelectedEvent	**CaptureBatch batch** - The batch that has been selected in the batch pane.

Table 4–1 (Cont.) Client Event Classes

Class	Description
oracle.oddc.clientextension.DBSearchEvent	**boolean displayHitlist** - If set to True, displays the database lookup results. **boolean exactMatch** - If set to True, the search value must be matched exactly. **String metadataID** - The ID of the metadata field being searched. **String metadataValue** - The value of the metadata field being searched. **List<DbSearchResultRow> rowResults** - The list of row results returned from the search. **boolean canceled** - If set to True, cancels the search.
oracle.oddc.clientextension.DocumentRemovedEvent	**CaptureDocument document** - The document being removed from the batch.
oracle.oddc.clientextension.DocumentCreatedEvent	**CaptureDocument document** - The document that has just been created.
oracle.oddc.clientextension.DocumentSelectedEvent	**CaptureDocument document** - The document that has been selected in the batch pane.
oracle.oddc.clientextension.DownloadItemEvent	**CaptureItem captureItem** - After a batch is opened, indicates the current item being downloaded from the server.
oracle.fieldedit.FieldEvent	**Traversal constants** static final int TRAVERSAL_UNDETERMINED = 0 static final int TRAVERSAL_FORWARD = 1 static final int TRAVERSAL_BACKWARD = 2 static final int TRAVERSAL_FORWARD_COMPONENT = 3 static final int TRAVERSAL_BACKWARD_COMPONENT = 4 **Boolean cancel** - If set to True, cancels the event. **DataField field** - The field this event is acting upon. **KeyEvent keyEvent** - The keyboard event used to generate this event. **int traversalDirection** - Indicates which direction the field focus is moving (see Traversal constants).

Table 4–1 (Cont.) Client Event Classes

Class	Description
ice.core.ImageCaptureEvent	**boolean cancel** - If this is set to True, the capture operation will be canceled. **Integer imageCount** - Indicates how many images have been captured. **String imageFileName** - Indicates the file name of image saved locally. **Integer xdpi** - For images, indicates the horizontal dots per inch. **Integer ydpi** - For images, indicates the vertical dots per inch. **Integer brightness** - The brightness value used to capture the image. **Integer contrast** - The contrast value used to capture the image. **boolean logicalBreak** - If set to True, indicates the start of a document. **List<File> sourceFiles** - When using the Import Source, contains the list of Files being imported. **String sourceFileName** - When using the Import Source, contains the name of the source file being imported. **BufferedImage image** - For images files, contains a BufferedImage object. **ImageCaptureEngine source** - The ImageCapture Engine that created this event. **ImageCaptureEngine.ImageFormat imageFormat** - Indicates the format that images will be saved as. Current values are: tiffG4, tiffJpegGray, tiffJpegColor, jpegGray, and jpegColor.
oracle.oddc.clientextension.RegionSelectedEvent	**MouseEvent mouseEvent** - The MouseEvent used to select the region within the image. **Rectangle selectionRectangle** - The rectangle selected within the image. **BufferedImage image** - The BufferedImage containing the selected portion of the image.
oracle.oddc.clientextension.UploadItemEvent	**CaptureItem captureItem** - After a batch is released, indicates the current item being uploaded to the server.

4.3 Capture Client Core Classes

This section defines Client Core Classes.

Table 4–2 Client Core Classes

Class	Description
oracle.oddc.client.data.CaptureBatch	**String batchId** - The internal ID for the batch. **String batchName** - The name of the batch. **String batchPath** - The local path where information for this batch is stored. **String createdBy** - The user that created the batch. **Date createdDate** - The date that the batch was created. **Int currentPriority** - The priority assigned to the batch. **CaptureBatchStatus currentStatus** - The status assigned to the batch. **CaptureDocuments documents** - The documents contained in the batch. **CaptureItems items** - The items associated with the batch. **Date lastModifiedDate** - The date that the batch was last modified. **String note** - The note assigned to the batch. **CaptureWorkspace workspace** - The workspace used to create the batch. **Method - void persist() throws BatchLockException, CaptureException** - Saves the batch record to the server.
oracle.oddc.client.data.CaptureBatchStatus	Defines the batch status. **String value** - The description of the status. **String id** - The internal ID of the status.
oracle.oddc.client.data.CaptureDataType	An enumeration that defines the data types for metadata field definitions. Values include: **NUMERIC, ALPHA, ALPHANUMERIC, DATE, FLOAT**
oracle.oddc.client.data.CaptureDocument	This class contains all properties of a document. **DocumentType documentType** - The document profile assigned to the document. **CaptureFields fields** - The metadata assigned to the document. **String id** - The internal ID for the document. **CaptureDocumentPages pages** - The pages contained in the document. **CaptureBatch parentBatch** - The batch that contains this document. **String title** - The title of the document. **Method - void persist() throws BatchLockException, CaptureException** - Saves the document, related document pages, and metadata to the server.

Table 4–2 (Cont.) Client Core Classes

Class	**Description**
oracle.oddc.client.data.CaptureDocumentPage	The class contains the properties of a document page. **String imageFilenameKey** - The local filename for this page. **String pageID** - The internal ID for the page. **int pageNumber** - The number of the page within the document. **CaptureDocument parentDocument** - The document that contains this page.
oracle.oddc.client.data.CaptureDocumentPages	This 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.
oracle.oddc.client.data.CaptureDocuments	This class is a collection of documents and is of type Vector<CaptureDocument>. Use the Vector methods to retrieve documents from instances of this class.
oracle.oddc.client.CaptureErrorManager	This class manages what error messages are logged. **Level logLevel** - The minimum level used to log messages. **Methods** ■ **void logMessage(Level level, String message)** - Logs a message with the specified log level. ■ **void logMessage(Level level, String message, Throwable errorException)** - Logs a message and error with the specified log level.
oracle.oddc.client.data.CaptureField	Contains the properties of a document metadata field. **String displayValue** - The value to display for this field. **String fieldname** - The name of the field. **int length** - The maximum length of the field. **boolean required** - If True, this field is required to have a value. **String value** - The value for this field.
oracle.oddc.client.data.CaptureFields	This 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.
oracle.oddc.client.data.CaptureItem	Contains properties of an item (single image or non-image file) associated with a document page. **String filename** - The name of the file for this item. **CaptureBatch parentBatch** - The batch containing this item. **String sourceFilename** - If this item was imported, contains the name of the file it was imported from.
oracle.oddc.client.data.CaptureItems	This 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.

Table 4–2 (Cont.) Client Core Classes

Class	Description
oracle.oddc.client.CaptureStateManager	The class contains properties related to the current state of the client. The instance of this class is available to all scripting events via the "Capture" property. **CaptureBatch activeBatch** - The active batch. **CaptureDocument activeDocument** - The active document. **DocumentType activeDocumentType** - The active document profile. **CaptureDocumentPage activePage** - The active document page. **ClientProfile activeProfile** - The active client profile. **String applicationUserPath** - The path the client uses for its application data. **String batchesPath** - The path the client uses to cache batch data. **String captureSystemId** - The ID of the Capture system that the client is connected to. **String computerName** - The name of the computer the client is running on. **String currentUser** - The user currently logged into the client. **CaptureErrorManager errorManager** - The Error Manager object used for logging information.
oracle.oddc.client.data.CaptureWorkspace	Contains all properties and operations for a workspace. **String id** - The internal ID associated with the workspace. **FieldDefinitions fieldDefinitions** - The metadata defined for this workspace. **String name** - The name of the workspace. **List<CaptureBatchStatus> statuses** - A list of batch statuses available to this workspace. **Method** - public DBLookupProfile getDBLookupProfile(String profileId) **throws CaptureException** - Retrieves the database lookup profile for the given database lookup profile ID.

Table 4–2 (Cont.) Client Core Classes

Class	Description
oracle.oddc.client.data.ClientProfile	Contains the properties of a client profile as defined in the Capture Workspace Console. **boolean alwaysDisplayHitList** - If True, after a database lookup is executed, the results are displayed regardless of the number of results returned. **boolean applyBrightness** - If True, applies the brightness and contrast settings to the selected Capture source. **int batchFilterDaysOldFrom** - A batch filter setting that specifies the minimum days old a batch can be. **int batchFilterDaysOldTo** - A batch filter setting that specifies the maximum days old a batch can be. **String batchFilterPrimarySortField** - The batch property used for the primary sort of the batches tree. **SortOrder batchFilterPrimarySortOrder** - The primary sort order for the batches tree. **String batchFilterSecondarySortField** - The batch property used for the secondary sort of the batches tree. **SortOrder batchFilterSecondarySortOrder** - The secondary sort order for the batches tree. **Integer batchFilterState** - The batch states to include in the batch filter **String batchPrefix** - The batch prefix to use for batches created using this profile. **String batchProcessorID** - The ID of the batch processor to use during post-processing. **String batchProcessorJobID** - The ID of the batch processor's job to use during post-processing. **BatchVisibility batchVisibility** - A batch filter setting that specifies when a user will see the batch in the batches tree. **long blankByteThreshold** - 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. **int brightness** - The brightness to apply to the selected Capture source. **CaptureType captureType** - Indicates whether the profile is capture-only, capture and index, or index-only. **int contrast** - The contrast to apply to the selected Capture source. **int dbLookupMaxRecords** - The maximum number of records to return from a database lookup. **String dbLookupProfileId** - The ID of the database lookup profile used by this profile. **ColorType defaultColor** - The default color type used during capture. **int defaultDpi** - The default DPI to use during capture. **int defaultPriority** - A batch created from this profile will be assigned this priority. **String defaultStatusId** - A batch created from this profile will be assigned this status . **String description** - The description for the profile.

Table 4–2 (Cont.) Client Core Classes

Class	Description
oracle.oddc.client.data.ClientProfile *(Cont.)*	**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. **String id** - The ID associated with this profile. **int maxPages** - The non-image file preview page limit. **String name** - The name of the profile. **NonImageAction nonImageAction** - The action to take for non-image files. **String picklistRelationshipProfile** - The dependent choice list used by this profile. **List<String> prefixes** - The batch prefixes used in the batch filter. **boolean preventDefaultColorOverride** - If True, the color cannot be overridden. **boolean preventDefaultDpiOverride** - If True, the dpi cannot be overridden. **List<Integer> priorities** - The batch priorities used in the batch filter. **int sepByteThreshold** - 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. **List<String> statuses** - The batch statuses used in the batch filter. **List<String> supportedDocumentTypes** - A list of document profile IDs which represent the document profiles that this profile can assign to a batch. **String workspaceId** - The ID of the workspace in which this profile is associated. **String workspaceName** - The name of the workspace in which this profile is associated. **EnumerationValues** ■ **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

Table 4–2 (Cont.) Client Core Classes

Class	Description
oracle.oddc.client.data.DBLookupProfile	This class represents a profile for database lookup. **Sort constants** ▪ **public static final Integer SORT_ASC = 0;** ▪ **public static final Integer SORT_DESC = 1;** **Methods** ▪ **public DBLookupResult execDBLookup(String searchID, String fieldID, String value, boolean exactMatch) throws CaptureException** - Executes a database lookup without sorting. Refer to the other execDBLookup method for parameter explanations. ▪ **public DBLookupResult execDBLookup(String searchID, String fieldID, String value, Boolean exactMatch, String primarySortField, Integer primarySortOrder, String secondarySortField, Integer secondarySortOrder) throws CaptureException** - This executes a database lookup. ▪ **searchID** - The ID of the database search defined for the database lookup profile. ▪ **fieldID** - The ID of the metadata field being searched. ▪ **value** - The value being searched upon. ▪ **exactMatch** - If True, the value must match exactly. ▪ **primarySortField** - The field ID used for the primary sort. ▪ **primarySortOrder** - The sort order of the primary sort. ▪ **secondarySortField** - The field ID used for the secondary sort. ▪ **secondarySortOrder** - The sort order of the secondary sort.
oracle.oddc.client.data.DBLookupResult	The class represents the result of a database lookup. **List<DbSearchFieldInfo> searchFieldInfoList** - A list of search field information describing the results returned by the database lookup. **List<DbSearchResultRow> searchResultRows** - A list of search result rows returned by the database lookup.
oracle.oddc.session.DbSearchResultRow	This class represents one row result returned from a database lookup. **List<String> results** - 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.
oracle.oddc.session.DbSearchFieldInfo	This class represents the field information describing the results of a database lookup. **String captureIndexDefID** - The metadata field ID. **String dbColumnName** - The name of the database column. **Integer dbColumnType** - The type of the database column. **Integer captureFieldType** - The data type of the metadata field.

Table 4–2 (Cont.) Client Core Classes

Class	Description
oracle.oddc.client.data.DocumentType	This class represents a document profile. A document profile dictates what metadata fields are available to documents created from this type. **FieldDefinitions fieldDefinitions** - The metadata applicable to the document profile. **String id** - The internal ID associated with the document profile. **String name** - The name for the document profile. **String description** - The description for the document profile.
oracle.oddc.client.data.DocumentTypes	This class is a map of document profiles. It is of type TreeMap<String, DocumentType> and the map key is the document profile ID. Use the TreeMap methods to retrieve the document profiles from instances of this class.

Table 4–2 (Cont.) Client Core Classes

Class	**Description**
oracle.oddc.client.data.FieldDefinition	This class represents a metadata field's definition. **ClientProfile.AutoPopulateType autoPopulateType** - Specifies how the field should be auto-populated. **CaptureDataType dataType** - The data type of this field. **String defaultValue** - The default value for this field. **boolean displayable** - Indicates whether this field will be displayed in the client. **int length** - The maximum length for this field. **String id** - The internal ID for this field. **String name** - The name of this field. **String inputMask** - The input mask. **boolean locked** - True if the field is locked. **float maxValue** - The maximum value for this field. **float minValue** - The minimum value for this field. **boolean pickListCaseInsensitive** - If True, the choice list is case insensitive. **String pickListID** - The ID of the choice list associated with this field. **String pickListParentFieldID** - The ID of the choice list parent field. **String pickListSourceID** - The source ID of the dependent choice list for this field. **String profileDisplayFormat** - The format used when displaying the field. **boolean required** - Indicates whether this field is required to have a value. **String validationExpression** - A regular expression used to validate the values entered for this field.
oracle.oddc.client.data.FieldDefinitions	This 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. Use the LinkedHashMap methods to retrieve the metadata field definitions from instances of this class.
oracle.oddc.client.batchbuilder.CaptureOperation	This is an enumeration that defines the capture operation being performed on the batch. Values include: **Create**, **Append**, **Insert**, and **Replace**

4.4 Capture Client FieldEdit Classes

The FieldEdit class is the user interface component for entering metadata values.

Table 4–3 Client FieldEdit Classes

Class	Description
oracle.fieldedit.field.DataField	This class represents the data for a single field of the FieldEdit component and is the base class for the various field types. **String caption** - The field caption. **String fieldname** - The field name. **Format displayFormat** - The display format for the field. **String inputMask** - The input mask **boolean lock** - If True, the field is locked. **long maxLength** - The maximum length of the field. **boolean required** - If True, this field is required to be entered. **String uncommittedText** - The text current in the field.
oracle.fieldedit.field.DateField	This class extends from DataField and represents a date field. **Date value** - The value, represented by a Date object.
oracle.fieldedit.field.FloatField	This class extends from DataField and represents a float field. **Float value** - The value, represented by a Float object.
oracle.fieldedit.field.IntegerField	This class extends from DataField and represents an integer field. **Integer value** - The value, represented by an Integer object.
oracle.fieldedit.field.PicklistField	This class extends from DataField and represents a Pick-list (Choice List) field. **boolean caseSensitive** - If true, the contents of this list are sensitive to case. **List<PicklistEntry> pickListEntries** - The list of entries in the choice list. **PicklistEntry value** - The current value of the choice list.
oracle.fieldedit.field.TextField	This class extends from DataField and represents an alphanumeric field. **String value** - The alphanumeric value of the field.
oracle.fieldedit.field.PicklistEntry	**public String getCommitValue()** - Returns the commit value of the choice list entry. **public String getDisplayValue()** - Returns the display value of the choice list entry.

5 Creating Recognition Processor Scripts

This chapter discusses 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.
- Updating corporate databases using bar code values.
- Using custom logic to determine which pages constitute document separation.
- Performing custom auditing of server activity.
- Sending custom email alerts.
- Canceling the committing of a batch due to invalid data.

For a list of Recognition Processor events, see Section 5.2, "Script Design."

This chapter covers the following topics:

- Section 5.1, "Batch Job Processing Order of Events"
- Section 5.2, "Script Design"
- Section 5.3, "Script Methods"

5.1 Batch Job Processing Order of Events

Events are executed in the following order in recognition processor batch jobs:

1. initialize
2. processBatch
3. restoreCaptureBatch (only if the batch has previously failed during document creation)
4. beginPhase
5. extractBatchItem
6. barcodesFoundonItem
7. endPhase
8. beginPhase
9. batchItemAllValidBarcodes
10. determineSeparatorPage

11. batchItemValidBarcode (This applies to the bar code on every page organization type. It only happens when *Optimize Bar Code Recognition* is turned on and the processor is unable to find a bar code on a batch item.)
12. endPhase
13. beginPhase
14. determineDocType
15. endPhase
16. beginPhase
17. beginDatabaseLookup
18. determineIndexValues
19. endPhase
20. beginPhase
21. renameOrigCaptureDocTitle
22. createCaptureDoc
23. endPhase
24. beginPhase
25. postProcess
26. endPhase
27. endBatchProcess

5.2 Script Design

This section describes classes that can be used in script design, which include the following:

- RecognitionJob
- BarcodeDefinition
- SeparatorDefinition
- SeparatorRuleDefinition
- DocumentDefinition
- RecognitionJobField
- RecognitionProcessorContext
- PostProcessContext
- ProcessorSeparatorPage
- ProcessorItem
- ProcessorDocument

5.2.1 RecognitionJob

The constants for the bar code symbologies are as follows.

Constants	Description

BARCODE_CODEABAR=0	Codeabar
BARCODE_CODE128=1	Code 128
BARCODE_CODE39=2	Code 39
BARCODE_CODE93=3	Code 93
BARCODE_EAN13=4	EAN-13
BARCODE_EAN8=5	EAN-8
BARCODE_INTERLEAVED25=6	Interleaved2/5
BARCODE_UCCEAN128=7	UCC/EAN 128
BARCODE_UPCA=8	UPC-A
BARCODE_UPCE=9	UPC-E
BARCODE_AIRLINE25=10	Airline(IATA) 2/5
BARCODE_CODE32=11	Code32
BARCODE_DATALOGIC25=12	Datalogic 2/5
BARCODE_INDUSTRIAL25=13	Industrial 2/5
BARCODE_ISBNADDON2=14	ISBN Addon 2
BARCODE_ISBNADDON5=15	ISBN Addon 5
BARCODE_MATRIX25=16	Matrix 2/5
BARCODE_POSTNETPLANET=17	Postnet/Planet
BARCODE_PATCHCODE=18	Patch Code
BARCODE_DATAMATRIX=19	Data Matrix
BARCODE_PDF417=20	PDF417
BARCODE_QRCODE=21	QR code

The following are constants for the document organization type.

Constant	Description
ORGANIZE_SINGLEPAGE=0	Organize documents based on a fixed number of pages per document.
ORGANIZE_SINGLEDOC=1	Do not perform document organization.
ORGANIZE_MULTIPAGE_ BARCODE=2	Organize documents based on the same bar code value on each page.
ORGANIZE_MULTIPAGE_SEP=3	Organize documents based on separator pages
ORGANIZE_MULTIPAGE_OTHER=4	Organize documents based on hierarchical separator pages.

The following are values used by database lookup.

Values	Description
DBLOOKUP_NONE=0	No database lookup is configured.
DBLOOKUP_BARCODE=1	Use a bar code value to perform database lookup.

Values	Description
DBLOOKUP_INDEXFIELD=2	Use the index field value to perform database lookup.

The following show actions to take when a database lookup finds multiple records.

Actions	Description
DBMULTIPLERECORD_USEFIRST=0	Use the first record found during database lookup.
DBMULTIPLERECORD_DONOTLINK=1	Do not populate the database lookup result.

The following show what action to take when a database lookup finds no match.

Actions	Description
DBNOMATCH_ALLOWCOMMIT=0	Permit the batch to be committed even when no database record is found.
DBNOMATCH_NOCOMMIT=1	Do not allow the batch to be committed when no match is found.

The following are operators.

Operators	Description
OPERATOR_OR = 0	The OR operator, used in cover page definition rules.
OPERATOR_AND = 1	The AND operator, used in cover page definition rules.

The following values show how the document type is dynamically determined.

Values	Description
DOCTYPE_NONE = 0	The document type is not dynamically determined.
DOCTYPE_BARCODE = 1	The document type is dynamically determined based on a bar code value.
DOCTYPE_SEPARATOR = 2	The document type is dynamically determined based on a separator page.

The following are actions to take when multiple bar code values are found for a bar code definition.

Actions	Description
MULTIBARCODE_USEFIRST=0	Use the first bar code value found.
MULTIBARCODE_USELAST=1	Use the last bar code value found.
MULTIBARCODE_CLEAR=2	Do not use the bar code values.

The following properties apply for this class.

Properties	Description
String workspaceName	Name of the workspace with which this job is associated.
String workspaceID	ID of the workspace with which this job is associated.
String jobID	Job ID.
Date lastModifiedDateTime	Date and time the job was last modified.
String lastModifiedUserID	ID of the user that last modified the job.
String name	Job name.
String description	Job description.
String scriptID	ID of the script with which this job is associated.
List<BarcodeDefinition> barcodes	List of bar code definitions.
Boolean autoDetectBarcodes	This determines whether Enable Auto-detect Bar Codes is turned on.
Boolean validateCheckSum	This determines whether Validate Optional Checksum is turned on.
List<Integer> symbologies	A list of selected bar code symbologies for recognition: valid values are from 0 - 21, as defined in the constants for bar code symbologies earlier in this section.
Integer batchOrganization	Document organization type; valid values are from 0 - 4, as defined in the constants for document organization type earlier in this section.
Integer documentPageCount	For the "Fixed number of pages per document" document organization type, this property refers to the maximum number of pages per document.
Integer pagesPerDoc2ReadBarcodes	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.
Integer maxPageCountPerDoc	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.
BarcodeDefinition multiPageDocBarcode	For the "Same bar code value on each page" document organization type, this property refers to the bar code that determines document separation.
Boolean optimizeBarcodeDetection	For the "Same bar code value on each page" document organization type, this property determines whether to optimize bar code detection
List<SeparatorDefinition> coverPages	InFor 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.
Integer multiBarcodeValuesOption	Actions to take If more than one value is found for a bar code within a document; valid values are 0-2 as defined in the constants.

Properties	Description
Integer dynamicDocType	Options on how the Dynamic Document Profile is determined; valid values are 0-2 as defined in the constants.
String defaultDocTypeID	ID for the Default Document Profile.
BarcodeDefinition docTypeBarCode	When the Document Profile is being dynamically determined using the bar code, this property represents the selected bar code.
List<DocumentDefinition> docTypeMappings	When the Document Profile is being dynamically determined using the bar code, this mapping represents the Document Profile and Bar Code Value Mappings.
List<RecognitionJobField> jobFields	Field mappings information.
Integer dblookupUsing	The type of value the database lookup will be using; valid values are 0-2 as defined in the constants.
BarcodeDefinition dblookupBarcodeField	This is the bar code definition that is selected for database lookup.
String dblookupIndexDefID	This is the metadata field ID that is selected for database lookup.
String dblookupProfile	Database lookup profile ID.
String dblookupSearchField	Database lookup search field ID.
Integer dblookupMultipleRecordAction	Actions to take when more than one record is found during database lookup; valid values are 0-1 as defined in the constants.
Integer dblookupNoMatchAction	Actions to take when no record is found during database lookup; valid values are 0-1 as defined in the constants.
String renamePrefix	Part of post-process setting. When there is no system error, this is the batch prefix to rename, if needed.
String renameEmail	Part of post-process setting. When there is no system error, this is the email address to send email notification to rename, if needed
String renameStatus	Part of post-process setting. When there is no system error, this is the batch status to change, if needed.
Integer renamePriority	Part of post-process setting. When there is no system error, this is the batch priority to change, if needed
String processorID	Part of post-process setting. When there is no system error, this is the batch processor ID to which the batch will be released.
String processorJobID	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.
String failureRenamePrefix	Part of post-process setting. When there is a system error, this is the batch prefix to rename, if needed.
String failureRenameEmail	Part of post-process setting. When there is a system error, this is the email address to which notification should be sent, if needed.

Properties	Description
String failureRenameStatus	Part of post-process setting. When there is a system error, this is the batch status to change, if needed.
Integer failureRenamePriority	Part of post-process setting. When there is a system error, this is the batch priority to change, if needed.
String failureProcessorID	Part of post-process setting. When there is a system error, this is the batch processor ID to which the batch will be released.
String failureProcessorJobID	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.

5.2.2 BarcodeDefinition

The following are the constants for the bar code validation rule type.

Constants	Description
VALIDATION_NONE=0	No validation rule is specified.
VALIDATION_LENGTH=1	Use the bar code length to validate.
VALIDATION_MASK=2	Use the mask to validate.
VALIDATION_ REGULAREXPRESSION=3	Use a regular expression to validate.
VALIDATION_PICKLIST=4	Use a choice list to validate.

The following are the properties for this class.

Properties	Description
String barcodeName	Bar code definition name.
Integer validationRule	Bar code validation rule; valid values are 0-4, as defined in the constants.
Integer validationLength	Validation length.
String validationMask	Validation mask.
String validationRegularExpression	Validation regular expression.
String pickListSourceID	Validation choice list source ID.
String pickListID	Validation choice list ID.

5.2.3 SeparatorDefinition

The following are the properties for this class.

Properties	Description
String name	Name of the separator page.
Boolean deleteUponCommit	Determines whether to delete the separator page after commit.
Integer operator	Operator used for rules; valid values are 0 and 1, as defined in the RecognitionJob constants.

Properties	Description
String docTypeID	If the document type is dynamically determined based on a separator page, this is the ID of the document type for this separator page.
List<SeparatorRuleDefinition> rules	Rules for this separator page.

5.2.4 SeparatorRuleDefinition

The following are the properties for this class.

Properties	Description
String name	Name of the rule.
Integer operator	Operator used for patch code and bar codes selected; valid values are 0 and 1, as defined in the RecognitionJob constants.
String patchCode	Patch code selected for this rule.
List<String> barcodes	Bar codes selected for this rule.

5.2.5 DocumentDefinition

The following are the constants for the document type mapping option.

Constants	Description
BARCODE_VALUE=0	This determines document type based on bar code value.
BARCODE_PICKLIST=1	This determines document type based on values in the choice list.

These are the properties for the DocumentDefinition class.

Properties	Description
String docTypeID	Document type ID.
Integer mappingType	This sets whether to determine document type based on bar code value or choice list; valid values are 0 and 1, as defined in the constants.
String value	Bar code value specified.
String pickListSourceID	Choice list source ID specified.
String pickListID	Choice list ID specified.

5.2.6 RecognitionJobField

These are the constants for the auto-populate type.

Constants	Description
AUTOPOPULATE_NONE=0	Do not auto-populate the index value.
AUTOPOPULATE_BARCODE=1	Auto-populate the index value with the bar code value.
AUTOPOPULATE_BATCHNAME=2	Auto-populate the index value with the batch name.

Constants	Description
AUTOPOPULATE_DEFAULT=3	Auto-populate the index value with a default value.
AUTOPOPULATE_INDEXDATE=4	Auto-populate the index value with the index date.
AUTOPOPULATE_SCANDATE=5	Auto-populate the index value with the scan date.

These are the properties for the RecognitionJobField class.

Properties	Description
String indexDefID	Metadata ID to populate with property values.
Integer autoPopulate	Auto-populate type; valid values are 0-5, as defined in the constants.
String populateValue	For the bar code type, this represents the bar code definition name; for the default type, this represents a default value.

5.2.7 RecognitionProcessorContext

Properties	Description
Logger logger	Logger for user to log additional entries.
RecognitionJob job	Current job being used.
BatchLockEntity batchLockEntity	Current batch that is being processed.
CaptureWorkspaceEntity workspaceEntity	Current workspace that is being used.
int phaseID	An integer that identifies the current phase: 0 - not in a phase. Examples include pre-process validation and batch clean up. 1 - bar code recognition 2 - document organization 3 - document classification 4 - indexing 5 - document creation 6 - post processing
boolean cancelAction	In certain calls, the user is allowed to cancel the action (for example, bar code recognition or database lookup).
BatchItemEntity batchItem	Current batch item being processed. This is specifically used during bar code recognition and bar code validation (part of the document organization phase).
Integer patchCodeRead	Patch code found on a batch item. This is only used during the bar code recognition phase.
List<String> barcodesRead	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.

Properties	Description
List<ProcessorItem> validBarcodes	This is a 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.
ProcessorItem validBarcode	This is specific to the bar code that determines document separation, and specific to the optimized bar code recognition setting.
ProcessorSeparatorPage separator	This is called for organization types that involve a separator page. If the separator is null, then this batch item is not a separator page.
ProcessorDocument document	This is used for the document classification, indexing, and document creation phase. It contains everything the user needs to know about the document.
String dbLookupValue	This is only used before database lookup is executed. The user can change the lookup value.
String unIndexedDocTitle	This is specific to the Document Creation phase. This property allows the user 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.
String extractPath	Path to which batch items were extracted. This is specific during the bar code recognition phase. The user should not modify this property.

5.2.8 PostProcessContext

Properties	Description
String renameBatch	Name that the batch will be renamed to during post process. If null, the batch will not be renamed.
Integer priority;	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.
BatchStatusEntity status;	Status entity object that the batch will be associated with during post process. If null, the batch status will remain the same.
int batchState;	If there were some errors during the recognition process, the batch state will be preset to BATCH_STATE_ERROR; otherwise, the batch state will be preset to BATCH_STATE_READY.
List<String> emailRecipients;	A list of email recipients that email notification will be sent to. If empty, no email will be sent.
String emailSubject;	Subject line of the email notification.
String emailMessage;	Main message body of email notification. If empty, no email will be sent.
String processorID;	The processor ID to which the current batch will be released.
String processorJobID;	The processor job ID to which the current batch will be released.

5.2.9 ProcessorSeparatorPage

Properties	Description
boolean include	Indicates whether this separator page will be deleted after commit.
int level;	This is only used in the hierarchy separator pages organization type. Level starts with 1.
String name;	Separator page name.
String batchItemID;	The batch item with which this separator page is associated.

5.2.10 ProcessorItem

Properties	Description

String name;
String value;
	ProcessorItem is a class that holds name/value pair data. In the script, it is used to hold barcode name/values.

5.2.11 ProcessorDocument

Properties	Description
String title;	Title of the document, which is populated during the document creation phase.
List<String> batchItems;	All batch items associated with this document. This is populated during the document organization phase.
List<ProcessorItem> validBarcodes;	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.
int failureStatus;	Status of the current document. ■ 0 - no error ■ 1 - failed to validate bar code. This is the case when the 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.
String docTypeID;	Document type ID associated with the document. If null, no ID is determined.
String comment;	Comments for the document. It is usually error detail for 'failureStatus,' which the user can customize through script.
String captureDocID;	This is only used in the "One Document" batch organization type, where the processor does not organize documents, and does not create any Capture documents. This ID is the Capture document ID.
ProcessSeparatorPage separator;	Separator page of this document. This applies to the "One Document" and "multiple page document with separator" organization types.
List<ProcessSeparator Page> hierarchySeparators;	Separator pages for this document. This applies to the "multiple pages with hierarchy separator" organization type.

Properties	Description
List<IndexValue> indexValues;	List of metadata names and values.

5.3 Script Methods

This section describes script methods, which include the following:

- initialize (RecognitionProcessorContext rpc)
- processBatch (RecognitionProcessorContext rpc)
- restoreCaptureBatch (RecognitionProcessorContext rpc)
- beginPhase (RecognitionProcessorContext rpc)
- endPhase (RecognitionProcessorContext rpc)
- extractBatchItem(RecognitionProcessorContext rpc)
- barcodesFoundOnItem(RecognitionProcessorContext rpc)
- batchItemAllValidBarcodes (RecognitionProcessorContext rpc)
- determineSeparatorPage(RecognitionProcessorContext rpc)
- batchItemValidBarcode (RecognitionProcessorContext rpc)
- determineDocType(RecognitionProcessorContext rpc)
- beginDatabaseLookup(RecognitionProcessorContext rpc)
- determineIndexValues(RecognitionProcessorContext rpc)
- renameOrigCaptureDocTitle (RecognitionProcessorContext rpc)
- createCaptureDoc(RecognitionProcessorContext rpc)
- postProcess(PostProcessContext postContext)
- endBatchProcess()

5.3.1 initialize (RecognitionProcessorContext rpc)

This is the very first call the recognition processor makes to script. There is no batch identified yet.

Properties populated in rpc are:

- 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.
- phaseID: 0

5.3.2 processBatch (RecognitionProcessorContext rpc)

This is called before the Recognition Processor processes the batch.

- phaseID: 0

- batchLockEntity: 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: The user can set the flag to true to skip processing of a batch.

5.3.3 restoreCaptureBatch (RecognitionProcessorContext rpc)

When the following occurs:

- The job organization type requires Capture batch.
- There is only one image document.
- This batch has more than one document.
- The batch state indicates that the processor failed at the document creation phase.

The processor makes sure that both batch and job have not been modified since the last process. Under this circumstance, the processor attempts to restore the batch to its original state by removing previous documents created by the recognition process. This method is invoked prior to restoring the batch, which allows the user to cancel restore, and make the batch invalid for processing.

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

5.3.4 beginPhase (RecognitionProcessorContext rpc)

This indicates the beginning of a phase.

- phaseID: There are six different phases (see RecognitionProcessorContext phaseID for detail).
- 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.

5.3.5 endPhase (RecognitionProcessorContext rpc)

This indicates the end of a phase.

- phaseID: There are six different phases (see RecognitionProcessorContext phaseID for detail).

5.3.6 extractBatchItem(RecognitionProcessorContext rpc)

This happens during the barcode recognition phase. The processor extracts batch items one at a time into a directory right before the processor performs bar code recognition on the page. Then the processor informs the user where the items are.

- phaseID: 1
- extractPath: The directory where the batch items are located.

5.3.7 barcodesFoundOnItem(RecognitionProcessorContext rpc)

This is invoked after the processor processes the batch item, and collected and recognized bar codes on this item.

- 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.

5.3.8 batchItemAllValidBarcodes (RecognitionProcessorContext rpc)

This occurs after the recognition processor has finished validating bar codes on a specific batch item.

- phaseID: 2
- batchItem: Current batch item that is used to perform bar code validation.
- validBarCodes: A list of name/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 shouldn't change the name, or add or remove items from the list.

5.3.9 determineSeparatorPage(RecognitionProcessorContext rpc)

This occurs after the processor has validated whether this page is a separator. This method is only invoked when a separator is defined for a recognition job. Properties used in this calls are:

- phaseID: 2
- batchItem: Current batch item that is to determine whether it is a separator or not.
- validBarCodes: A list of name/value pair of the valid bar codes found on the batch item. This list includes all bar codes definitions in the recognition job.
- separator: This object is null unless this batch item is a valid separator. If you want to make changes, you need to either set it to null, or populate it with valid data.

Level is used for the hierarchy separator page type only. For the other organization type, this value is ignored. Level should begin with 1.

You can change the level determined by the processor. However, if the level does not fit into a recognition job definition, the processor uses either the highest level (level<=0) or lowest level (level>=max defined level).

5.3.10 batchItemValidBarcode (RecognitionProcessorContext rpc)

This method passes in one valid bar code recognized on this batch item. This call will only happen when the batch organization type is bar code on every page, and optimized bar code recognition is turned on,

When the processor cannot find a bar code on a page, it will try to determine the separator bar code value on the next page. This is called right after the processor has determined the bar code value on the next page.

- phaseID: 2
- batchItem: Next page batch item that is to determine the separator bar code value.

- validBarcode: Name/value pair for the separator bar code. User can change the value if needed.

5.3.11 determineDocType(RecognitionProcessorContext rpc)

This 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 processor is unable to identify it.

- phaseID: 3
- document: Contains the current document information. Some properties are specific to certain organization type. The docTypeID needs to be examined here, and changed if needed.

5.3.12 beginDatabaseLookup(RecognitionProcessorContext rpc)

This is called after the recognition processor has determined the lookup value, and before the actual execution of the lookup is called.

- phaseID: 4
- dbLookupValue: User can modify the lookupValue.
- cancelAction: User can cancel lookup.

5.3.13 determineIndexValues(RecognitionProcessorContext rpc)

This is called after the Recognition Processor has determined all metadata values for a particular processor document. The user gets a chance to modify values.

- phaseID: 4
- document: Contains the current document information. Some properties are specific to certain organization types. The indexValues needs to be examined here, and changed if needed.

5.3.14 renameOrigCaptureDocTitle (RecognitionProcessorContext rpc)

This is called before the processor renames the original document into "unindexed." This applies to all batch organization types except the "One Document Only" type.

- phaseID: 5
- unIndexedDocTitle: The user can change the title.

5.3.15 createCaptureDoc(RecognitionProcessorContext rpc)

Before the 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 "One Document Only" organization type, changing batch items does not affect the outcome.

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

5.3.16 postProcess(PostProcessContext postContext)

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

- postContext: The user can make changes to affect the post-process outcome, if needed.

5.3.17 endBatchProcess()

This indicates that the Recognition Processor has finished processing the batch.

6

Creating Import Processor Scripts

This chapter discusses creating Import Processor scripts.

The Import Processor provides many features, and if you need to customize the importing process, there are a variety of customization methods.

This chapter contains the following sections:

- Section 6.1, "Scripting for the Import Processor"
- Section 6.2, "Import Processor Classes"

6.1 Scripting for the Import Processor

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

If an Import Job specifies a script to use during processing, the Import Processor Bean will create an instance of the JDK's ScriptRuntime class and initialize it with the script specified in the job. The Import Processor Bean, Import Manager Bean, and import sources all share this scripting runtime.

This section defines the following events:

- Import Processor Events
- Email Source Events
- Folder Source Events
- List File Source Events

6.1.1 Import Processor Events

The following table provides definitions of Import Processor events.

Events	Description
`public void preProcess(ImportProcessorContext ctx);`	This event occurs prior to the pre-processing of the import source. Initialization code can be performed here.
`public process(ImportProcessorContext ctx;)`	This event signals the start of the import process.
`public void postProcess(ImportProcessorContext ctx);`	This event occurs after the import source has been processed. Clean-up code can be performed here.
`public void preCreateBatch(ImportProcessorContext ctx);`	This event occurs immediately after a new batch is started.
`public void postCreateBatch(ImportProcessorContext ctx);`	This event occurs immediately after a batch is created, but before any documents have been created.
`public void preCreateDocument(ImportProcessorContext ctx);`	This event occurs prior to a new document being created.
`public void postCreateDocument(ImportProcessorContext ctx);`	This event occurs after a new document has been created.
`public void preImportFile(ImportProcessorContext ctx);`	This event occurs prior to a file being imported.
`public void postImportFile(ImportProcessorContext ctx);`	This event occurs after a file is imported.
`public void preRelease(ImportProcessorContext ctx);`	This event occurs prior to a batch being released.
`public void postRelease(ImportProcessorContext ctx);`	This event occurs after a batch has been released.
`public void preDatabaseSearch(ImportProcessorContext ctx);`	This event occurs prior to a database lookup.
`public void processDatabaseSearchResults(ImportProcessorContext ctx);`	This event occurs after the database lookup has returned the search results.

6.1.2 Email Source Events

The following table provides descriptions of email source events.

Events	Description
`public void newMessage(ImportProcessorContext ctx, EmailSourceContext emailCtx);`	This event occurs when a new email message is about to be processed.
`public void newAttachment(ImportProcessorContext ctx, EmailSourceContext emailCtx);`	This event occurs when a new email attachment is about to be processed.
`public void deleteMessage(ImportProcessorContext ctx, EmailSourceContext emailCtx);`	This event occurs in the email message post-processing step when an email message is about to be deleted.
`public void moveMessage(ImportProcessorContext ctx, EmailSourceContext emailCtx);`	This event occurs in the email message post-processing step when an email message is about to be moved to an email folder.

6.1.3 Folder Source Events

The following table provides descriptions of folder source events.

Events	Description
`public void newFolder(ImportProcessorContext ctx, FolderSourceContext folderCtx);`	This event occurs when a new folder is about to be processed.
`public void deleteDocumentFile(ImportProcessorContext ctx, FolderSourceContext folderCtx);`	This event occurs in the folder post-processing step when a file from the folder is about to be deleted.
`public void renameDocumentFile(ImportProcessorContext ctx, FolderSourceContext folderCtx);`	This event occurs in the folder post-processing step when a file from the folder is about to be renamed.

6.1.4 List File Source Events

The following table provides descriptions of list file source events.

Events	Description
`public void newFolder(ImportProcessorContext ctx, ListFileSourceContext listFileCtx);`	This event occurs when a new folder containing list files is about to be processed.
`public void newListFile(ImportProcessorContext ctx, ListFileSourceContext listFileCtx);`	This event occurs when a new list file is about to be processed.
`public void newListFileLine(ImportProcessorContext ctx, ListFileSourceContext listFileCtx);`	This event occurs when a new line in the list file is about to be processed.
`public void deleteListFile(ImportProcessorContext ctx, ListFileSourceContext listFileCtx);`	This event occurs in the list file post-processing step when a list file is about to be deleted.
`public void renameListFile(ImportProcessorContext ctx, ListFileSourceContext listFileCtx);`	This event occurs in the list file post-processing step when a list file is about to be renamed.

6.2 Import Processor Classes

This section defines the following Import Processor classes:

- ImportJob
- ImportProcessorContext Class
- Capture Core Classes used by Import Processor
- Email Source Classes
- Folder Source Classes
- List File Source Classes

6.2.1 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. The following table defines the properties for an Import Job.

Property	Description
String jobID	A value that uniquely identifies the job in the system. This will be a GUID.
String workspaceID	The identifier of the workspace to which the job belongs.
String jobName	A human-readable name for the job.

Property	Description
String dbSearchID	The identifier of the database search to use when processing the job.
String dbSearchFieldID	The identifier of the database search field to use when processing the job.
Integer imageDownsample	This 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
Integer jpegQuality	The JPEG quality ratio 0 to 99.
String batchPrefix	The batch prefix to use when creating batch names.
String defaultBatchStatusID	The identifier of the batch status to associate with batches created by this job.
Integer defaultPriority	The default priority assigned to batches ranging from 0 to 10.
String defaultDocumentTypeID	The default document profile for documents created by this job.
Integer searchResultOption	Determines how to handle database lookups that return more than one result. 0 - Use the first record 1 - Ignore results (do not populate fields)
String scriptID	The unique identifier of a script to use for this job.
Integer importFrequency	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)
Integer hour	If the importFrequency is set to Daily, this specifies the hour of the day.
Integer minute	If the importFrequency is set to Daily, this specifies the minute of the day.
Date lastCheck	The date/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.
Map<String, FieldMappingInfo> fieldMappings	A set of values that map Capture fields to import source metadata fields.
String importSourceClassName	The name of the Java class that provides the implementation of the import source for this job.

Property	Description
ImportSourceConfiguration importSourceConfig	Contains the configuration for the import source defined for this job
String batchProcessorClassName	The 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.
String batchProcessorJobID	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.
Integer imageFailureAction	This is the action to take if an invalid image is encountered. 0 - Abort the batch 1 - Skip the item
Locale locale	Specifies the locale of the list file source.
String defaultDateFormat	Specifies the default date format of dates in the list file source.
String description	The description of this job.
String encoding	Specifies the file encoding of the list file source.
Boolean isJobOnline	Indicates whether this job should be processed.

6.2.2 ImportProcessorContext Class

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	Description
Boolean cancel	When this boolean value is set to True, it will cancel the operation being performed.
Boolean cancelDBSearch	When this boolean value is set to True, it will cancel the database lookup.
DBSearchResults dbSearchResults	This contains the results from a database lookup.
String sourceName	The name of the import source that the current Import Job is configured to use.
Logger logger	This is an instance of the Import Processor's Logger class, which can be used to log information related to processing.
ImportManagerSession importManager	This is the import manager session bean used in this context.
ScriptEngine scriptEngine	The scripting engine used to invoke methods in Import Processor scripts.
ImportJob importJob	This is the current Import Job being processed.
BatchLockEntity ble	This contains the batch lock entity for the batch, after a new batch has been created.
String importSourceFile	This is the name of the file currently being processed.

Property	Description
DocumentEntity documentEntity	The document entity associated with the file currently being processed.
DocumentPageEntity documentPageEntity	This is the document page entity associated with the file currently being processed.
ImportHATokenEntity importHATokenEntity	This is the high availability token associated with the current batch.
Integer lastMultiPageTiffNumber	This contains the current page number of a multi-page TIFF file being processed.
CaptureWorkspaceEntity workspaceEntity	This is the workspace entity associated with the current batch.
WorkspaceManager workspaceManager	This is the workspace manager associated with the current Import Job.

6.2.3 Capture Core Classes used by Import Processor

The following table describes the Capture Code classes used by the Import Processor.

Class	Properties	Description
oracle.odc.data.DBSearchResults	List<DBSearchResultRow>	A list of rows from the database lookup.
	List<DBSearchFieldInfo> fieldInfoList	A list of search field info describing the columns used in the database lookup.
oracle.odc.data.DBSearchResultRow	List<String> results	A list of results from the database lookup. Each item in the list represents a column.
oracle.odc.data.DBSearchFieldInfo	String captureIndexDefID	The ID of the Capture metadata field definition.
	String dbColumnName	The name of the database lookup column.
	Integer dbColumnType	The database lookup column type.
	Integer captureFieldType	The Capture metadata field definition type.

6.2.4 Email Source Classes

The following table describes email source classes. See the Javamail API documentation for the Folder and Message class definitions.

Class	Properties	Description
oracle.odc.importprocessor.email.EmailSourceContext	String account	The name of the email account currently being processed.
	Folder folder	The email folder currently being processed.
	Message message	The email message currently being processed.
	String attachmentFilename	The file name of the email message attachment currently being processed.

6.2.5 Folder Source Classes

The following table describes folder source classes.

Class	Properties	Description
oracle.odc.importprocessor.folder.FolderSourceContext	String folderName	The name of the directory currently being processed.
	String documentFilename	The name of the file currently being processed.
	String renamedDocumentFilename	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.

6.2.6 List File Source Classes

The following table describes list file source classes.

Class	Properties	Description
oracle.odc.importprocessor.listfile.ListFileSourceContext	String folderName	The name of the folder currently being processed.
	String listFilename	The name of the list file currently being processed.
	String listFileLine	The contents of the line currently being processed in the list file.
	String documentFilename	The name of the file currently being processed from the current line in the list file.
	String renamedListFilename	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.

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`