Oracle® Document Capture

Developer's Guide

Release 10*g*R3

E13864-01

November 2010

ORACLE®

Developer's Guide for Oracle Document Capture, Release 10*g*R3

E13864-01

Primary Author: Sarah Howland

Contributor: Ken Peterka, Rob Abbe, Dan Sievers, Jun Liang, Sara Johnson, Carl Diedrich, Vince Cook, and Richard Lindman

Contents

5 Import Server Macros

6 Customizing Capture

List of Figures

Preface

The *Developer's Guide for Oracle Document Capture* contains information to develop VBA-compatible (Visual Basic for Applications) macros to customize the Oracle Document Capture applications for your organization.

Audience

This document is intended for developers responsible for customizing Oracle Document Capture functionality.

Related Documents

For more information, see the following documents in the Oracle Document Capture Release 10*g*R3 documentation set:

- *Oracle Document Capture Release Notes*
- *Installation Guide for Oracle Document Capture*
- *Administrator's Guide for Oracle Document Capture*
- *User's Guide for Oracle Document 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.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Using Macros in Oracle Document Capture

This section covers the following topics:

1.1 About Oracle Document Capture Macros

Oracle Document Capture supports a Visual Basic for Applications (VBA) compatible programming language for writing and debugging macros. You can develop macros to customize the Oracle Document Capture components for your organization.

You can incorporate macros in these components:

- Scan
- Index
- Recognition Server
- Import Server
- Commit profiles

1.2 Steps For Developing and Incorporating Macros

Follow these steps to develop and incorporate macros into Oracle Document Capture components.

1. Display the Manage Macros window.
   - If developing a scan, index or commit (Electronic Document Provider) macro, choose **Manage Macros** from the System menu in Oracle Document Capture.
   - If developing a Recognition Server or Import Server macro, open the component, choose **Pause** from the Server menu, and choose **Macro Manager** from the Setup menu.
2. In the Category field, select the type of macro you want to create.

   Macros for the category you selected are listed.
3. Click the **New** button and enter a name and description for the macro.

A blank macro is added to the Macros list.

4. Click the **Design** button.

   The Design window displays.

   Using objects and events for the macro type you selected, develop the macro. For information about objects and events, refer to the section in this guide that corresponds to the macro type.

   Note: You can import and export macro files.

   For information about the setup window, see "Adding a Setup Window to a Macro" on page 1-3.

5. In the Capture component, assign the macro to a profile or batch job.

   For example, assign a scan macro to one or more scan profiles, and a Recognition Server macro to one or more Recognition Server batch jobs.

   To assign an EDP macro to a commit profile, create a commit profile in Capture Administration using the Electronic Document Provider commit driver and select the EDP macro.

1.2.1 Designing Macros

The Macro Editor allows you to design macros. Each Oracle Document Capture program will expose one or more event objects that you can select from the Object list. Once selected, the corresponding events (procedures) will appear in the Proc list.

The macro editor supports intellisense, which automatically displays the parameters needed for functions and procedures. Once you have developed a macro, you can quickly test to see if it will compile, by clicking the **Start** button. If an error is encountered within the macro, a message is displayed with the error description and line number. In addition, the line with the error number is highlighted in red.

1.3 Using ActiveX Libraries

Referencing ActiveX libraries allows you to select another application's objects that you want available in your code by setting a reference to that application's object library. When developing macros, set a reference to the Captovation Capture Data Objects library. This will expose the Capture (EDO) object hierarchy and unleash the power of IntelliSense, making your coding work go easier.

To see which object references are available, click the **References** button in the Manage Macros Design screen. (The first time you do this for the current macro, it may take a little while for the list to be populated with all possible entries.)

Place a checkmark next to libraries you wish to reference in your macro. One object is likely to be checked already, depending on the type of macro you are creating. For example, the Index library will be checked if you are writing an index macro, and so on.

1.3.1 Tips On Using References

The following information describes how to use references:

- If you are not using any objects in a referenced library, minimize the number of object references Visual Basic must resolve by unchecking references' checkboxes;

this reduces the time required to compile the macro code. You can only remove references for items not being used in your macro. If you remove a reference to an object currently used in your macro, you'll receive an error the next time you refer to that object.

- To avoid possibly breaking compatibility between versions of Capture, it is recommended that after developing a new macro you remove the reference to the Captovation Capture Data Objects library and retype EDO objects as regular objects. IntelliSense will cease to work, but the code itself will run fine.

In this example, Capture objects are declared as such:

```
Dim oPage As edoBatchPage
Dim oIdx As edoBatchPageIndex
```

In this example, Capture objects are retyped as regular objects after the reference to their object library has been removed:

```
Dim oPage As Object     'edoBatchPage
Dim oIdx  As Object     'edoBatchPageIndex
```

You can comment out the original EDO object types to help you remember which kind of EDO objects they are, and to facilitate switching back to using the reference when you do further development work on your macro.

1.4 Debugging Macros

Use commands in the Debug menu of the Design window to debug Oracle Document Capture macros. These options allow you to run a macro within the editor, then proceed to step through the code as the events are being called from the corresponding Capture program.

During execution of the macro, you can easily observe and change the values of variables. To aid in the debugging process, you can also set breakpoints on specific lines within the macro, so that the macro immediately stops on that line during execution.

1.5 Adding a Setup Window to a Macro

All Oracle Document Capture macros can support their own setup window. This macro development feature allows you to develop generic macros that can be easily configured for a wide variety of environments. For example, you could develop a generic Index macro that is capable of performing dollar amount validation on a per field basis. By implementing a setup window, the macro could allow an administrator to enter the field name(s) to validate. As a result, the macro would not have to be modified to support different fields.

Settings are stored on a per macro basis. Therefore, it is possible to import a macro multiple times and specify different settings for each instance. Once developed, you can display the setup window for each macro instance. You select the macro from the Macros list (as described in "Steps For Developing and Incorporating Macros" on page 1-1) and click the **Setup** button on the Manage Macros toolbar.

To support a macro setup window, the macro must respond to two events in the eCaptureMacro object: LoadGlobalSettings and Setup. (The Connect event fires first during normal macro execution as well as during Setup.) The **LoadGlobalSettings** event occurs each time the macro is started, which allows the macro to retrieve the settings it needs to run correctly. The Setup event occurs only when the user clicks the Setup button.

Event	Connect
Description	The eCapture connection object must be passed into this event. This object will allow a macro to access Capture data during setup and prevent the need for a second login. This event should be the first event fired during normal macro execution as well as during Setup.
Syntax	eCaptureMacro_Connect(Connection as Object)
Parameter	Connection: Represents a connection to Capture. (See "Connection Object" on page 6-12.)

Event	LoadGlobalSettings
Description	This event is generated as soon as a macro is executed. This event allows a macro to obtain any user-defined settings.
Syntax	eCaptureMacro_LoadGlobalSettings(ByVal MacroName As String, ByVal Settings As Object)
Parameters	MacroName: The user-defined name of the macro being executed. Settings: see "Settings Object" on page 6-15

Event	Setup
Description	This event is generated when a user selects a macro and clicks the Setup button within the Manage Macros window.
Syntax	eCaptureMacro_Setup(ByVal MacroName As String, ByVal Settings As Object)
Parameters	MacroName: The user-defined name of the macro being executed. Setting: see "Settings Object" on page 6-15

To create a setup dialog, use the built-in User Dialog Editor.

For information about using and calling Dialog functions, press **F1** and refer to the macro help.

2

Scan Macros

This section covers the following topics:

2.1 About Scan Macros

Many events keep you notified of Scan activity. Below are some ideas for developing Scan macros:

- Automatically delete images files after they have been imported.
- Reorder image files as they are imported.
- Perform additional auditing of user activity.
- Prevent scan operators from deleting batches.

2.1.1 Scan Event Names

Scan event names between Scan for ISIS and Scan for Adrenaline are different. Scan for ISIS events are prefixed with *ISISScanTool* while Scan for Adrenaline events are prefixed with *ecScan*. However, many of the base event names are the same between Scan programs. For a list of Scan events, see "Scan for ISIS Events" on page 2-2 or "Scan for Adrenaline Events" on page 2-9.

2.1.2 Order of Events For Scanning or Importing

The following is the basic order of events during batch scanning or file importing:

1. ecScan_BeginScanning
2. ecScan_ScanStart
3. ecScan_ScanPageStart
4. ecScan_ScanSetEndorseText
5. ecScan_ScanPageEnd
6. ecScan_ScanPageDone
7. ecScan_ScanDone

2.2 Scan for ISIS Events

Choose from the following events.

Event	BatchDelete
Description	This event occurs when a batch is deleted.
Syntax	ISISScanTool_BatchDelete(BatchNames() As String, Cancel As Boolean)
Parameters	*BatchNames* is an array of batches being deleted. *Cancel* set to True cancels the delete operation.

Event	BatchFilter
Description	This event occurs when the Batch Scanning screen is being refreshed. The event is fired for every batch that matches the specified prefix. This allows a macro to control the batches that are displayed in the list.
Syntax	ISISScanTool_BatchFilter(Batch As Object, Cancel As Boolean)
Parameters	*Batch* is the Capture batch object that represents the batch (see "Batch Object" on page 6-10). *Cancel* set to True prevents the batch from being displayed in the batch list.

Event	BatchNoteAdded
Description	This event occurs when a note is added to a batch.
Syntax	ISISScanTool_BatchNoteAdded(Note As String, BatchName As String)
Parameters	*Note* is the text added to the note. *BatchName* is the name of the batch affected.

Event	BatchNoteDeleted
Description	This event occurs when a note is deleted.
Syntax	ISISScanTool_BatchNoteDeleted(BatchName As String)
Parameters	*BatchName* is the name of the batch affected.

Event	BatchPriorityEdit
Description	This event occurs when the user modifies the priority of a batch.
Syntax	ISISScanTool_BatchPriorityEdit(BatchName As String, NewPriority as Integer, OldPriority As Integer, Cancel As Boolean)
Parameters	*BatchName* is the name of the batch affected. *NewPriority* is the new priority number being assigned to the batch. *OldPriority* is the previous priority number of the batch. *Cancel* set to True cancels the priority update for the batch.

Event	BatchScanComplete
Description	This event occurs when the batch scan finishes.
Syntax	ISISScanTool_BatchScanComplete()
Parameters	None

Event	BatchSearch
Description	This event occurs just before searching of a batch.
Syntax	ISISScanTool_BatchSearch(SearchCriteria As Object, Cancel As Boolean)
Parameters	*SearchCriteria* object contains the batch criteria being sought (see "SearchCriteria Object" on page 6-15). *Cancel* set to True cancels the search operation.

Event	BatchStatusEdit
Description	This event occurs when the user modifies the status of a batch.
Syntax	ISISScanTool_BatchStatusEdit(BatchName As String, NewStatus As String, OldStatus As String, Cancel as Boolean)
Parameters	*BatchName* is the name of the batch affected. *NewStatus* is the value of the new status being assigned to the batch. *OldStatus* is the previous batch status value. *Cancel* set to True cancels the status update for the batch.

Event	BeginScanning
Description	This event occurs just after the user clicks the Begin Scanning button.
Syntax	ISISScanTool_BeginScanning(Cancel As Boolean)
Parameters	*Cancel* set to True cancels the operation.

Event	CreateBatch
Description	This event occurs when the batch is created.
Syntax	ISISScanTool_CreateBatch(BatchName as String, Cancel As Boolean)
Parameters	*BatchName* is the name of the batch being created. *Cancel* set to True cancels the operation.

Event	DisplayInfo
Description	This event occurs just before the Batch Profile Information window is displayed.
Syntax	ISISScanTool_DisplayInfo(Cancel As Boolean)
Parameters	*Cancel* set to True cancels the operation.

Event	ImportFilesSelected
Description	This event occurs after a user selects files to import. Within this event, you could reorder the files in the FileList object.
Syntax	ISISScanTool_ImportFilesSelected(FileList() As String, ImportMode As Long, Cancel As Boolean)
Parameters	*FileList* is an array of file names selected by the user. *ImportMode* is the method used to import the files. Possible values are: SM_APPEND = 1, SM_INSERT = 2, SM_REPLACE = 3, SM_TEST = 4. *Cancel* set to True cancels the file importing.

Event	ImportPostProcess
Description	This event occurs immediately after a file is imported. You could use this event to delete a file after importing.
Syntax	ISISScanTool_ImportPostProcess(ByVal FileName As String)
Parameters	*FileName* the name of the file imported.

Event	ImportPreProcess
Description	This event occurs just before an image file is imported.
Syntax	ISISScanTool_ImportPreProcess(ByVal FileName As String, CancelPage As Boolean, CancelScan As Boolean)
Parameters	*FileName* is the name of the file to import. Setting *CancelPage* to True will cause Scan to skip importing of the file. Setting *CancelScan* to True will cause Scan to abort the importing process.

Event	MacroStart
Description	This event occurs when the macro code starts to execute. This occurs when a profile linked to the macro is selected in the Batch Scanning screen. **Note**: Use eCaptureMacro_Connect instead of the MacroStart event. For more information, see "Adding a Setup Window to a Macro" on page 1-3 and "Connection Object" on page 6-12.
Syntax	ISISScanTool_MacroStart(ToolProxy As Object)
Parameters	*ToolProxy* is an ECO object which provides a connection to the Capture data objects and the Capture client objects.

Event	MacroStop
Description	This event occurs when macro execution is stopped. This can happen if the Batch Scanning screen is closed or if another profile is selected.
Syntax	ISISScanTool_MacroStop()

Event	MacroStop
Parameters	None

Event	RenameBatch
Description	This event is called if a user attempts to rename a batch.
Syntax	ISISScanTool_RenameBatch(ByVal Batch as Object, NewBatchName as String, Cancel As Boolean)
Parameters	*Batch* is the Capture batch object that represents the batch (see "Batch Object" on page 6-10). *NewBatchName* is the name of the new batch. Set *Cancel* to True to abort the batch rename.

Event	RenameBatchComplete
Description	This event is called after a user has renamed a batch.
Syntax	ISISScanTool_RenameBatchComplete(ByVal Batch as Object, OriginalBatchName as String)
Parameters	*Batch* is the Capture batch object that represents the batch (see "Batch Object" on page 6-10). *OriginalBatchName* is the previous batch name.

Event	ReviewAppend
Description	This event occurs when a user clicks the append button on the Batch Review toolbar.
Syntax	ISISScanTool_ReviewAppend(Cancel As Boolean)
Parameters	*Cancel* set to True cancels the operation.

Event	ReviewBatch
Description	This event occurs just before the Review Batch window is displayed.
Syntax	ISISScanTool_ReviewBatch(ByVal Batch as Object, Cancel As Boolean)
Parameters	*Batch* is the Capture batch object that represents the batch being reviewed (see "Batch Object" on page 6-10). *Cancel* set to True cancels the operation.

Event	ReviewChangePage
Description	This event occurs whenever a new page is displayed in the Batch Review window.
Syntax	ISISScanTool_ReviewChangePage (PageNum As Integer, FileName As String)
Parameters	*PageNum* is the page number of the currently displayed page. *FileName* is the complete path and file name of the currently displayed page.

Event	ReviewClose
Description	This event occurs when the Review Batch window closes.
Syntax	ISISScanTool_ReviewClose()
Parameters	None

Event	ReviewCopyPages
Description	This event occurs when pages are copied within a batch using the thumbnails.
Syntax	ISISScanTool_ReviewCopyPages(nPages() As Integer, Cancel As Boolean)
Parameters	*nPages* is an array containing the page numbers to be copied. *Cancel* set to True will cancel the operation.

Event	ReviewDeletePages
Description	This event occurs just before pages are deleted.
Syntax	ISISScanTool_ReviewDeletePages(nPages() As Integer, Cancel As Boolean)
Parameters	*nPages()* is an array containing the page numbers to be deleted. *Cancel* set to True cancels the delete operation.

Event	ReviewInsert
Description	This event occurs when a user clicks the Insert Page button on the Review Batch toolbar.
Syntax	ISISScanTool_ReviewInsert(PageNum As Integer, Cancel As Boolean)
Parameters	*PageNum* is the page number to be inserted. *Cancel* set to True will cancel the operation.

Event	ReviewKeyDown
Description	This event occurs when the user presses down a key while the Reviewing screen is active.
Syntax	ISISScanTool_ReviewKeyDown(KeyCode As Integer, Shift As Integer)
Parameters	*KeyCode* is an integer that represents a key pressed on the keyboard. *Shift* is an integer that corresponds to the state of the SHIFT, CTRL, and ALT keys at the time of the event. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT are pressed, the value of shift is 6.

Event	ReviewKeyPress
Description	This event occurs when a user presses a keyboard key while the Reviewing screen is active.
Syntax	ISISScanTool_ReviewKeyPress(KeyAscii As Integer)
Parameters	*KeyAscii* is an integer that returns a standard numeric ASCII keycode. Keyascii is passed by reference.

Event	ReviewMovePages
Description	This event occurs when the user moves pages within a batch using thumbnails.
Syntax	ISISScanTool_ReviewMovePages(nPages() as Integer, Cancel As Boolean)
Parameters	*nPages()* is an array containing the page numbers of the pages to be moved. *Cancel* set to True will cancel the operation.

Event	ReviewOpen
Description	This event occurs when the user clicks the Review Batch button.
Syntax	ISISScanTool_ReviewOpen(ScanUI As Object)
Parameters	See "ScanUI Object" on page 6-14.

Event	ReviewPostDeletePages
Description	This event is called after a user has deleted batch pages from within the Review window.
Syntax	ISISScanTool_ReviewPostDeletePages(nPages() As Integer)
Parameters	*nPages()* is an array containing the page numbers deleted.

Event	ReviewPreDeletePages
Description	This event is called just before a user deletes batch pages from within the Review window.
Syntax	ISISScanTool_ReviewPreDeletePages(Cancel as Boolean)
Parameters	Set *Cancel* to True to abort the operation.

Event	ReviewReplace
Description	This event occurs when a user clicks the Replace button on the Review Batch toolbar.
Syntax	ISISScanTool_ReviewReplace(PageNum As Integer, Cancel As Boolean)
Parameters	*PageNum* is the page number to be replaced. *Cancel* set to True will cancel the operation.

Event	ScanDone
Description	This event occurs just after scanning into a batch.
Syntax	ISISScanTool_ScanDone(TotalImages as Integer)
Parameters	*TotalImages* is the total number of images scanned.

Event	ScannerSettings
Description	This event occurs just before the Scanner Settings dialog is displayed.
Syntax	ISISScanTool_ScannerSettings (Cancel As Boolean)
Parameters	*Cancel* set to True cancels the operation.

Event	ScanPageDone
Description	This event occurs just after the scanned image is saved to disk.
Syntax	ISISScanTool_ScanPageDone(PageState As Integer)
Parameters	*PageState* is the page state value, where 1=Remove page and 2=Keep page.

Event	ScanPageEnd
Description	This event occurs just after a page is scanned.
Syntax	ISISScanTool_ScanPageEnd(ByVal PEFront As Boolean, ByVal PEImageSize As Long, ByVal PEImageHeight As Integer, ByVal PEImageWidth As Integer, Cancel As Boolean)
Parameters	*PEFront* is a boolean representing whether the page is the front side of an image if duplex scanning. *PEImageSize* is a long representing the size in bytes of the image. *PEImageHeight* is the height of the image in pixels. *PEImageWidth* is the width of the image in pixels. If *Cancel* is set to true, the page will be discarded.

Event	ScanPageStart
Description	This event occurs just before the scanned image is scanned.
Syntax	ISISScanTool_ScanPageStart(BatchPage as Object, PreviewMode as Boolean)
Parameters	*BatchPage* is the Capture BatchPage object that will use the image about to be scanned. *PreviewMode* is True if scanning in preview mode (in which case the BatchPage object will not be applicable.)

Event	ScanPreview
Description	This event occurs just before the Scan Preview operation.

Event	ScanPreview
Syntax	ISISScanTool_ScanPreview (Cancel As Boolean)
Parameters	*Cancel* set to True cancels the operation.

Event	ScanProfileSelected
Description	This event is called after a user has selected a Scan Profile.
Syntax	ISISScanTool_ScanProfileSelected(Profile As String, ScanUI as Object)
Parameters	*Profile* is the name of the profile selected. *ScanUI* refers to the Scanning User Interface object (See "ScanUI Object" on page 6-14.)

Event	ScanSetAnnotateText
Description	Allows a macro to change the annotation text.
Syntax	ISISScanTool_ScanSetAnnotateText(AnnotateText As String)
Parameters	*AnnotateText* is the text to be annotated to the scanned image.

Event	ScanSetEndorseText
Description	Allows a macro to determine the text that will be endorsed/imprinted. **Note**: Manual endorsing must be enabled for a scan profile.
Syntax	ISISScanTool_ScanSetEndorseText(EndorseText As String)
Parameters	*EndorseText* the text to be passed to scanner for endorsing. In most instances, the character '#' is recognized as a counter but the text meaning ultimately is determined by the ISIS driver.

Event	ScanStart
Description	This event occurs just before scanning into a batch begins.
Syntax	ISISScanTool_ScanStart(ByVal Batch As Object, oScanControl as Object)
Parameters	*Batch* is the eCapture Batch object to be scanned into. *oScanControl* is an object that exposes methods to allow a macro to Get/Set specified ISIS tag values.

2.3 Scan for Adrenaline Events

Choose from the following events:

Event	BatchDelete
Description	This event occurs when a batch is deleted.
Syntax	ecScan_BatchDelete(sBatches() As String, *Cancel* As Boolean)
Parameters	*sBatches* is an array of batches being deleted. *Cancel* set to True cancels the delete operation.

Event	BatchNoteAdded
Description	This event occurs when a note is added to a batch.
Syntax	ecScan_BatchNoteAdded(Note As String, BatchName As String)
Parameters	*Note* is the text added to the note. *BatchName* is the name of the batch affected.

Event	BatchNoteDeleted
Description	This event occurs when a note is deleted.
Syntax	ecScan_BatchNoteDeleted(BatchName As String)
Parameters	*BatchName* is the name of the batch affected.

Event	BatchPriorityEdit
Description	This event occurs when the user modifies the priority of a batch.
Syntax	ecScan_BatchPriorityEdit(BatchName As String, NewPriority as Integer, OldPriority As Integer, Cancel As Boolean)
Parameters	*BatchName* is the name of the batch affected. *NewPriority* is the new priority number being assigned to the batch.*OldPriority* is the previous priority number of the batch.*Cancel*set to True cancels the priority update for the batch.

Event	BatchScanComplete
Description	This event occurs when the batch scan finishes.
Syntax	ecScan_BatchScanComplete(ByVal Batch as Object)
Parameters	*Batch* is the Capture batch object that represents the batch being completed (see "Batch Object" on page 6-10).

Event	BatchSearch
Description	This event occurs just before searching of a batch.
Syntax	ecScan_BatchSearch(SearchCriteria As Object, Cancel As Boolean)

Event	BatchSearch
Parameters	*SearchCriteria* object contains the batch criteria being sought (see "SearchCriteria Object" on page 6-15). Cancel set to True cancels the search operation.

Event	BatchStatusEdit
Description	This event occurs when the user modifies the status of a batch.
Syntax	ecScan_BatchStatusEdit(BatchName As String, NewStatus As String, OldStatus As String, Cancel as Boolean)
Parameters	*BatchName* is the name of the batch affected. *NewStatus* is the value of the new status being assigned to the batch. *OldStatus* is the previous batch status value. Cancel set to True cancels the status update for the batch.

Event	BeginScanning
Description	This event occurs just after the user clicks the Begin Scanning button.
Syntax	ecScan_BeginScanning (Cancel As Boolean)
Parameters	Cancel set to True cancels the operation.

Event	CreateBatch
Description	This event occurs when the batch is created.
Syntax	ecScan_CreateBatch(BatchName as String)
Parameters	*BatchName* is the name of the batch being created.

Event	DisplayInfo
Description	This event occurs just before the Scan Profile Information window is displayed.
Syntax	ecScan_DisplayInfo(Cancel As Boolean)
Parameters	*Cancel* set to True cancels the operation.

Event	ImportFilesSelected
Description	This event occurs after a user selects files to import. Within this event, you could reorder the files in the FileList object.
Syntax	ecScan_ImportFilesSelected(FileList As Object, ImportMode as Long, Cancel As Boolean)

Event	ImportFilesSelected

Parameters	FileList is a collection of file names selected by the user. ImportMode is the method used to import the files. Possible values are: SM_APPEND = 1, SM_INSERT = 2, SM_REPLACE = 3, SM_TEST = 4. Cancel set to True cancels the file importing.
	The following is an example of how to loop through the FileList collection:
	`For nIndex = 1 to FileList.Counts`
	`Next`
	`FileName = FileList.Item(nIndex)`

Event	ImportPostProcess
Description	This event occurs immediately after a file is imported. You could use this event to delete a file after importing.
Syntax	ecScan_ImportPostProcess(ByVal FileName As String)
Parameters	*FileName* the name of the file imported.

Event	ImportPreProcess
Description	This event occurs just before an image file is imported.
Syntax	ecScan_ImportPreProcess(ByVal FileName As String, CancelPage As Boolean, CancelScan As Boolean)
Parameters	*FileName* is the name of the file to import. Setting *CancelPage* to True will cause Scan to skip importing of the file. Setting *CancelScan* to True will cause Scan to abort the importing process.

Event	RenameBatch
Description	This event is called if a user attempts to rename a batch.
Syntax	ecScan_RenameBatch(ByVal Batch As Object, NewBatchName As String, Cancel As Boolean)
Parameters	*Batch* is the Capture batch object that represents the batch (see "Batch Object" on page 6-10). *NewBatchName* is the name of the new batch. Set *Cancel* to True to abort the batch rename.

Event	RenameBatchComplete
Description	This event is called after a user has renamed a batch.
Syntax	ecScan_RenameBatchComplete(ByVal Batch As Object, OriginalBatchName As String)
Parameters	*Batch* is the Capture batch object that represents the batch (see "Batch Object" on page 6-10). *OriginalBatchName* is the previous batch name.

Event	ReviewAppend
Description	This event occurs when a user clicks the append button on the Batch Review toolbar.
Syntax	ecScan_ReviewAppend(Cancel As Boolean)
Parameters	*Cancel* set to True cancels the operation.

Event	ReviewBatch
Description	This event occurs just before the Review Batch window is displayed.
Syntax	ecScan_ReviewBatch(ByVal Batch as Object, Cancel As Boolean)
Parameters	*Batch* is the Capture batch object that represents the batch being reviewed (see "Batch Object" on page 6-10). *Cancel* set to True cancels the operation.

Event	ReviewChangePage
Description	This event occurs whenever a new page is displayed in the Batch Review window.
Syntax	ecScan_ReviewChangePage(PageNum As Integer, FileName As String)
Parameters	*PageNum* is the page number of the currently displayed image. *FileName* is the complete path and file name of the currently displayed image.

Event	ReviewClose
Description	This event occurs when the Review Batch window closes.
Syntax	ecScan_ReviewClose()
Parameters	None

Event	ReviewCopyPages
Description	This event occurs when pages are copied within a batch using thumbnails.
Syntax	ecScan_ReviewCopyPages(nPages() As Integer, Cancel As Boolean)
Parameters	*nPages* is an array containing the page numbers of the pages about to be copied. *Cancel* set to True will cancel the operation.

Event	ReviewDeletePages
Description	This event occurs just before an image is deleted.
Syntax	ecScan_ReviewDeletePages(nPages() As Integer, Cancel As Boolean)

Event	ReviewDeletePages
Parameters	*nPages()* is an array containing the page numbers of the pages about to be deleted. *Cancel* set to True cancels the delete operation.

Event	ReviewInsert
Description	This event occurs when a user clicks the Insert Page button on the Review Batch toolbar.
Syntax	ecScan_ReviewInsert(PageNum As Integer, Cancel As Boolean)
Parameters	*PageNum* is the page number to be replaced. Cancel set to True will cancel the operation.

Event	ReviewKeyDown
Description	This event occurs when the user releases a key while the Batch Review window is active.
Syntax	ecScan_ReviewKeyDown(KeyCode As Integer, Shift As Integer)
Parameters	*KeyCode* is an integer that represents a key press on the keyboard. (See "Keycodes" on page A-1). *Shift* is an integer that corresponds to the state of the SHIFT, CTRL, and ALT keys at the time of the event. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT are pressed, the value of shift is 6.

Event	ReviewKeyPress
Description	This event occurs when a user presses a keyboard key while the Batch Review window is active.
Syntax	ecScan_ReviewKeyPress(KeyAscii As Integer)
Parameters	*KeyAscii* is an integer that returns a standard numeric ANSI keycode. *Keyascii* is passed by reference.

Event	ReviewMovePages
Description	This event occurs when pages are moved within a batch via thumbnails.
Syntax	ecScan_ReviewMovePages(nPages() as Integer, Cancel As Boolean)
Parameters	*nPages()* is an array containing the page number of pages that are about to be moved. *Cancel* set to True will cancel the operation.

Event	ReviewOpen
Description	This event occurs when the user clicks the Review Batch button.
Syntax	ecScan_ReviewOpen(ScanUI As Object)
Parameters	See "ScanUI Object" on page 6-14.

Event	ReviewPostDeletePages
Description	This event is called after a user has deleted batch pages from within the Review window.
Syntax	ecScan_ReviewPostDeletePages(nPages() As Integer)
Parameters	*nPages()* is an array containing the page numbers deleted.

Event	ReviewPreDeletePages
Description	This event is called just before a user deletes batch pages from within the Review window.
Syntax	ecScan_ReviewPreDeletePages(Cancel As Boolean)
Parameters	Set *Cancel* to True to abort the operation.

Event	ReviewReplace
Description	This event occurs when a user clicks the Replace button on the Review Batch toolbar.
Syntax	ecScan_ReviewReplace(PageNum As Integer, Cancel As Boolean)
Parameters	*PageNum* is the page number to be replaced. *Cancel* set to True will cancel the operation.

Event	ScanDone
Description	This event occurs just after scanning into a batch.
Syntax	ecScan_ScanDone(TotalImages as Integer)
Parameters	*TotalImages* is the total number of images scanned.

Event	ScanGetAnnotateText
Description	(Reserved for future use.) Allows a macro to change the annotation text.
Syntax	ecScan_ScanGetAnnotateText(AnnotateText As String)
Parameters	None

Event	ScanGetEndorseText
Description	Allows a macro to determine the text that will be endorsed/imprinted. **Note**: Manual endorsing must be enabled for a scan profile.
Syntax	ecScan_ScanGetEndorseText(EndorseText As String)
Parameters	None

Event	ScannerSettings
Description	This event occurs just before the Scanner Settings dialog is displayed.
Syntax	ecScan_ScannerSettings(Cancel As Boolean)
Parameters	*Cancel* set to True cancels the operation.

Event	ScanPageDone
Description	This event occurs just after the scanned image is saved to disk.
Syntax	ecScan_ScanPageDone(PageState as Integer)
Parameters	*PageState* is the page state value, where 1=Remove page and 2=Keep page.

Event	ScanPageEnd
Description	This event occurs just after a page is scanned.
Syntax	ecScan_ScanPageEnd(ByVal PEFront As Boolean, ByVal PEImageSize As Long, ByVal PEImageHeight As Integer, ByVal PEImageWidth As Integer, Cancel As Boolean)
Parameters	*PEFront* is a Boolean representing whether the current image is the front side of a scanned page (for duplex scanning). *PEImageSize* is the size of the image in bytes. *PEImageHeight* is the height of the image in pixels. *PEImageWidth* is the width of the image in pixels. If *Cancel* is set to True, the page is discarded.

Event	ScanPageStart
Description	This event occurs just before the scanned image is scanned.
Syntax	ecScan_ScanPageStart(ctlKimgp As Object)
Parameters	*ctlKimgp* refers to the Kofax Kimgp control, which is a component of the Kofax ImageControls toolkit. The toolkit is available for purchase from Kofax.

Event	ScanPreview
Description	This event occurs just before the Scan Preview operation.
Syntax	ecScan_ScanPreview(Cancel As Boolean)
Parameters	*Cancel* set to True cancels the operation.

Event	ScanProfileSelected
Description	This event is called after a user has selected a Scan Profile.
Syntax	ecScan_ScanProfileSelected(ByVal ProfileName As String, ByVal ScanUI As Object)
Parameters	*Profile* is the name of the profile selected. *ScanUI* refers to the Scanning User Interface object (see "ScanUI Object" on page 6-14).

Event	ScanStart
Description	This event occurs just before scanning into a batch begins.
Syntax	ecScan_ScanStart(ByVal Batch As Object, ctlKscan As Object, ctlKimgp As Object)
Parameters	*ctlKscan* refers to the Kofax KScan control, which is a component of the Kofax ImageControls toolkit. The toolkit is available for purchase from Kofax. *CtlKimgp* refers to the Kofax Klmgp control, which is a component of the Kofax ImageControls toolkit. The toolkit is available for purchase from Kofax.

3

Index Macros

This section covers the following topics:

3.1 About Index Macros

While an Index user is indexing documents, many events keep you informed of user activity. You can write code in these events to modify the default behavior of Index. Below are some ideas for developing Index macros:

- Develop a Mod 10 check digit routine for a field to ensure accurate data entry.
- Restrict a user from leaving a field if the value is invalid.
- Restrict a user from leaving a page if the value is invalid.
- Trap keystrokes to prevent invalid characters from being typed.
- Preprocess OCR data to filter out unwanted characters.
- Automatically search databases that are not accessible via the Database Lookup feature.

3.2 Index Objects

This section discusses the different index objects.

3.2.1 ctlIndexing Object

Description: This object represents the fields and view.

Properties		
ActiveFieldName	String	Returns the name of the field that has focus.
ActiveFieldValue	String	Returns the value of the field that has focus.

Properties		
DisplayMode	Integer	Sets/returns the relative position of the image panel to fields panel in Indexing control. dmImageRight = 0, image is to the right of the fields dmImageLeft = 1, image is to the left of the fields dmImageTop = 2, image is above the fields dmImageBottom = 3, image is below the fields
DisplayZone	Boolean	Sets/returns whether zone should be drawn on the image when a field with zone defined is active.
FileName	String	Sets/returns the file name of the currently displayed image.
LeftBandingMode	Integer	Sets/returns the action that will be taken when the user draws a rectangle on the image with the left mouse button down. bmNone = 0, no action is taken bmCreateZone = 1, action is to create a zone on the image bmZoomToRectangle = 2, action is zoom to the rectangle area
LockEdit	Boolean	Sets/returns whether the active field value can be modified or not.
MaxLength	Integer	Sets/returns the maximum length allowed for the active field.
Page	Integer	Sets/returns the page number that is in display.
RightBandingMode	Integer	Sets/returns the action that will be taken when the user draws a rectangle on the image with the right mouse button down. bmNone = 0 no action is taken bmCreateZone = 1 action is to create a zone on the image bmZoomToRectangle = 2 action is zoom to the rectangle area bmCustom=3=custom via macro
Rotation	Integer	Sets/returns the rotation in degrees of the current image in the viewer.
RowIndex	Integer	Sets/returns the currently active field position.
ScaleToGray	Boolean	Sets/returns whether the image is scaled to gray.
SplitPercent	Single	Sets/returns the splitter bar position.
TextBoxFont	StdFont	Sets/returns the font used in the field value.

Properties		
Zoom	Integer	Sets/returns the zoom percentage of the current image in the viewer.

Methods		
AddField	Add a field into the Indexing control.	AddField(FieldName As String, FieldType As enumFieldTypes, FieldValue As String, InputMask As String, DoubleKey As Boolean, Left As Long, Top As Long, Width As Long, Height As Long, Locked As Boolean, sFormat As String, Length As Long) FieldName: The name of the field. FieldType: ftFreeText = 0 for fields that do not have a mask or a pick list. ftMaskText = 2 for fields that have a mask defined. ftCombo = 3 for fields that have a pick list. FieldValue: The value of the field. InputMask: The mask defined for the field (if any). DoubleKey: Indicates whether this field needs double key verification. Left, Top, Width, and Height: Zone location (if defined) for the field. Locked: Indicates whether the user is allowed to change the field value. sFormat: The field's input format. Length: The field's maximum number of characters allowed.
AddPickListValue	Adds a pick list item to a pick list field. **Note**: The field must be defined as a pick list for the profile.	AddPickListValue(FieldName as String, PickListValue As String) FieldName: The name of the field to add the pick list item to. FieldValue: The pick list item value to add.
ClearAllLines	Clears values of all fields in the Indexing control.	
ClearLine	Clears the value of the active field in the Indexing control.	
ClearPickList	Clear the pick list for a field.	ClearPickList(FieldName as String) FieldName: The name of the field to clear.
FieldCaption	Returns the caption of the field displayed in the Indexing control.	FieldCaption(FieldName As String) As String FieldName: The name of the field whose caption you want.

Methods		
FieldValue	Returns the value of a field.	FieldValue(FieldName As String) As String FieldName: The name of the field to obtain the value from.
FitToWindow	Resize the image so that the whole image fits inside the image panel of the Indexing control.	FitToWindow()
PrintImage	Sends the currently displayed image to the printer.	PrintImage()
Refresh	Refreshes the image and/or index fields.	RefreshOption (Integer) 1 = Image pane 2 = Index fields pane 3 = Both image and index fields pane
RemoveAllFields	Removes all fields displayed in the Indexing control.	RemoveAllFields()
SaveImage	Saves the modified image in the viewer to disk.	SaveImage()
SetFieldCaption	Changes the caption of a displayed field. **Note**: This does not change the underlying field used to capture data.	SetFieldCaption(FieldName As String, Caption As String) FieldName: The name of the field to receive the new caption. Caption: The value of the new field caption.
SetFieldValue	Sets a value for a specific field.	FieldName (String): The name of the field that needs to be modified. FieldValue (String): The new value for the field.
UpdateThumbnailCaptionSuffix	Updates the Thumbnail caption of the batch page. This only applies when ReviewMode is True.	UpdateThumbnailCaptionSuffix(BatchPage As Object, ByVal bUsePatchCode As Boolean, ByVal sSuffix As String)
ValidateField	Performs a double key validation.	ValidateField(Continue As Boolean, Shift As Boolean)Continue: If set to false, focus will not leave this field. Shift-Indicates whether the Shift key is pressed or not.

3.2.2 DBSearch Object

Description: This object allows a macro to initiate an external database search. A database search must be defined for the indexing profile used.

Properties		
Available	Boolean	Returns whether the external database has been successfully connected, and ready to perform search.
IsFieldMapped (FieldName As String)	Boolean	Returns whether a specific field is mapped in the profile. FieldName is the name of the field to check.
Profile	String	Accepts an edoDBSearch object to use when the search is executed.

Methods		
Search	Performs the search.	FieldName (String): Name of the search field. FieldValue (String): Value to search. If the user wants to do a wild-card search, the % sign must be in the value. ctlIndexing (Object): Reference to the ctlIndexing object, so that the search result will be properly displayed.

3.2.3 IndexingUI Object

Description: Allows user through macro to navigate through batch pages.

Properties	
Current Page	Returns the current page number.

Methods		
PreviousPage	Displays the previous page in the batch.	nRowIndex (Integer): Determines which index field will get the focus when the page is displayed.
NextPage	Displays the next page in the batch.	Parameters same as PreviousPage method.
GoToPage	Displays a specific page in the batch.	nPage (Integer): The batch page to display. nRowIndex (Integer): Determines which index field will get the focus when the page is displayed.
UpdateStatusBar	Displays a custom message within the Capture status bar.	Parameter: Caption as String: The text to display in the status bar.

3.3 Index Events

Choose from the following events.

Event	ApplyValuesToRemainingPages
Description	This event is called just before the remaining pages in the batch are assigned the current page's index values (i.e., user clicked the Apply Values to Remaining Pages button).
Syntax	ecIndex_ApplyValuesToRemainingPages(ByVal CurrentPage As Integer, Cancel As Boolean)
Parameters	*CurrentPage* is the page number of the currently displayed page. *Cancel* set to True will cancel the operation.

Event	ApplyValuesToSeparator
Description	This event is called just before the remaining pages between separators are assigned the current page's index values (i.e., user clicked the Apply Values to Separator button).
Syntax	ecIndex_ApplyValuesToSeparator(ByVal FirstPage As Integer, ByVal CurrentPage As Integer, ByVal LastPage As Integer, Cancel As Boolean)
Parameters	*FirstPage* is the page number of the first page between separators. *CurrentPage* is the page number of the currently displayed page. *LastPage* is the page number of the last page between separators. *Cancel* set to True will cancel the operation.

Event	BatchClose
Description	This event occurs when a user closes a batch.
Syntax	ecIndex_BatchClose()
Parameters	None

Event	BatchDelete
Description	This event is called when a batch is deleted from the batch list.
Syntax	ecIndex_BatchDelete(ByVal BatchName As String, ByVal Pages As Integer)
Parameters	*BatchName* is the name of the batch deleted. *Pages* is the total number of pages in the batch.

Event	BatchesPreDelete
Description	This event is called before batch(es) are deleted from the batch list.
Syntax	ecIndex_BatchesPreDelete(sBatches() As String, Cancel As Boolean)
Parameters	*sBatches()* is an array of batches selected for deletion. *Cancel* set to True will cancel the operation.

Event	BatchNoteDelete
Description	This event is called when the user chose to delete a note from a batch.

Event	BatchNoteDelete
Syntax	ecIndex_BatchNoteDelete(ByVal BatchName As String, ByVal OldNote As String, Cancel As Boolean)
Parameters	*BatchName* is the name of the batch affected. *OldNote* is the current note for the batch. *Cancel* set to True will cancel the operation.

Event	BatchNoteEdit
Description	This event is called when a user adds or changes a batch note.
Syntax	ecIndex_BatchNoteEdit(ByVal BatchName As String, NewNote As String, ByVal OldNote As String, Cancel As Boolean)
Parameters	*BatchName* is the name of the batch affected. *NewNote* is the text of the new note. *OldNote* is the text of the previous note. *Cancel* set to True will cancel the operation.

Event	BatchOpen
Description	This event occurs when a user opens a batch.
Syntax	ecIndex_BatchOpen(Batch As Object, IndexingUI As Object, ctlIndexing As Object)
Parameters	Use the Batch object to access data elements of a batch. *Batch As Object* represents the batch being opened (see "Batch Object" on page 6-10). *IndexingUI As Object* refers to an object which allows the user to move between pages as the end user would (see "IndexingUI Object" on page 3-5). *ctlIndexing As Object* refers to an object which allows the developer to manipulate the fields of the current page (see "ctlIndexing Object" on page 3-1).

Event	BatchPostcommit
Description	This event occurs just after the user commits a batch.
Syntax	ecIndex_PostCommit()
Parameters	None

Event	BatchPrecommit
Description	This event occurs just before the user commits the batch.
Syntax	ecIndex_BatchPrecommit(Cancel As Boolean)
Parameters	*Cancel* set to True will cancel the operation.

Event	BatchPreOpen
Description	This event is called just before a batch is opened.
Syntax	ecIndex_BatchPreOpen(Batch As Object, Cancel As Boolean)
Parameters	*Batch* is the Capture batch object that represents the batch (see "Batch Object" on page 6-10). *Cancel* set to True will cancel the operation.

Event	BatchPriorityEdit
Description	This event occurs when the user modifies the priority of a batch.
Syntax	ecIndex_BatchPriorityEdit(BatchName As String, NewPriority as Integer, OldPriority As Integer, Cancel As Boolean)
Parameters	*BatchName* is the name of the batch affected. *NewPriority* is the new priority number being assigned to the batch. *OldPriority* is the previous priority number of the batch. *Cancel* set to True cancels the priority update for the batch.

Event	BatchSearch
Description	This event occurs when the user searches for batches. Use the Search Criteria object to override the selected batch search criteria.
Syntax	ecIndex_BatchSearch(SearchCriteria As Object, Cancel As Boolean)
Parameters	*SearchCriteria* is the search criteria chosen by the user (see "SearchCriteria Object" on page 6-15). *Cancel* set to True cancels the batch search.

Event	BatchStatusEdit
Description	This event occurs when the user modifies the status of a batch.
Syntax	ecIndex_BatchStatusEdit(BatchName As String, NewStatus As String, OldStatus As String, Cancel as Boolean)
Parameters	*BatchName* is the name of the batch affected. *NewStatus* is the value of the new status being assigned to the batch. *OldStatus* is the previous batch status value. *Cancel* set to True cancels the status update for the batch.

Event	ClearAllIndexes
Description	This event occurs when the user clicks the Clear All button.
Syntax	ecIndex_ClearAllIndexes(Cancel As Boolean)
Parameters	*Cancel* set to True cancels the operation.

Event	DBSearchResult
Description	This event occurs for every field when the database lookup record is selected/returned.
Syntax	ecIndex_DBSearchResult(ByVal FieldName As String, Value As String).
Parameters	*FieldName* is the Capture field that is about to be populated. *Value* is the value to populate the field with. This can be changed by the macro.

Event	DocumentNext
Description	User clicked the next document button.
Syntax	ecIndex_DocumentNext(Cancel As Boolean)
Parameters	*Cancel* set to True cancels the operation.

Event	DocumentNextComplete
Description	Index completed navigating to the next document.
Syntax	ecIndex_DocumentNextComplete(FromPage As Integer, ToPage As Integer)
Parameters	*FromPage* is the page user was on prior to navigating to the next document. *ToPage* is the page displayed after navigating to the next document.

Event	DocumentPostcommit
Description	This event occurs during the batch commit process and gets executed just after each document is committed.
Syntax	ecIndex_DocumentPostCommit(Document As Object)
Parameters	*Document* is an object that contains the document data elements. See "Document Object" on page 6-13.

Event	DocumentPrecommit
Description	This event occurs during the batch commit process and gets executed just before each document is committed.
Syntax	ecIndex_DocumentPreCommit(Document As Object, Cancel As Boolean)
Parameters	*Document* is an object that contains the document data elements. See "Document Object" on page 6-13. *Cancel* set to True will cancel the operation. **Note:** This event works as described when committing directly from Index. It does not affect committing via Commit Server.

Event	DocumentPrevious
Description	User clicked the previous document button.
Syntax	ecIndex_DocumentPrevious(Cancel As Boolean)
Parameters	*Cancel* set to True cancels the operation.

Event	DocumentPreviousComplete
Description	Index completed navigating to the previous document.
Syntax	ecIndex_DocumentPreviousComplete(FromPage As Integer, ToPage As Integer)

Event	DocumentPreviousComplete
Parameters	*FromPage* is the page user was on prior to navigating to the previous document. *ToPage* is the page displayed after navigating to the previous document.

Event	ExternalDBInitiate
Description	This event occurs shortly after a user selects a batch to index and Index connects to an external database.
Syntax	ecIndex_ExternalDBInitiate (DBSearch As Object)
Parameters	*DBSearch* is an object that allows you to obtain search information and execute external database searches from a macro.

Event	ExternalDBTerminate
Description	This event occurs when Index disconnects from an external database.
Syntax	ecIndex_ExternalDBTerminate()
Parameters	None

Event	FieldGetIndexValue
Description	When Index retrieves a saved index value from the database, this event occurs BEFORE the value is displayed to the user. This event must fire even if the index field has no saved value.
Syntax	ecIndex_FieldGetIndexValue (ByVal FieldName As String, DisplayValue As String)
Parameters	*FieldName* is the name of the currently selected field. *DisplayValue* is the current value in the field.

Event	FieldGotFocus
Description	This event occurs when the focus moves to a new index field.
Syntax	ecIndex_FieldGotFocus (FieldName As String, FieldValue As String)
Parameters	*FieldName* is the name of the currently selected field. *FieldValue* is the current value in the field.

Event	FieldKeyDown
Description	This event occurs when the user releases a key on an index field.
Syntax	ecIndex_FieldKeyDown(KeyCode As Integer, Shift As Integer)

Event	FieldKeyDown
Parameters	*KeyCode* is an integer that represents a key press on the keyboard. (See "Keycodes" on page A-1). *Shift* is an integer that corresponds to the state of the SHIFT, CTRL, and ALT keys at the time of the event. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT are pressed, the value of shift is 6.

Event	FieldKeyPress
Description	This event occurs when a user presses a keyboard key.
Syntax	ecIndex_FieldKeyPress(Key Ascii As Integer)
Parameters	*KeyAscii* is an integer that returns a standard numeric ANSI keycode. Keyascii is passed by reference; changing it sends a different character to the index field. Changing *keyascii* to 0 cancels the keystroke so the index field receives no character.

Event	FieldKeyUp
Description	This event occurs when the user releases a key while in an index field.
Syntax	ecIndex_FieldKeyUp(KeyCode As Integer, Shift As Integer)
Parameters	*KeyCode* is an integer that represents a key press on the keyboard. (See "Keycodes" on page A-1). *Shift* is an integer that corresponds to the state of the SHIFT, CTRL, and ALT keys at the time of the event. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT are pressed, the value of shift is 6.

Event	FieldLostFocus
Description	This event occurs when the focus moves from an index field.
Syntax	ecIndex_FieldLostFocus(ByVal FieldName As String, ByVal FieldValue As String, ByVal bLastField As Boolean, Continue As Boolean)
Parameters	*FieldName* is the name of the field that lost the focus. *FieldValue* is the value of the field. *bLastField* is True if the field that lost focus is the last displayed index field. Set *Continue* to True to allow the lost focus to occur. Set it to False to prevent the user from leaving the field.

Event	FieldSaveIndexValue
Description	This event occurs when Index saves a displayed index field value to the Page object for the currently displayed page. It does not occur multiple times during such operations as Apply Values to Remaining Pages or Apply Values to Separator Page.
Syntax	ecIndex_FieldSaveIndexValue(ByVal FieldName As String, DisplayValue As String, FieldValue As String)
Parameters	*FieldName* is the name of the field whose value is being saved. *DisplayValue* is the displayed value of the field. *FieldValue* is the commit value of the field.

Event	FilterBatch
Description	This event occurs when the indexing batch list is refreshed. It fires for every batch matching the batch prefix. This allows a macro to only display specific batches in the list.
Syntax	ecIndex_FilterBatch(ByVal Batch As Object, Cancel As Boolean)
Parameters	*Batch* is a Capture batch object representing the batch that is about to be added to the list (see "Batch Object" on page 6-10). If *Cancel* is set to True, Index will abort the operation.

Event	Find
Description	Event occurs when the user attempts to search the batch.
Syntax	ecIndex_Find(nSearchType As Integer, FieldValue As String, Cancel As Boolean)
Parameters	*nSearchType* specifies the type of search (Search_ UnindexedPage = 1 or Search_IndexValues = 2). *FieldValue* is the index value sought if searching for an index value. *Cancel* set to True will cancel the operation.

Event	FindFound
Description	Event occurs when a page is found after a Find operation.
Syntax	ecIndex_FindFound(ByVal Page As Integer, ByVal FieldName As String, ByVal FieldValue As String, Cancel As Boolean)
Parameters	*Page* is the page number found, *FieldName* is the name of the field that contains the index value (if applicable), *FieldValue* is the value sought. Set *Cancel* to True to prevent the display of the found document and allow Index to continue searching.

Event	GotoPage
Description	Event occurs when the user specifies a page to display.
Syntax	ecIndex_GotoPage(PageNum As Integer, Cancel As Boolean)
Parameters	*PageNumber* is the page number to be displayed in the image viewer. *Cancel* set to True will cancel the operation.

Event	MouseDown
Description	Event occurs when a mouse button is pressed on an image.
Syntax	ecIndex_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
Parameters	*Button* is the button pressed, where: 1=Left button is pressed (vbLeftButton), 2=Right button is pressed (vbRightButton) and 4=Middle button is pressed (vbMiddleButton). *Shift* is the alternate key pressed, where: 1=SHIFT key is pressed (vbShiftMask), 2=CTRL key is pressed (vbCtrlMask) and 4= ALT key is pressed (vbAltMask).

Event	MouseUp
Description	Event occurs when a mouse button is released over an image.
Syntax	ecIndex_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
Parameters	*Button* is the button pressed, where: 1=Left button is pressed (vbLeftButton), 2=Right button is pressed (vbRightButton) and 4=Middle button is pressed (vbMiddleButton). *Shift* is the alternate key pressed, where: 1=SHIFT key is pressed (vbShiftMask), 2=CTRL key is pressed (vbCtrlMask) and 4= ALT key is pressed (vbAltMask).

Event	OCRPostProcess
Description	This event occurs immediately after a field value was OCRed. This event allows you to pre-process the OCR data (Value) before it is placed into the index field.
Syntax	ecIndex_OCRPostProcess(ByVal FieldName As String, Value As String)
Parameters	*FieldName* is the name of the field that is about to receive the OCR data. *Value* is the data that was OCR. This variable can be modified to affect the actual data put into the index field.

Event	OCRPreProcess
Description	This event occurs just before a field is about to be OCRed.
Syntax	ecIndex_OCRPreProcess(Cancel As Boolean)
Parameters	If *Cancel* is set to True, Index will abort the operation.

Event	PageDisplay
Description	This event occurs when a page is displayed.
Syntax	ecIndex_PageDisplay(PageNumber As Integer)
Parameters	*PageNumber* is the page number just displayed in the image viewer.

Event	PageDuplicate
Description	This event occurs when a page(s) is being duplicated.
Syntax	ecIndex_PageDuplicate(PageFrom As Integer, PageTo As Integer, Copies As Integer, Cancel As Boolean)
Parameters	*PageFrom* is the starting page number to be duplicated. *PageTo* is the ending page number to be duplicated. *Copies* is the total number of duplicates. *Cancel* set to True will cancel the operation.

Event	PageGoto
Description	This event occurs when a user goes to a specified page.
Syntax	ecIndex_PageGoto (Cancel As Boolean)
Parameters	If *Cancel* is set to True, Index will abort the operation.

Event	PageNext
Description	This event occurs when a user goes to the next page.
Syntax	ecIndex_PageNext (Cancel As Boolean)
Parameters	If *Cancel* is set to True, Index will abort the operation.

Event	PagePostDelete
Description	This event occurs after pages have been deleted from the batch.
Syntax	ecIndex_PagePostDelete(PagesDeleted() As Integer)
Parameters	*PagesDeleted* is an array of page numbers deleted.

Event	PagePostDuplicate
Description	This event occurs after pages have been duplicated within a batch.
Syntax	ecIndex_PagePostDuplicate()
Parameters	None

Event	PagePreDelete
Description	This event occurs when pages are about to be deleted.
Syntax	ecIndex_PagePreDelete(Cancel As Boolean)
Parameters	*Cancel* set to True will cancel the operation.

Event	PagePreDuplicate
Description	This event occurs just before pages are about to be duplicated within the batch.
Syntax	ecIndex_PagePreDuplicate(Cancel As Boolean)

Event	PagePreDuplicate
Parameters	*Cancel* set to True will cancel the operation.

Event	PagePrevious
Description	This event occurs when a user goes to the previous page.
Syntax	ecIndex_PagePrevious(Cancel As Boolean)
Parameters	If *Cancel* is set to True, Index will abort the operation.

Event	RegionSelected
Description	This event occurs when a region is selected on an image.
Syntax	ecIndex_RegionSelected(ByVal Button As Long, ByVal X1 As Long, ByVal Y1 As Variant, ByVal X2 As Variant, ByVal Y2 As Variant)
Parameters	*Button* is the button pressed, where: 1=Left button is pressed (vbLeftButton), 2=Right button is pressed (vbRightButton) and 4=Middle button is pressed (vbMiddleButton).

Event	RenameBatch
Description	Event occurs when a batch is being renamed.
Syntax	ecIndex_RenameBatch(ByVal Batch As Object, NewBatchName As String, Cancel As Boolean)
Parameters	*Batch* is a Capture batch object representing the batch being renamed (see "Batch Object" on page 6-10). *NewBatchName* is the batch's new name. *Cancel* set to True cancels the renaming.

Event	RenameBatchComplete
Description	Index completed renaming the batch.
Syntax	ecIndex_RenameBatchComplete(ByVal Batch As Object, ByVal OriginalName As String)
Parameters	*Batch* is a Capture batch object representing the batch that was renamed (see "Batch Object" on page 6-10). *OriginalName* is the batch's name before it was renamed. *Cancel* set to True cancels the renaming.

Event	ThumbnailCopyPages
Description	Event occurs when pages are being copied using the thumbnail area.
Syntax	ecIndex_ThumbnailCopyPages(nPages() As Integer, Cancel As Boolean)
Parameters	*nPages* is an array of the pages being copied. *Cancel* set to True will cancel the operation.

Event	ThumbnailDeletePages
Description	Event occurs when pages are being deleted using the thumbnail area.
Syntax	ecIndex_ThumbnailDeletePages(nPages() As Integer, Cancel As Boolean)
Parameters	*nPages* is an array of the pages being deleted. *Cancel* set to True will cancel the operation.

Event	ThumbnailMovePages
Description	Event occurs when pages are being moved within the thumbnail area.
Syntax	ecIndex_ThumbnailMovePages(nPages() As Integer, Cancel As Boolean)
Parameters	*nPages* is an array of the pages being moved. *Cancel* set to True will cancel the operation.

4

Recognition Server Macros

This section covers the following topics:

4.1 About Recognition Server Macros

Below are common uses for Recognition Server macros:

- Split a single bar code value into multiple field values.
- Assign bar code value(s) to proper fields.
- Update corporate databases using bar code values read.
- Use custom logic to determine which pages constitute document separation.
- Perform custom auditing of server activity.
- Send custom email alerts.
- Cancel the committing of a batch due to invalid data.

For a list of Recognition Server events, see "Recognition Server Events" on page 4-4.

4.1.1 Batch Job Processing Order of Events

Below is the order in which events are executed when processing a batch job:

1. StartBatchJob
2. StartBatch
3. StartBatchPage
4. ValidateBarCodes (only if bar codes were recognized)
5. IsDocSeparator
6. StartDBSearch (only if external database searching is specified for the batch job)
7. SaveFieldValues
8. EndBatchPage
9. UpdateBatch
10. BatchPreCommit

11. ErrorCommitBatch (only if an error occurs during a document commit)
12. BatchPostCommit
13. EndBatch
14. EndBatchJob

4.2 Recognition Server Objects

This section describes recognition server objects.

4.2.1 Audit Object

Description: This object is used to perform auditing. It can be used to log data to the batch job's log file, add an audit record to the Capture database, or send an email message.

Methods:

Sub AuditToDB(lActionID as Long, sActionDesc As String, sFileCabinet as String, sBatch as String, lActionDataInt As Long, fActionDataFloat As String, sActionDataText1 as String, sActionDataText2 As String, sActionDataText3 As String, sActionDataText4 As String, sActionDataText5 As String)

Parameters:

- lActionID: Unique number assigned to the action being audited.
- sActionDesc: Description of the action being audited.
- sFileCabinet: Name of the batch's parent FileCabinet.
- sBatch: Name of the batch currently being processed.
- ActionDataInt: Optional number value to audit.
- fActionDataFloat: Optional real number value to audit.
- sActionDataText1…5: Optional text field data.

Sub EmailNotify(sProfile As String, sTo As String, sCC As String, sSubject As String, sBody as String)

Parameters:

- sProfile: MAPI profile to use
- sTo: Recipient's email address
- sCC: Carbon copy email address
- sSubject: Subject for the email message
- sBody: Content of the email message

Sub LogToFile(lEventId as Long, nEventType as EventStatus, sEventDesc As String)

Parameters:

- lEventId: Number identifying the event.
- nEventType: Must be one of the following event types (Critical = 1, Warning = 2, Information = 4)
- sEventDesc: Data to log to file.

4.2.2 BarCode Object

Description: This object represents a bar code that was read during recognition of an image.

Properties		
Symbology	String	Symbology of the bar code (e.g., Code 39, 128, etc.).
Valid	Boolean	Indicates whether or not the bar code is considered valid.
Value	String	Value of the bar code.

4.2.3 BarCodesRead Object

Description: The object is a collection of BarCode objects that were created during bar code recognition for the current image.

Properties		
Count	Integer	Number of BarCode objects in the collection.
Item (Index)	Object	Reference to a BarCode object.

4.2.4 BatchJobBarCode Object

Description: This object represents a barcode defined for a batch job.

Properties		
Name	String	Name of the bar code defined.
ValidationRule	ValidationRuleData	Must be one of the following (vrNone = 0, vrLength = 1, vrMask = 2, vrPickList = 3).
BarCodeParameter	String	The value of the Mask or Length if ValidationRule is vrLength or vrMask respectively.
BarCodeParameters	String()	The array of pick list items if the ValidationRule is vrPickList.
Value	String	Value of the bar code.

4.2.5 BatchJobBarCodes Object

Description: This object is a collection of BatchJobBarcode objects.

Properties		
Count	Integer	Number of bar codes defined for the batch job.
Item (Index)	Object	Reference to a BatchJobBarCode object.

4.2.6 IndexField Object

Description: The IndexField object represents a field associated with a File Cabinet.

Properties		
Name	String	Name of the field.
Value	String	Value of the field.

4.2.7 IndexFields Object

Description: The IndexFields object is a collection of all document index fields. The index fields available are determined based upon the File Cabinet selected for the batch job.

Properties		
Count	Integer	Number of IndexField objects in the collection.
Item (Index)	Object	Reference to an IndexField object.

4.3 Recognition Server Events

Choose from the following events.

Event	BatchPostCommit
Description	This event occurs immediately after Recognition Server commits a batch.
Syntax	ecAutoFile_BatchPostCommit()
Parameters	None

Event	BatchPreCommit
Description	This event occurs just before Recognition Server attempts to commit a batch. If desired, this event can cause Recognition Server to abort the batch commit process.
Syntax	ecAutoFile_BatchPreCommit(Cancel As Boolean, RenameBatch As Boolean)
Parameters	*Cancel* set to True will cause Recognition Server to abort the batch commit process. *RenameBatch* if set to True will cause the batch to get renamed if pages remain after processing.

Event	CommitProfileBegin
Description	Event occurs just prior to the batch being committed by a commit profile.
Syntax	ecAutoFile_CommitProfileBegin(ByVal CommitProfile As Object)
Parameters	*CommitProfile* refers to a File Cabinet Commit Profile (see "CommitProfile Object" on page 6-12.)

Event	CommitProfileEnd
Description	Event occurs after a batch has been committed by a commit profile.
Syntax	ecAutoFile_CommitProfileEnd(ByVal CommitProfile As Object)

Event	CommitProfileEnd
Parameters	*CommitProfile* refers to a File Cabinet Commit Profile (see "CommitProfile Object" on page 6-12.)

Event	DocumentCommitCanceled
Description	Event occurs if a document commit was canceled or aborted due to an error.
Syntax	ecAutoFile_DocumentCommitCanceled(ByVal Document As Object)
Parameters	*Document* refers to the document being committed (see "Document Object" on page 6-13).

Event	DocumentPostcommit
Description	Event occurs after a document has been committed.
Syntax	ecAutoFile_DocumentPostcommit(Document As Object)
Parameters	*Document* refers to the document being committed (see "Document Object" on page 6-13).

Event	DocumentPrecommit
Description	Event occurs just before a document is about to be committed.
Syntax	ecAutoFile_DocumentPrecommit(Document As Object, Cancel As Boolean)
Parameters	*Document* refers to the document being committed (see "Document Object" on page 6-13). *Cancel* set to True will cancel the operation.

Event	EndBatch
Description	This event occurs when Recognition Server has finished processing a batch. It will occur regardless of whether or not the batch was successfully processed (i.e., committed).
Syntax	ecAutoFile_EndBatch()
Parameters	None

Event	EndBatchJob
Description	This event occurs when Recognition Server has finished processing a batch job. It will occur even if there were no batches processed for the batch job.
Syntax	ecAutoFile_EndBatchJob()
Parameters	None

Event	EndBatchPage
Description	This event occurs when Recognition Server has finished processing a batch page (i.e., image).
Syntax	ecAutoFile_EndBatchPage()

Event	EndBatchPage
Parameters	None

Event	ErrorCommitBatch
Description	This event is generated when an error occurs during a document commit.
Syntax	ecAutoFile_ErrorCommitBatch(Document As Object, RenameBatch As Boolean, ByVal ErrorNumber as Long, ByVal ErrorDescription as String)
Parameters	*Document:* The current document object being processed (see "Document Object" on page 6-13). *RenameBatch*Determines whether or not the batch will be renamed when the batch is in error. *ErrorNumber:* Number of the error being reported. *ErrorDescription:* Description of the error being reported.

Event	IsDocSeparator
Description	This event occurs when the Document Processing Method of a batch job is set to *Multiple pages per document*. This event allows the macro to indicate whether or not the current page is a document separator. This event is fired for every image within the batch.
Syntax	ecAutoFile_IsDocSeparator(SeparatorPage As Boolean, IncludeInDocument As Boolean)
Parameters	*SeparatorPage:* To indicate that the current page is a document separator, set this parameter to True. *IncludeInDocument:* If the current page/image is a separator page, this parameter indicates whether or not it should be kept in the document. To include the separator page, set this parameter to True; otherwise, set it to False.

Event	SaveFieldValues
Description	This event occurs when Recognition Server saves the field values for a document. This event allows a macro to change the index values. This event gets fired for each document. For OneDocPerPage, it fires for each page. For OneDocPerBatch, it fires once at the end of the batch. It also fires when IsDocSeparator returns true or Recognition Server detects a separator page.
Syntax	ecAutoFile_SaveFieldValues(IndexFields As Object)
Parameters	*IndexFields:* See "IndexFields Object" on page 4-4.

Event	StartBatch
Description	This event occurs when Recognition Server begins processing a batch. It is recommended to keep a reference to the Batch object in a module level variable so that it can be used in later events.
Syntax	ecAutoFile_StartBatch(Batch As Object, Cancel As Boolean)
Parameters	*Batch:* See "Batch Object" on page 6-10. *Cancel:* Set to True to abort processing of the batch.

Event	StartBatchJob
Description	This event occurs when Recognition Server begins to process a batch job (i.e., a scheduled batch job event occurs). It is recommended to keep a reference to the oAudit object in a module level variable so that it can be used in later events.
Syntax	ecAutoFile_StartBatchJob(ByVal sBatchJob As String, Audit As Object, Cancel As Boolean)
Parameters	*sBatchJob:* Name of the batch job being processed. *Audit:* See "Audit Object" on page 4-2. *Cancel:* Set to True to abort processing of the batch job.

Event	StartBatchPage
Description	This event occurs when Recognition Server starts processing a page (image).
Syntax	ecAutoFile_StartBatchPage(BatchPage As Object)
Parameters	*BatchPage:* See "BatchPage Object" on page 6-11.

Event	StartDBSearch
Description	This event occurs when Recognition Server is about to perform a database lookup if configured for the batch job being processed. This event allows you to modify the value being sought or cancel the database lookup.
Syntax	ecAutoFile_StartDBSearch(ByVal FieldName As String, FieldValue As String, Cancel As Boolean)
Parameters	*FieldName:* Name of the database field being searched. *FieldValue:* Value to search for. *Cancel:* Set to True to abort the database lookup.

Event	UpdateBatch
Description	This event will occur if pages remain after a batch has been processed. It allows you to modify the batch status, priority, or note. For example, if the macro aborted the processing of the batch, you may want to record the reason it was aborted in the note field. This event is called right before the commit. Therefore, if a macro aborts the processing of a batch, this event will not get fired. If the status, priority, or note is changed in this event, the values will get carried forward to an error batch. There are three places where a macro can abort. **1.** StartBatchJob **2.** StartBatch **3.** BatchPreCommit Only the BatchPreCommit call is fired after the UpdateBatch call. So for 1 and 2, UpdateBatch is not fired. For 3, it will be fired.
Syntax	ecAutoFile_UpdateBatch(Status As String, Priority As Integer, Note As String)

Event	UpdateBatch
Parameters	*Status:* Batch status *Priority:* Batch priority *Note:* Batch note

Event	ValidateBarCodes
Description	This event occurs when Recognition Server has performed bar code recognition and found bar codes on the current page. The event allows a macro to assign bar code values to the correct bar code fields defined for the batch job. In addition, a macro can indicate which bar codes are valid. Invalid bar code values are recorded in the log file for the batch job.
Syntax	ecAutoFile_ValidateBarCodes(BatchJobBarCodes As Object, BarCodesRead As Object)
Parameters	*BatchJobBarCodes:* See "BatchJobBarCodes Object" on page 4-3. *BarCodesRead:* See "BarCodesRead Object" on page 4-3.

5

Import Server Macros

This section covers the following topics:

5.1 About Import Server Macros

Out of the box, the Import Server provides many features and functionality. However, if you need to customize the importing process, the Import Server offers a variety of customization methods through different types of macros.

The Import Server provides the following macro types:

Macro type	Description
Import Server	An Import Server macro can be applied to any batch job, regardless of the associated import provider.
Folder/List File Provider	This macro allows you to control some of the internal functionality of the Folder/List File import provider.
Custom Provider	This macro allows you to easily build a custom import provider for Import Server, in case none of the import providers yields the functionality you need.
Castelle FaxPress Provider	This macro allows you to control some of the internal functionality of the Castelle FaxPress import provider.
Email Provider	This macro allows you to control some of the internal functionality of the Email provider.

You can develop macros for Import Server to perform a wide variety of functions. Some ideas include:

- Skip the importing of certain image files
- Change Capture batch properties
- Skip the importing of a batch

- Add page level index values during importing
- After importing, move images to a different folder

5.1.1 Batch Job Processing Order of Events

The diagrams that follow illustrate the order in which Import Server macro functions execute:

Figure 5–1 Process Job

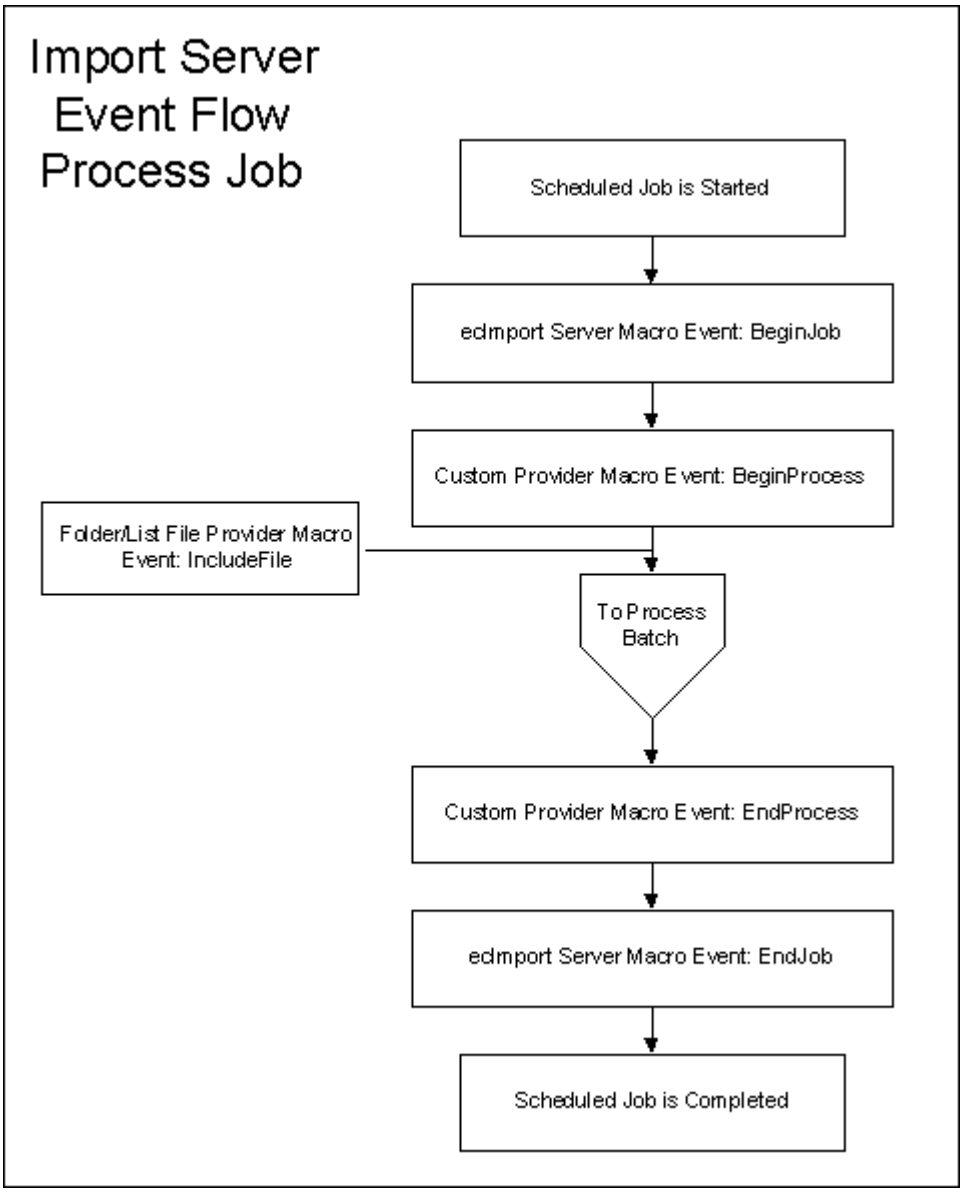


Figure 5–2 Process Batch

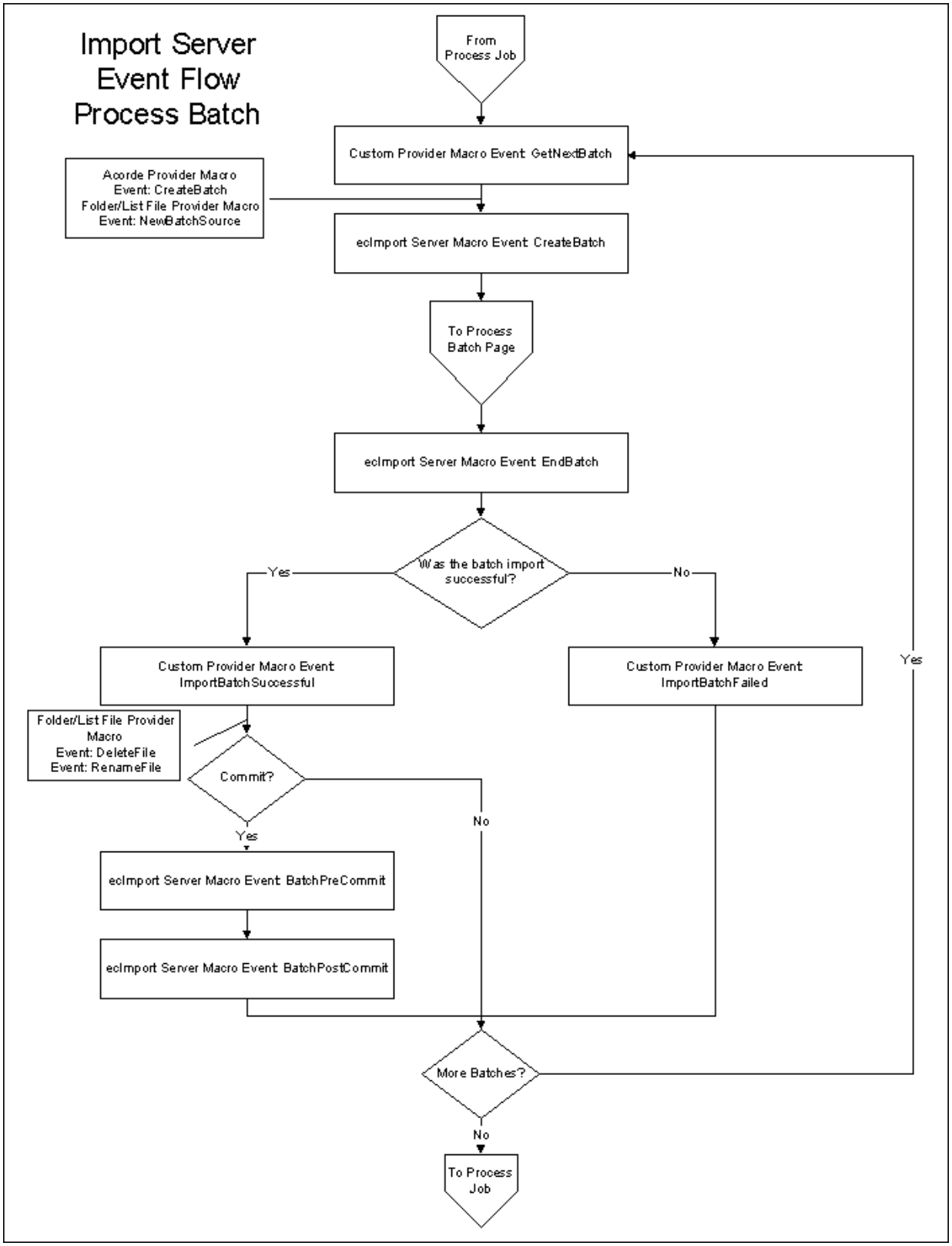
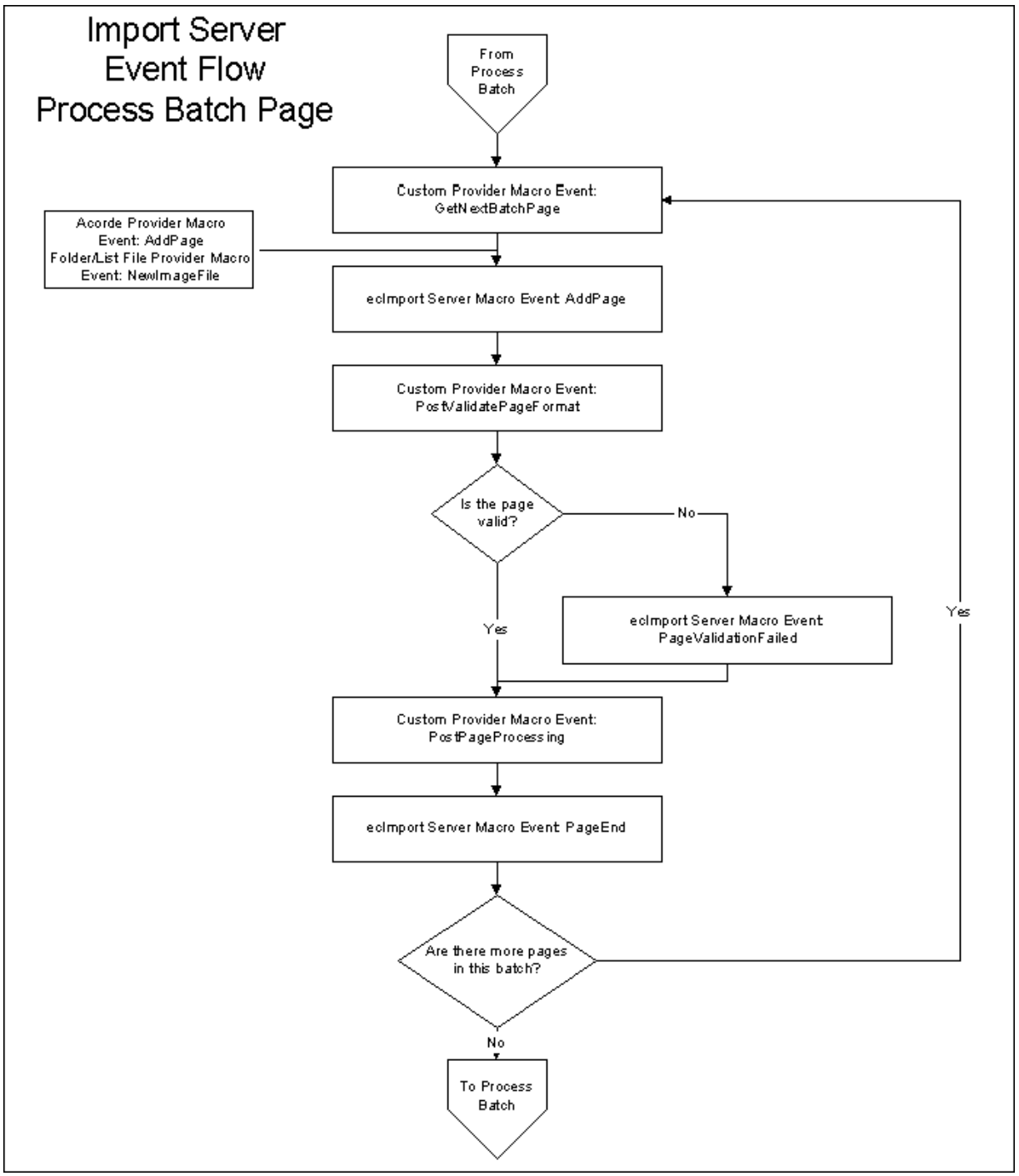


Figure 5–3 Process Batch Page



5.2 Import Server Events

Choose from the following events.

Event	AddPage
Description	The AddPage event occurs after a batch page has been retrieved from a provider but before the image is validated and added to the Capture batch.

Event	AddPage
Syntax	ecImport_AddPage(ispBatchPage As Object, Cancel As Boolean)
Parameters	*ispBatchPage* represents an Import Server Provider batch page object (see "ispBatchPage Object" on page 5-15). *Cancel* set to true will cancel the importing of the page. **Note**: Whether the page would be skipped or the batch would be aborted depends on the settings for the job on the General Settings tab.

Event	AllowTextExtraction
Description	If a macro enables text extraction, Import Server will extract the ASCII text data generated during the virtual printing process for non-image files, which will cause the ProcessTextPage event to occur.
Syntax	ecImport_AllowTextExtraction(bAllowTextExtraction As Boolean)
Parameters	Set *bAllowTextExtraction* to True if a macro desires to obtain the text data during the virtual printing process of non-image files.

Event	BatchPostCommit
Description	The BatchPostCommit event occurs just after Import Server commits a batch (whether the user selected to commit directly or through Commit Server).
Syntax	ecImport_BatchPostCommit()
Parameters	None

Event	BatchPreCommit
Description	The BatchPreCommit event occurs just before Import Server attempts to commit a batch. If desired, this event can cause Import Server to abort the batch commit process.
Syntax	ecImport_BatchPreCommit(Cancel As Boolean)
Parameters	*Cancel* set to True will cause Import Server to abort the batch commit process.

Event	BeginFile
Description	This event is called when an image or non-image file is about to be processed.
Syntax	ecImport_BeginFile(ispBatchPage As Object, Cancel As Boolean)
Parameters	*ispBatchPage* represents an Import Server Provider batch page object (see "ispBatchPage Object" on page 5-15). *Cancel* set to true will cause the image or non-image file to not be processed.

Event	BeginJob
Description	The BeginJob event occurs when processing of an Import Server job begins. This will occur before the provider begins processing.

Event	BeginJob
Syntax	ecImport_BeginJob(isJob As Object, isAudit As Object, Cancel As Boolean)
Parameters	*isJob* represents the job that is processing (see "ispJob Object" on page 5-14). *isAudit* represents the Import Server Audit object (see "ispAudit Object" on page 5-14). *Cancel* set to true will cause Import Server to abort processing of the job.

Event	CommitProfileBegin
Description	This event occurs just before a commit profile is executed.
Syntax	ecImport_CommitProfileBegin(ByVal CommitProfile As Object)
Parameters	*CommitProfile* refers to a File Cabinet Commit Profile (see "CommitProfile Object" on page 6-12.)

Event	CommitProfileEnd
Description	This event occurs just after a commit profile is executed.
Syntax	ecImport_CommitProfileEnd(ByVal CommitProfile As Object)
Parameters	*CommitProfile* refers to a File Cabinet Commit Profile (see "CommitProfile Object" on page 6-12.)

Event	ConvertNonImage
Description	This event is called only when a NON-image file is about to be processed.
Syntax	ecImport_ConvertNonImage(ispBatchPage As Object, Cancel As Boolean)
Parameters	*ispBatchPage* represents an Import Server Provider batch page object (see "ispBatchPage Object" on page 5-15). *Cancel* set to true will cause the non-image file to not be processed.

Event	CreateBatch
Description	The CreateBatch event occurs just after a new batch is created in Capture to receive the imported pages.
Syntax	ecImport_CreateBatch(edoBatch As Object, Cancel As Boolean)
Parameters	*edoBatch* represents a Capture batch object (see "ispBatchPage Object" on page 5-15). *Cancel* set to true will cause Import Server to cancel the batch being imported.

Event	DatabasePostLookup
Description	This event occurs just after performing a database search.
Syntax	ecImport_DatabasePostLookup(Page as Object, Error As Boolean)

Event	DatabasePostLookup
Parameters	*Page* represents the page being imported. *Error* set to True cancels the database lookup in the event of an error.

Event	DatabasePreLookup
Description	This event occurs just prior to performing a database search.
Syntax	ecImport_DatabasePreLookup(Page as Object, ValueSought As String, Cancel As Boolean)
Parameters	*Page* represents the page being imported. *ValueSought* is the value to be matched in the database table. *Cancel* set to True cancels the database lookup.

Event	DocumentPostcommit
Description	This event occurs during the batch commit process and gets executed just after each document is committed.
Syntax	ecImport_DocumentPostcommit(Document As Object)
Parameters	*Document* is an object that contains the document data elements.

Event	DocumentPrecommit
Description	This event occurs during the batch commit process and gets executed just before each document is committed.
Syntax	ecImport_DocumentPrecommit(Document As Object, Cancel As Boolean)
Parameters	*Document* is an object that contains the document data elements. Cancel set to True will cancel the operation.

Event	EndBatch
Description	The EndBatch event occurs after all pages have been imported into the new Capture batch.
Syntax	ecImport_EndBatch(Success As Boolean, Abort As Boolean)
Parameters	*Success* indicates if the batch was successfully imported. *Abort* set to true, the new batch will be aborted and removed.

Event	EndFile
Description	This event corresponds to the BeginFile event and is called after the file has been processed.
Syntax	ecImport_EndFile()
Parameters	None

Event	EndJob
Description	The EndJob event occurs when processing of the current job has completed.

Event	EndJob
Syntax	ecImport_EndJob()
Parameters	None

Event	EndPage
Description	The EndPage event occurs after a batch page has been processed completely, regardless of whether or not the page is valid.
Syntax	ecImport_EndPage()
Parameters	None

Event	ErrorCommitBatch
Description	This event occurs when a document fails to be committed.
Syntax	ecImport_ErrorCommitBatch(Document As Object, ByVal ErrorNumber As Long, ByVal ErrorDescription As String)
Parameters	*Document* is an object that contains the document data elements. *ErrorNumber* is a number that represents the error. *ErrorDescription* is the error description. When a commit error occurs, the document will remain in the batch.

Event	OnPageCanceled
Description	This is called when BeginFile, ConvertNonImage, Or AddPage has set Cancel to True.
Syntax	ecImport_OnPageCanceled(CancelAction As ecImportVBA.eCancelAction)
Parameters	CancelAction: eCancelAction is one of the following values: caUseBatchJobSetting = 0 (Default) caSkip = 1 caAbort = 2 Setting *CancelAction* to caUseBatchJobSetting uses the value from the batch job's setting to determine whether to skip the page or abort the batch. *caSkip* will skip the processing of the page and remove the file. *caAbort* will abort the processing of the batch and leave all files intact.

Event	PageValidationFailed
Description	The PageValidationFailed event occurs when an invalid batch page is detected.
Syntax	ecImport_PageValidationFailed()
Parameters	None

Event	ProcessTextPage
Description	This event allows a macro access to the ASCII print stream data for non-image files, which is generated during the virtual printing process. This capability allows a macro to search for key index data within the text file and automatically assign index values to a document.
Syntax	ecImport_ProcessTextPage(ispBatchPage As Object, oTextPage As Object, Cancel As Boolean)
Parameters	*ispBatchPage* represents an Import Server Provider batch page object (see "ispBatchPage Object" on page 5-15). See "oTextPage Object" on page 5-20. *Cancel* set to True cancels the operation.

5.3 Castelle FaxPress Provider Events

Choose from the following events.

Event	AddPage
Description	Occurs before a page is added to a Capture batch.
Syntax	FaxPress_AddPage(ispBatchPage As ISP.ispBatchPage, ByRef bCancel As Boolean)
Parameters	*ispBatchPage:* Information about the page to be addedbCancel-Allows the programmer to cancel the adding of a page

Event	CreateBatch
Description	Occurs before the Capture batch is created.
Syntax	FaxPress_CreateBatch(oFaxPressJob As FaxPressJob, ByRef bCancel As Boolean)
Parameters	*oFaxPressJob:* Information about the FaxPress job being used to create the batch. *bCancel:* Allows the programmer to cancel the creation of the batch.

Event	RemoveFaxJob
Description	Occurs before an imported fax job is removed from the FaxPress system.
Syntax	FaxPress_RemoveFaxJob(oFaxPressJob As FaxPressJob, ByRef bCancel As Boolean)
Parameters	*oFaxPressJob:* Information about the FaxPress job about to be removed *bCancel:* Allows the programmer to cancel the removal of the fax job.

5.4 Folder/List File Provider Events

Choose from the following events.

Event	DeleteFile
Description	The DeleteFile event occurs whenever a file is about to be deleted as part of post-processing of a batch. If importing from a folder and the Delete Image Files option was selected or if importing from a list file and the Delete Image Files box was checked, this event will occur for every image file that was successfully imported into a Capture batch. In addition, this event will occur if importing from a list file and the Delete List file option was selected.
Syntax	FolderListFile_DeleteFile(FileName as String, Cancel As Boolean)
Parameters	*FileName:* File name about to be deleted. *Cancel:* Set to true to cancel the file deletion.

Event	IncludeFile
Description	The IncludeFile event occurs when the Folder / List File Provider is compiling a list of list files (if importing from a list file) or image files (if importing from a folder) in the current import folder. The current import folder will be searched for filenames that match the import file mask and this event will be fired for each file whose name matches that mask.
Syntax	FolderListFile_IncludeFile(FileName as String, Cancel As Boolean)
Parameters	*FileName:* Name of the file to import. *Cancel:* If set to true, Import Server will exclude the file from importing.

Event	NewBatchSource
Description	The NewBatchSource event occurs when a new batch is about to be created because the current list file is about to be changed (if importing from a list file) or because the current import folder is about to change (if importing from a folder).
Syntax	FolderListFile_NewBatchSource(SourceType as Integer, SourceValue as String, Cancel As Boolean)
Parameters	*SourceType:* This parameter will contain either a 0 (meaning the folder is changing) or a 1 (meaning the list file is changing). *SourceValue:* If SourceType is 0, this parameter will contain the new import folder name. If the SourceType is 1, this parameter will contain the name of the new list file. *Cancel:* Set to true will cause the Folder / List File provider to cancel processing of the new batch source.

Event	NewImageFile
Description	The NewImageFile event occurs when an image file is about to be processed.
Syntax	FolderListFile_NewImageFile(FileName as String, Cancel As Boolean)
Parameters	*FileName:* Image file name to process. *Cancel:* Set to true will cause this image file to be excluded from processing. **Note**: This event will only be fired once for a multiple page TIFF file.

Event	RenameFile
Description	The RenameFile event will occur whenever an image file's or list file's name is about to be changed during post-processing of a batch. This includes renaming a file by changing the file's extension or by adding a prefix in front of the filename.
Syntax	FolderListFile_RenameFile(OrigName as String, NewName as String, Cancel As Boolean)
Parameters	*OrigName:* Name of the original file. *NewName:* Name of the new file. This parameter may be modified by the macro (e.g. the macro could move the original file to another directory by modifying the NewName parameter.) *Cancel*: Set to true will cancel the file rename.

5.5 Custom Provider Events

The Import Server Custom Provider is available as a shell to create new import providers. This may be required if the File/List File provider does not provide the desired functionality.

Choose from the following events.

Event	BeginProcess
Description	The BeginProcess event occurs when a batch job is executed.
Syntax	Custom_BeginProcess(Cancel As Boolean)
Parameters	*Cancel:* Set to true will cancel the current job.

Event	EndProcess
Description	The EndProcess event occurs when the current batch job has finished processing.
Syntax	Custom_EndProcess()
Parameters	None

Event	GetNextBatch
Description	The GetNextBatch event occurs when the server is requesting a new batch to process.
Syntax	Custom_GetNextBatch(ispBatch As Object)
Parameters	*ispBatch:* Represents an Import Server batch object (see "ispBatch Object" on page 5-15). The properties of this object must be populated with information about the current batch. At a minimum, the Name property needs to be populated (at least one character must not be a space). If there are no more batches to process, populate the Name property of the ispBatch object with an empty string.

Event	GetNextBatchPage
Description	The GetNextBatchPage event occurs when the server is requesting the next batch page.

Event	GetNextBatchPage
Syntax	Custom_GetNextBatchPage(ispBatchPage As Object, bLastPage As Boolean)
Parameters	*ispBatchPage:* Represents an Import Server batch page object (see "ispBatchPage Object" on page 5-15). This object must be populated with the data required to import the current page. At a minimum, the FileName property of the ispBatchPage object must be populated with the path of the image file. *bLastPage:* Set to true if this is the last page to process for the batch. Alternately, if an empty string is passed back in the FileName property of the ispBatchPage object, the server will assume there are no more pages.

Event	ImportBatchFailed
Description	The ImportBatchFailed event occurs if the batch import failed for any reason.
Syntax	Custom_ImportBatchFailed()
Parameters	None

Event	ImportBatchSuccessful
Description	The ImportBatchSuccessful event occurs when the batch has successfully been imported into Capture.
Syntax	Custom_ImportBatchSuccessful()
Parameters	None

Event	PostPageProcessing
Description	The PostPageProcessing event occurs when the server has completed processing the current batch page.
Syntax	Custom_PostPageProcessing(ispBatchPage As Object)
Parameters	*ispBatchPage:* Represents the batch page just imported (see "ispBatchPage Object" on page 5-15).

Event	PostValidatePageFormat
Description	The PostValidatePageFormat event occurs after the server performs validation on an image file.
Syntax	Custom_PostValidatePageFormat(ispBatchPage As Object, bValid As Boolean)
Parameters	*ispBatchPage:* Represents the current page being imported (see "ispBatchPage Object" on page 5-15). You can use the properties of this object to determine why the page failed validation. *bValid:* If True, the page is valid. If False, the page failed validation.

5.6 Email Provider Events

Choose from the following events.

Event	AddPage
Description	This event occurs just before the page is sent to the server to validate the image and add it to a batch.
Syntax	Email_AddPage(ispBatchPage As Object, Cancel As Boolean)
Parameters	*ispBatchPage*: ispBatchPage object of the page about to be added. See "ispBatchPage Object" on page 5-15. Set *Cancel* to true to cancel the batch page addition.

Event	CreateBatch
Description	This event occurs just before a new batch is created.
Syntax	Email_CreateBatch(ispBatch As Object, EmailMsg As Object, Cancel As Boolean)
Parameters	*ispBatch*: ispBatch object of the batch about to be created. See "ispBatch Object" on page 5-15. *EmailMsg:* EmailMsg object containing information being used to create the batch. See "EmailMsg Object" on page 5-19. Set *Cancel* to true to cancel the batch creation.

Event	DeleteMessage
Description	This event occurs just before an email message is deleted.
Syntax	Email_DeleteMessage(EmailMsg As Object, Cancel As Boolean)
Parameters	*EmailMsg:* EmailMsg object containing information about the email to be deleted. See "EmailMsg Object" on page 5-19. Set *Cancel* to true to cancel deletion of the email message.

Event	ForwardMessage
Description	This event occurs just before an email is forwarded.
Syntax	Email_ForwardMessage(EmailMsg As Object, ByRef ForwardList As String, Cancel As Boolean)
Parameters	*EmailMsg*: EmailMsg object containing information about the email to be forwarded. See "EmailMsg Object" on page 5-19. *ForwardList:* Semicolon-delimited list of email addresses to which to forward the message. Set *Cancel* to true to cancel the email message forwarding.

5.7 Import Server Objects

This section describes import server objects.

5.7.1 ispAudit Object

Description: This object is used to allow VBA-compatible events to hook into the Import Server auditing capabilities.

Method	Parameters	Description
LogToFile	EventID: long EventStatus EventDesc: string	This method will log an event to the current job's log file. The event will also be displayed in the event list on Import Server's log tab. EventStatus: A flag indicating how serious this event is, where: 1 = Critical 2 = Warning 4 = Information EventID: Number identifying the event. EventDesc: A description of the event
AuditToDB	ActionID: long ActionDesc: string FileCabinet: string BatchID: string Batch: string ActionDataInt: integer ActionDataFloat: single ActionDataText1: string ActionDataText2: string ActionDataText3: string ActionDataText4: string ActionDataText5: string	This method will allow you to enter information into the audit table in Capture. ■ ActionID: Unique number assigned to the action being audited. ■ ActionDesc: Description of the action being audited. ■ FileCabinet: The file cabinet this action pertains to. ■ BatchID: The batch's internal identifier assigned by Capture. ■ Batch: The name of the batch this action pertains to. ■ ActionDataInt: Optional number value to audit. ■ ActionDataFloat: Optional real number value to audit. ■ ActionDataText1-ActionDataText5: A string of data associated with this action (255 char max.)
EmailNotify	Profile: string To: string CC: string Subject: string Body: string	This method will allow you to send an email notification through a MAPI client. ■ Profile: MAPI profile to use. ■ To: A list of email addresses to send the message to. ■ Multiple email addresses should be separated by ';'. ■ CC: A list of email addresses to carbon copy the message to. ■ Multiple email addresses should be separated by ';'. ■ Subject: Subject of the email message. ■ Body: Content of the email message.

5.7.2 ispJob Object

Description: This object represents an Import Server job.

Properties	Type	Description
Name	String	The name of the job.
ParentProvider	String	The name of the import provider for this job.
FileCabinet	String	The name of the file cabinet this job will import into.

Properties	Type	Description
Prefix	String	The batch prefix that will be used when creating new batches.
DefaultStatus	String	The default status that will be assigned to a batch.
DefaultPriority	Integer	The default priority (0-10) that will be assigned to a batch.
ServerMacro	String	The name of the server macro assigned to this job.
ProviderMacro	String	The name of the import provider macro assigned to this job.
LogFile	String	The path to the log file for this job.
InvalidSkip	Integer	0: Skip batch when a page fails validation. 1: Skip page if it fails validation.
MaxBatchCount	Integer	The maximum number of batches to import when this job is run.
CommitBatch	Boolean	Indicates if batches should be committed after importing.
UseCommitServer	Boolean	Indicates if the Commit Server should be used when committing a batch.
CreateTime	String	The date/time of when the job was created.
CreateUser	String	The user that created the job.
LastModifyTime	String	The date/time that the job was last modified.
LastModifyUser	String	The user that last modified the job.
Settings	Object	A property bag containing the import provider settings for this job. The properties contained in this object are specific to the import provider related to the job.

5.7.3 ispBatch Object

Description: The ispBatch object represents a batch being imported. This object is populated by an Import Server provider.

Properties	Type	Description
Name	String	Name of the batch. This name does not have any relation to the name of the Capture batch that is created. It must be filled in with data to identify that a batch has been found by the provider.
CreateTime	Date	The date and time that will be passed to the new Capture batch as the creation date of the batch.
Comment	String	The batch comment that will be added to the new Capture batch.
Status	String	The status that will be assigned to the new Capture batch. If this property is not specified, the default status selected for the Job on the general settings tab will be used.
Priority	Integer	The priority that will be assigned to the new Capture batch. If this property is not specified, the default priority selected for the Job on the general settings tab will be used.

5.7.4 ispBatchPage Object

Description: The ispBatchPage object represents a batch page (i.e., image) being imported. This object is populated by an Import Server provider.

Properties	Type	Description
FileName	String	Fully qualified path to the image file being imported.

Properties	Type	Description
IsTempFile	Boolean	Indicates if the FileName property represents a temporary file.
MultiPageNumber	Integer	If the image being imported is a multiple page TIFF file, this property indicates the page number in the file being imported. In all other cases, this value is 1.
ParentFile	String	If the image being imported is from a multiple page TIFF file, the ParentFile property will contain the path to the source multiple page TIFF file, and the FileName property will contain the path to a temporary file containing the current image.
Delete	Boolean	A flag indicating if the current page was successfully imported and can be deleted / renamed when the processing of the current batch is complete.
ErrDesc	String	A description of why the page was not imported.
IncludeInDocument	Boolean	Indicates whether or not the page will be filed.
PageComplete	Boolean	Indicates whether or note the page has correct index information so that it can be committed.
BarcodeCount	Integer	The number of barcodes in the Barcodes property.
Barcodes	String Array	A string array of barcode values for the page.
PatchCode	Long	The patch code value associated with the page, where: No patch code present=0 PATCHVALUE_I=8 PATCHVALUE_II=2 PATCHVALUEIII=1 PATCHVALUEIV=16 PATCHVALUEVI=32 PATCHVALUE_T=4
Indexes	Object	Collection of ispIndex objects (see "ispIndex Object" on page 5-18). Contains index values for this page, which are listed in "Indexes object" on page 5-16.

5.7.4.1 Indexes object

Property	Count
Description	The number of indexes defined in the collection.
Return Type	Long
Parameters	None

Property	Item
Description	This returns an object reference to the index object in the collection indicated by the Key. The Key can be the fieldname or a numeric index into the collection (starting with 1).
Return Type	ispIndex object
Parameters	Key As Variant

Method	AddWithValue
Description	This adds an ispIndex object to the collection with a Key of the fieldname and a value of the FieldValue.
Return Type	None
Parameters	FieldName as String, FieldValue as String

Method	Clear
Description	Clears out the indexing collection.
Return Type	None
Parameters	None

Method	ClearValue
Description	Keeps all the indexing objects in the collection but resets all the values to an empty string.
Return Type	None
Parameters	None

Method	Remove
Description	Removes the object reference to the index object indicated by the Key from the collection. The Key can be the fieldname or a numeric index into the collection (starting with 1).
Return Type	None
Parameters	Key as Variant

5.7.5 ispFolderListSettings Property Bag

Description: This property bag represents the Folder / List File provider settings for a specific job. The provider specific settings are stored in the setting property of the isJob object.

Properties	Type	Description
Import Type	Integer	0: Import from folder. 1: Import from list file.
ImportPath	String	Path to import from.
ImportMask	String	The mask of image files / list files to import (e.g., *.LST, *.*, etc.)
FolderIncludeSubFolders	Boolean	If true, indicates that sub folders will be processed when importing from the specified ImportPath
FolderImportOption	Integer	0: Delete files after processing. 1: Rename file extension after processing. 2: Add prefix to file after processing.
FolderImportRename	String	The extension to use when renaming a file after processing.
FolderImportPrefix	String	The prefix to add to a file after processing.

Properties	Type	Description
FolderPrimaryOrder	Integer	Indicates which data element to use when sorting files in a folder prior to processing. soNone = 0 soFileName = 1 soFileExtension = 2 soFileCreateDate = 3 soFileModifiedDate = 4 soFileAccessedDate = 5
FolderSecondaryOrder	Integer	Indicates which data element to use when performing a secondary sort on the files in a folder prior to processing. soNone = 0 soFileName = 1 soFileExtension = 2 soFileCreateDate = 3 soFileModifiedDate = 4 soFileAccessedDate = 5
FolderPrimaryASC	Boolean	If true, the primary sort will be in ascending order. Otherwise, it will be in descending order.
FolderSecondaryASC	Boolean	If true, the secondary sort will be in ascending order. Otherwise, it will be in descending order.
FileFieldDelimiter	String	The character used to delimit fields in a list file.
FileFieldCount	Integer	Number of fields defined for an import file.
FileFieldx	String	Field name assigned to a position. x is a number representing a field position.
FileImportOption	Integer	0: Delete list file after processing. 1: Rename extension of list file after processing.
FileImportRename	String	The extension to use when renaming a list file after processing.
FileDeleteImageFiles	Boolean	If true, the image files referenced in a list file will be deleted after processing of the list file.
BatchSizeOption	Integer	0: Start a new batch when max pages are reached. 1: Start a new batch for each file.
BatchSizeMaxPages	Integer	The maximum number of pages allowed in a batch. Used if BatchSizeOption = 0.

5.7.6 ispIndex Object

Description: This object represents an index field for a specific page.

Properties	Type	Description
Name	String	Field Name
Value	String	Field Name

5.7.7 FaxPressJob Object

Description: This object contains information read directly from FaxPress about a fax that has been received. It contains such information as the number of pages received and the date and time of receipt.

All property values (except FileNameRoot and ControlFileName) are set by the FaxPress system.

Properties	Type	Description
jobId	Long	The internal job ID of the fax job.
ObjectID	Long	The internal Object ID of the fax job.
flags	Long	Internal flags.
JobStatus	Integer	Status of the job, where: QUEUED = 1, HELD = 2, DELAYED = 3, PRINTING = 4, DIALING = 5, ACTIVE = 6, RETRY = 7, FAILED = 8, UNROUTED = 9, ROUTED = 10, COMPLETE = 11, INCOMPLETE = 12, CAUTION = 13, READY = 14, SENT = 15.
fileType	Integer	Type of file, where: ASCII = 1, PCX = 2, PCN = 3, PCL = 4, JOB = 5, CFX = 6, MMR = 10, DOC = 20, XLS = 21.
resolution	Integer	Resolution of the fax, where: STANDARD = 7, FINE = 8, SFINE1 = 9, SFINE2 = 10, SFINE3 = 11.
received	Date	When the fax arrived.
Pages	Integer	Length of the fax in pages.
RemoteID	String	Remote terminal ID of the origin of the fax.
FileNameRoot	String	The file name of the control file created when the fax is exported from FaxPress.
ControlFileName	String	The file name (including path) of the control file created when the fax is exported from FaxPress.

5.7.8 EmailMsg Object

Description: This object holds information about the current email message being processed.

Properties	Type	Description
BodyText	String	Contains the body text of the message.
ContentSubType	String	Contains the content sub type of the message. For example, if the email header contains Content-Type: text/plain, the content sub type is plain.
ContentType	String	Contains the content type of the message. For example, if the email header contains Content-Type: text/plain, the content type is text.
MsgDate	String	The date and time of the message.
From	String	The friendly name of the sender (if available).
FromAddr	String	The email address of the sender.
MessageID	String	The message ID from the email header.
Priority	String	The priority of the message from the email header.

Properties	Type	Description
ReplyTo	String	The friendly name of the entity specified to receive all replies to this message.
ReplyToAddr	String	The reply to email address specified in the message.
Size	Long	The size (in bytes) of the message.
Subject	String	The subject of the message.
Attachments	cAttachments	A collection of Attachment objects.
Recipients	cRecipients	A collection of Recipient objects.

5.7.9 Attachment Object

Description: This object holds information about an email attachment.

Properties	Type	Description
ContentID	String	Content ID taken from the MIME part header.
ContentSubType	String	The content sub type of the MIME part header. For example, if the MIME part header contains Content-Type: application/x-zip-compressed, the content sub type is x-zip-compressed.
ContentType	String	The content type of the MIME part header. For example, if the MIME part header contains Content-Type: application/x-zip-compressed, the content type is application.
Description	String	The text description of the MIME part or attachment.
Encoding	String	A description of the encoding scheme used on the MIME part.
Filename	String	The filename of the attachment.
Name	String	The name of the attachment (similar to Filename but taken from another section of the MIME part header).
Size	Long	The size (in bytes) of the attachment in its decoded state.

5.7.10 Recipient Object

Description: This object holds information about the email recipient.

Properties	Type	Description
Address	String	Email address of the recipient.
Name	String	Friendly name of the recipient.
RecipType	Long	Recipient type (1 = Normal, 2 = CC, 3 = BCC)

5.7.11 oTextPage Object

Description: This object includes information regarding the virtual printing process of a non-image file. It references a file containing all of the ASCII data capture.

Properties	Type	Description
Filename	String	The name of the file containing the page's text in ASCII format
HorizontalDPI	Integer	The horizontal DPI of the page

Properties	Type	Description
Orientation	Long	The orientation of the page (1 = Portrait, 2 = Landscape)
PixelHeight	Long	The height of the page in pixels
PixelWidth	Long	The width of the page in pixels
VerticalDPI	Integer	The vertical DPI of the page

6

Customizing Capture

This section covers the following topics:

6.1 Differences Between Capture VBA and Microsoft VBA

The following table is a feature comparison of the Capture VBA compliant engine and Microsoft's VBA engine. All Oracle Document Capture components use the Capture VBA Compliant engine.

Feature Comparison	Capture VBA	Microsoft VBA
#Const		X
#If...Then...#Else		X
Abs	X	X
Add	X	X
Addition (+)	X	X
All financial functions		X
All intrinsic data types except Variant	X	X
All traditional Basic file I/O	X	X
And	X	X
AppActivate	X	X
Array	X	X
Asc	X	X
AscB	X	X
AscW	X	X
Atn	X	X
Call	X	X
CBool	X	X
CByte	X	X
CCur	X	X

Feature Comparison	Capture VBA	Microsoft VBA
CDate	X	X
CDbl	X	X
Chr	X	X
ChrB	X	X
ChrW	X	X
CInt	X	X
Class		
Clipboard	X	X
CLng	X	X
Collection		X
Collection access using !	X	X
Comments using ' or Rem	X	X
Const	X	X
Cos	X	X
Count	X	X
CreateObject	X	X
CSng	X	X
CStr	X	X
CVar	X	X
CVDate	X	X
Date	X	X
Date statement		X
DateAdd	X	X
DateDiff	X	X
DatePart	X	X
DateSerial	X	X
DateValue	X	X
Day	X	X
Debug.Print	X	X
Declare (for declaring DLLs)	X	X
Declaring arrays with lower bound <> 0	X	X
Deftype	X	X
DeleteSetting	X	X
Dim	X	X
Division (/)	X	X
Do...Loop	X	X
DoEvents		X
Empty	X	X

Feature Comparison	Capture VBA	Microsoft VBA
End	X	X
Equality (=)	X	X
Eqv	X	X
Erase	X	X
Erl		X
Err	X	X
Err Object	X	X
Error	X	X
Eval		
Eval Function	X	X
Execute		
Execute Statement	X	X
Exp	X	X
Exponentiation (^)	X	X
Expressions containing a rangeof values using the To keyword	X	X
Expressions containing Is keyword or any comparison operators	X	X
False	X	X
Filter		X
Fix	X	X
Fixed-length strings	X	X
For Each...Next	X	X
For...Next	X	X
FormatCurrency		X
FormatDateTime		X
FormatNumber		X
FormatPercent		X
Function	X	X
Friend	X	X
GetObject	X	X
GetSetting	X	X
GetSettings	X	X
GoSub...Return		X
GoTo	X	X
Greater Than (>)	X	X
Greater Than or Equal To (>=)	X	X
Hex	X	X
Hour	X	X

Feature Comparison	Capture VBA	Microsoft VBA
If...Then...Else	X	X
Imp	X	X
Inequality (<>)	X	X
InputBox	X	X
InStr	X	X
InStrB	X	X
InStrRev	X	X
Int	X	X
Integer Division (\)	X	X
Is	X	X
IsArray	X	X
IsDate	X	X
IsEmpty	X	X
IsNull	X	X
IsNumeric	X	X
IsObject	X	X
Item	X	X
Join		X
LBound	X	X
LCase	X	X
Left	X	X
LeftB	X	X
Len	X	X
LenB	X	X
Less Than (<)	X	X
Less Than or Equal To (<=)	X	X
Like	X	X
Line labels	X	X
Line numbers		X
LinkExecute		X
LinkPoke		X
LinkRequest		X
LinkSend		X
LoadPicture		X
Log	X	X
LSet	X	X
LTrim	X	X
Mid	X	X

Feature Comparison	Capture VBA	Microsoft VBA
Mid Statement	X	X
MidB	X	X
Minute	X	X
Modulus arithmetic (Mod)	X	X
Month	X	X
MonthName	X	X
MsgBox	X	X
Multiplication (*)	X	X
MyCollection!Foo	X	X
Negation (-)	X	X
New	X	X
Nothing	X	X
Now	X	X
Null	X	X
Oct	X	X
On Error GoTo	X	X
On...GoSub	X	X
On...GoTo	X	X
OnError	X	X
Option Base	X	X
Option Compare		X
Option Explicit	X	X
Option Private Module		X
Optional	X	X
Or	X	X
ParamArray	X	X
Private	X	X
Project References	X	X
Property Get	X	X
Property Let	X	X
Property Set	X	X
Public	X	X
Randomize	X	X
ReDim	X	X
Remove		X
Replace	X	X
Resume		X
Resume Next	X	X

Feature Comparison	Capture VBA	Microsoft VBA
RGB Function	X	X
Right		
RightB		
Rnd		
Round	X	X
RSet	X	X
RTrim	X	X
SaveSetting	X	X
ScriptEngine		
ScriptEngineBuildVersion		
ScriptEngineMajorVersion		
ScriptEngineMinorVersion		
Second	X	X
Select Case	X	X
SendKeys	X	X
Set	X	X
Sgn	X	X
Shell	X	X
Sin	X	X
Space	X	X
Split		X
Sqr	X	X
Static	X	X
Stop	X	X
Str	X	X
StrComp	X	X
StrConv	X	X
String	X	X
String concatenation (&)	X	X
StrReverse	X	X
Sub	X	X
Subtraction (-)	X	X
Tan	X	X
Time	X	X
Time statement		X
TimeSerial	X	X
TimeValue	X	X
Trim	X	X

Feature Comparison	Capture VBA	Microsoft VBA
True	X	X
Type...End Type	X	X
TypeName	X	X
TypeOf	X	X
UBound	X	X
UCase	X	X
Val	X	X
VarType	X	X
Wait	X	
Weekday	X	X
WeekdayName	X	X
While...Wend	X	X
With	X	X
Xor	X	X
Year	X	X

6.2 Capture Electronic Document Provider (EDP) Macros

An EDP macro allows you to modify the batch commit process. Once you develop an EDP macro, you can assign it to one or more commit profiles. (See the *Administrator's Guide for Oracle Document Capture* for more information about commit profiles.) This ability to assign different EDP macros to commit profiles provides you with the flexibility to modify the commit process on a per application basis.

For a list of EDP events, see "EDP Macro Events" on page 6-8.

An EDP macro is used regardless of which Capture tool actually performs a batch commit (i.e., Index, Recognition Server, Oracle Distributed Document Capture Server or Commit Server). As a result, EDP macros are a good choice when you want to consolidate code that gets executed during a batch commit. For example, you may want to automatically update a corporate database for each document committed. If this code is developed in an EDP macro, it does not matter which Capture program performs the batch commit.

If you develop an EDP macro, you are responsible for ensuring that document images are copied from the batch folder and moved into a safe (archival) location. As each document is successfully committed within a batch, Capture will remove the document's images and associated index data from the batch. If there are no more pages remaining in the batch, Capture will delete the batch from Capture.

6.2.1 Error Handling in EDP Macros

EDP macros are responsible for a successful batch commit. Therefore, it is imperative that EDP macros inform Capture if an error occurs, so it does not remove documents that failed. The BeginCommit, ReleaseDocument, and EndCommit events include two parameters called lErrorNum and sErrorDesc. If an error occurs in these events, you must set lErrorNum to a non-zero value and provide the reason for the error in the sErrorDesc parameter.

Important: .lErrorNum is a Long variable type, while sErrorDesc is a String variable type. Failure to properly set lErrorNum and sErrorDesc variables in error situations can cause loss of batch images and index data.

6.2.2 EDP Macro Events

Choose from the following events.

Event	BeginCommit
Description	This event occurs when a batch is about to be committed.
Syntax	eCaptureEdp_BeginCommit(ByVal BatchPath As String, ByVal CommitPath As String, ByVal Batch As Object, lErrorNum As Long, sErrorDesc As String)
Parameters	*BatchPath* is the root batch path as configured in the Capture Batch Setup. *CommitPath* is the root commit path as configured in the Capture Batch Setup. *Batch* is the Capture batch object that represents the batch being committed ("Batch Object" on page 6-10). *lErrorNum* is the error number to report if an error occurs. sErrorDesc is the error description to report if an error occurs.

Event	EndCommit
Description	This event occurs when all documents have finished committing. This event represents the completion of a batch commit.
Syntax	eCaptureEdp_EndCommit(Batch As Object, lErrorNum as Long, sErrorDesc As String)
Parameters	*Batch* is the Capture batch object that represents the batch being committed (see "Batch Object" on page 6-10).*lErrorNum* is the error number to report if an error occurs. *sErrorDesc* is the error description to report if an error occurs.

Event	PostCommitDocument
Description	Event occurs immediately after a document has been committed.
Syntax	eCaptureEdp_PostCommitDocument()
Parameters	None

Event	PreCommitDocument
Description	Event occurs just prior to the Release document event. This event allows the macro access to the original image file names, which are referenced with the Document object. In addition, it allows the macro to specify a override the default Output format for the associated Commit Profile.
Syntax	eCaptureEdp_PreCommitDocument(ByVal Document As Object, OutputDriverName As String, Cancel As Boolean, lErrorNum As Long, sErrorDesc As String)

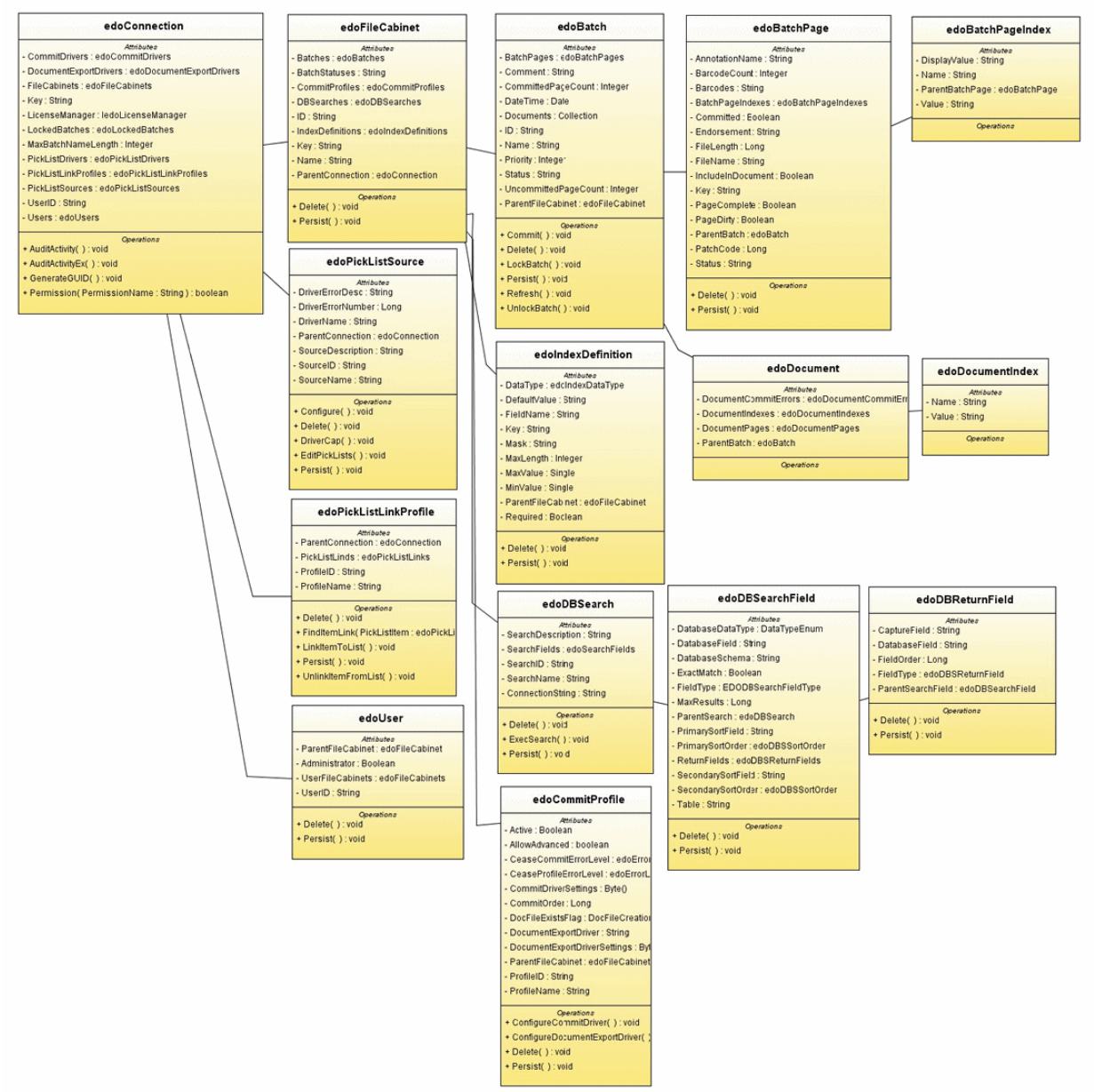
Event	PreCommitDocument
Parameters	*Document* object represents the document being committed (see "Document Object" on page 6-13). *OutputDriverName* is the name of the document output driver being used for the Commit Profile. Valid Output Driver names include: Native File Format, TIFF - Single Page, TIFF - Multiple Page, PDF - Image Only, PDF - Searchable. Set *Cancel* to true to abort the committing of the current document.

Event	ReleaseDocument
Description	This event occurs when a document must be committed. The EDP macro is responsible for moving all the images corresponding to the document into a safe place.
Syntax	eCaptureEdp_ReleaseDocument(ByVal Document As Object, lErrorNum As Long, sErrorDesc As String)
Parameters	*Document* object represents the document being committed (see "Document Object" on page 6-13). *lErrorNum* is the error number to report if an error occurs. *sErrorDesc* is the error description to report if an error occurs.

Event	SetErrorLevel
Description	If an EDP macro returns an error in any of the other events, Capture will execute this event to request an error level. Setting the ErrorLevel is optional, since Capture defaults the error level to Critical.
Syntax	eCaptureEdp_SetErrorLevel(ErrorLevel As Long)
Parameters	*ErrorLevel* values include: 0 = none 1 = warning or critical 2 = critical

6.3 Captovation Capture Objects

The following diagram represents capture objects.



6.3.1 Batch Object

Description: This object represents an entire batch.

Properties		
BatchPages	Object	A collection of BatchPage objects.
Comment	String	The batch comment/note.
DateTime	Date	Date and time the batch was created.
ID	String	Unique ID for the batch.

Properties		
Name	String	Name of the batch.
ParentFileCabinet	Object	Reference to the batch's parent file cabinet (see "FileCabinet Object" on page 6-14).
Priority	Long	Priority for the batch.
Status	String	Status of the batch.

6.3.2 BatchPage Object

Description: This object represents a single page within a batch.

Properties		
AnnotationName	String	Electronic annotation placed on the page.
BarCodeCount	Long	Number of bar codes found on the page.
Barcodes	String()	An array of bar code values recognized.
BatchPageIndexes	Object	A reference to the BatchPageIndexes collection.
Committed	Boolean	Indicates whether or not the batch as been committed.
Endorsement	String	Value endorsed/imprinted on the page.
FileLength	Long	Size of the page (i.e. image file) in bytes.
FileName	String	Fully qualified path to the image file.
IncludeInDocument	Boolean	Indicates whether or not the page will be filed.
PageComplete	Boolean	Indicates whether or not the page has correct index information so that it can be committed.
ParentBatch	Object	Reference to the page's parent batch object.
PatchCode	Long	Patch code value associated with the page, where: No patch code present=0 PATCHVALUE_I=8 PATCHVALUE_II=2 PATCHVALUEIII=1 PATCHVALUEIV=16 PATCHVALUEVI=32 PATCHVALUE_T=4
Status	String	Reserved for future use.

6.3.3 BatchPageIndex Object

Description: This object represents an index field for a specific page.

Properties		
Name	String	Field Name
Value	String	Field Value

6.3.4 CommitProfile Object

Description: This object refers to a commit profile assigned to a file cabinet and its associated settings.

Properties		
Active	Boolean	Determines if the profile will be used when a commit is issued on a batch object.
AllowAdvanced	Boolean	Determines if the advanced commit level handling will apply. If false defaults are used.
CeaseCommitErrorLevel	edoErrorLevel	The error level that will stop the entire commit process. The commit will be stopped at the current document and no other profiles will be executed.
CeaseProfileErrorLevel	Integer	The error level that will stop the current profile. The current profile will stop at the current document that caused the error and the next commit profile in the sequence will be executed.
CommitDriver	String	The name of the commit driver to use when committing.
CommitDriverSettings	Byte()	Driver specific settings. Settings are configured by the admin module and should not be modified by a macro.
CommitOrder	Long	The position in the commit order where the profile will be executed.
DocFileExistsFlag	DocFileCreation Flags	Determines what should happen if a document file exists. Abort, Append or Overwrite. DocCreationFlags include: docOverwrite = 0 docAppend = 1 docAbort = 2
DocumentExportDriver	String	The name of the export driver that will be used to create the output documents.
DocumentExportDriverSettings	Byte()	Driver specific settings for the export driver. Configured by the admin module and not normally used in macros.
ParentFileCabinet	edoFileCabinet	The file cabinet that owns the profile. The same as all other edo objects.
ProfileID	String	The unique id of the profile
ProfileName	String	The human readable name of the profile

6.3.5 Connection Object

Description: Connection is an edoConnection object.

Properties	
DBSearches As edoDBSearches	Returns a collection of database search profiles.
FileCabinets As edoFileCabinets	Returns a collection of file cabinets.
Key As String	Returns a unique string value associated with this particular connection.
LockedBatches As edoCollection	Returns a collection of locked batches.

Properties	
MaxBatchNameLength As Long	Sets/Returns the maximum length of the batch name.
PickListDrivers As edoPickListDrivers	Returns a collection of installed pick-list drivers.
PickListLinkProfiles As edoPickListLinkProfiles	Returns a collection of defined pick-list relationship profiles.
PickListSources As edoPickListSources	Returns a collection of defined pick-list sources.
UserID As String	Returns the current log in User ID.
Users As edoUsers	Returns a collection of defined users.

Methods	
Sub AuditActivity(ByVal ActionID As Long, ByVal ActionDesc As String, ByVal FileCabinet As String, ByVal Batch As String, [ByVal ActionDataInt As Long], [ByVal ActionDataFloat As Single], [ByVal ActionDataText1 As String], [ByVal ActionDataText2 As String], [ByVal ActionDataText3 As String], [ByVal ActionDataText4 As String], [ByVal ActionDataText5 As String])	Inserts an audit record into the ecAudit table.
Function GenerateGUID() As String	Returns a unique GUID
Function Permission(PermissionName As String) As Boolean	Returns a boolean to indicate whether the logged on user has a specific permission. Currently the only valid permission name is Administrator.

6.3.6 Document Object

Description: The document object contains the document index information (i.e., fields and field values) along with the corresponding pages (images) that belong to the document.

Properties		
DocumentIndexes	Object	Collection of DocumentIndex objects (see "DocumentIndex Object" on page 6-13)
DocumentPages	Object	Collection of DocumentPage objects (see "DocumentPage Object" on page 6-13
ParentBatch	Object	Reference to the document's parent batch object (see "Batch Object" on page 6-10

6.3.7 DocumentIndex Object

Description: This object represents an index field for a specific page. See "BatchPageIndex Object" on page 6-11.

6.3.8 DocumentPage Object

Description: This object represents a single batch page. See "BatchPage Object" on page 6-11.

6.3.9 FileCabinet Object

Description: Contains a document field name and corresponding value.

Properties		
ID	String	Unique ID for the file cabinet.
IndexDefinitions	Object	Reference to a collection of index definitions for the file cabinet (see "IndexDefinition Object" on page 6-14).
MacroName	String	Name of the macro assigned to the file cabinet.
Name	String	File cabinet name.

6.3.10 IndexDefinition Object

Description: This object represents a file cabinet field.

Properties		
DataType	edoIndexDataType	Data type for the field (edoTypeAlpha, edoTypeAlphaNumeric, edoTypeDate, edoTypeFloat, edoTypeNumeric).
DefaultValue	String	Default value for the field.
FieldName	String	Name of the field.
Mask	String	Mask for the field.
MaxLength	String	Maximum character length of the value.
MaxValue	Single	Maximum value of a numeric field type.
MinValue	Single	Minimum value of a numeric field type.
Required	Boolean	Indicates whether or not the field is required.

6.3.11 ScanUI Object

Description: Provides programmatic methods to control the user interface.

Properties		
ReviewMode	Boolean	Returns a Boolean flag indicating whether the scan tool is reviewing a batch or not.
TotalPages	Integer	Returns the total number of pages in the batch.

Methods		
RefreshBatches	Refreshes the Batch List using BatchPrefix as a criteria.	RefreshBatches(ByVal BatchPrefix As String
UpdateThumbnailCaptionSuffix	Updates the Thumbnail caption of the batch page. This only applies when ReviewMode is True.	UpdateThumbnailCaptionSuffix(BatchPage As Object, ByVal bUsePatchCode As Boolean, ByVal sSuffix As String)

Methods		
ViewPage	Displays the page indicated in the parameter PageNumber. This only applies when ReviewMode is True.	ViewPage(ByVal PageNumber As Integer)

6.3.12 SearchCriteria Object

Description: The SearchCriteria object represents the search settings selected by the user on the Batch Search Criteria window.

Properties		
DateFrom	String	Search start date, a valid date in string format.
DateTo	String	Search end date, a valid date in string format.
Name	String	Name(s) of the batch(es) to search against. It can contain wild card characters. Wild card characters conforms to VBs Like operator.
Note	String	Search phrase that appears in batch note.
Priorities	Integer	An integer array contains the valid priorities to search against.
PrioritiesCount	Integer	The number of priorities in the previous array.
Priority(Byval Index As Integer)	Integer	Returns a specific priority in the array.
Statuses	String	An array of valid batch statues to search against.
StatusCount	Integer	The number of statuses in the previous array.
Status(Byval Index As Integer)	Integer	Returns a specific status in the array.

6.3.13 Settings Object

Description: This object contains settings for a macro.

Properties		
Contents	String	This is a read-only property that retrieves the entire property bag in on large string.

Methods		
ReadProperty	Returns the value of a specific property.	PropertyName (String): Name of the property to retrieve.
WriteProperty	Saves the value of a specific property.	PropertyName (String): Name of the property to save. PropertyValue (String): Value of the property to save.

A

Keycodes

Keyboard keycodes A - Z are the same as their ASCII equivalents:

Keycode	Key	Keycode	Key
65	A key	78	N key
66	B key	79	O key
67	C key	80	P key
68	D key	81	Q key
69	E key	82	R key
70	F key	83	S key
71	G key	84	T key
72	H key	85	U key
73	I key	86	V key
74	J key	87	W key
75	K key	88	X key
76	L key	89	Y key
77	M key	90	Z key

Keyboard keycodes 0 - 9 are the same as their ASCII equivalents:

Keycode	Key	Keycode	Key
48	0 key	53	5 key
49	1 key	54	6 key
50	2 key	55	7 key
51	3 key	56	8 key
52	4 key	57	9 key

Keys on the Numeric Keypad:

Keycode	Key	Keycode	Key
96	0 key	104	8 key
97	1 key	105	9 key

Keycode	Key	Keycode	Key
98	2 key	106	MULTIPLICATION SIGN (*) key
99	3 key	107	PLUS SIGN (+) key
100	4 key	13	ENTER key
101	5 key	109	MINUS SIGN (-) key
102	6 key	110	DECIMAL POINT (.) key
103	7 key	111	DIVISION SIGN (/) key

Function Keys:

Keycode	Key	Keycode	Key
112	F1 key	120	F9 key
113	F2 key	121	F10 key
114	F3 key	122	F11 key
115	F4 key	123	F12 key
116	F5 key	124	F13 key
117	F6 key	125	F14 key
118	F7 key	126	F15 key
119	F8 key	127	F16 key

Miscellaneous Keys:

Keycode	Key	Keycode	Key
1	Left mouse button	34	PAGE DOWN key
2	Right mouse button	35	END key
3	CANCEL key	36	HOME key
4	Middle mouse button	37	LEFT ARROW key
8	BACKSPACE key	38	UP ARROW key
9	TAB key	39	RIGHT ARROW key
12	CLEAR key	40	DOWN ARROW key
13	ENTER key	41	SELECT key
16	SHIFT key	42	PRINT SCREEN key
17	CTRL key	43	EXECUTE key
18	MENU key	44	SNAPSHOT key
19	PAUSE key	45	INS key
20	CAPS LOCK key	46	DEL key
27	ESC key	47	HELP key
32	SPACEBAR key	144	NUM LOCK key
33	PAGE UP key		

B

Copyright and Patent Notices

Index

A

B

C

D

E

F

G

I

K

L

M

N

O

P

R

S

T

U

V