PaperVision® Capture

Administration Guide
PaperVision Capture Release 79

November 2014

DIGITECH
SYSTEMS

Digitech Systems, Inc.

8400 E. Crescent Parkway, Suite 500

Greenwood Village, CO 80111

Phone: (303) 493-6900 Fax: (303) 493-6979

www.digitechsystems.com

Table of Contents

For Stacey Geringer - this might be of interest to you....

Chapter 12 Custom Code Configuration

With PaperVision Capture's custom code engine, you can write Visual Basic.NET or C# code that can be run at any time during batch processing. Additionally, Digitech Systems provides a .NET Application Programming Interface (API) that you can use for read/write access to batch metadata, documents, images, OCR data, and index values.

Job steps within job definitions contain the custom code capabilities. Each job step can trigger custom code events. These events differ by job step. For example, Indexing job steps can initiate the "Saving Indexes" custom code event. So, on the **Job Definitions** window, you can configure the custom code that the system will run when index values are being saved.

WARNING: Changes made to a batch via custom code that runs in a manual job step may not be reflected in the Operator Console unless your custom code specifies the appropriate user-interface refresh level. See **public enum UIRefreshLevel** under "Enumerations" on page 291 for more information.

Digitech Systems also provides a **Custom Code** job step, which is not event-based. Instead, it will run any code you specify. PaperVision Capture runs **Custom Code** job steps in the background as automatic processes, so you do not see them running within the user interface in PaperVision Capture.You can also use **Custom Code** job steps for validating or manipulating data and interfacing with an external application, such as an external database or line-of-business application.

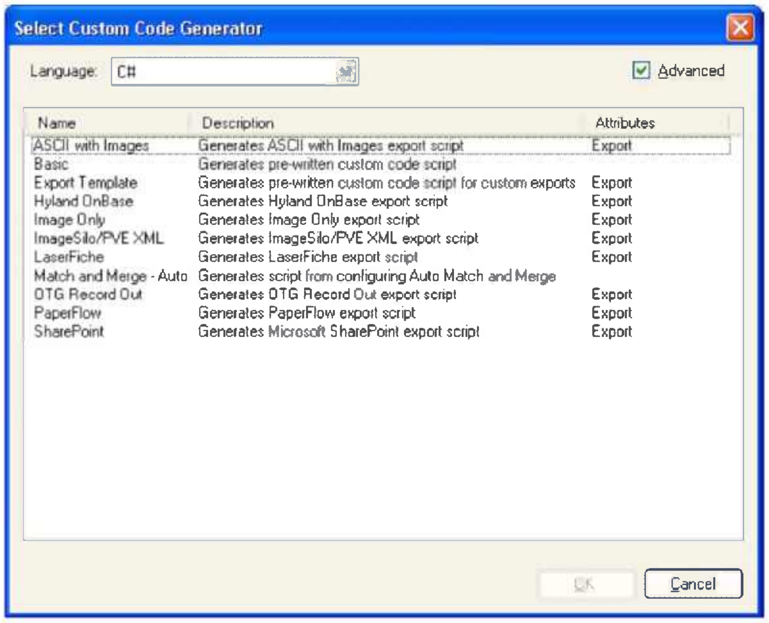
Custom Code Generators

When you configure the **Custom Code** step, you can select either the C# or Visual Basic programming language and the custom code generator that runs automatically during batch processing. Custom Code generators include all PaperVision Capture exports, the **Match and Merge Wizard**, and customizable scripts that contain generic code that you can edit and compile directly on the **Script Editor** window. You can configure custom code generators using dialog boxes that display only the applicable properties for your selection. Default settings are provided for each generator within drop-down menus, editable fields, and check boxes (indicating a default true or false setting). The **Basic** custom code generator provides a generic code template, and the **Export Template** custom code generator provides a generic template for custom exports that you can run automatically during batch processing.

IMPORTANT: You can use the Visual Basic programming language only with the **Match and Merge - Auto**, **Basic**, and **Export Template** custom code generators.

To select a Custom Code Generator

1. After you have logged in to the **PaperVision Capture Administration Console**, expand **Entities**, and then expand *Entity Name*.
2. Click **Capture Jobs**. A listing of jobs appears on the right pane.
3. Select the job you want to edit, and then click **Edit Job** .
4. If necessary, click **Check Out Job** so you can edit it.
5. On the workspace of the **Job Definitions** window, double-click the **Custom Code** job step to display the **Properties** tab on the left pane.
6. On the **Properties** tab, expand **Custom Code Events (Step Level)**.
7. Click **Step Executing**, and then click the ellipsis button to open the **Select Custom Code Generator** dialog box. Each custom code generator and corresponding description are listed.



Name	Description	Attributes
ASCII with Images	Generates ASCII with Images export script	Export
Basic	Generates pre-written custom code script	
Export Template	Generates pre-written custom code script for custom exports	Export
Hyland OnBase	Generates Hyland OnBase export script	Export
Image Only	Generates Image Only export script	Export
ImageSilo/PVE XML	Generates ImageSilo/PVE XML export script	Export
LaserFiche	Generates LaserFiche export script	Export
Match and Merge - Auto	Generates script from configuring Auto Match and Merge	
OTG Record Out	Generates OTG Record Out export script	Export
PaperFlow	Generates PaperFlow export script	Export
SharePoint	Generates Microsoft SharePoint export script	Export

Select Custom Code Generator

8. If you want to see only the generators that you can configure using the provided dialog boxes, rather than editing code in the **Script Editor**, then clear the **Advanced** check box.

NOTE: To remove existing custom code, on the **Properties** tab, expand **Custom Code Events [Step Level]**. Right-click **Step Executing**, and then click **Reset** . Additionally, to prevent the **Select Scripting Language** dialog box from appearing each time you configure custom code, select **Suppress this dialog when creating new custom code.**

---

9. If the **Advanced** check box is selected, from the **Language** list, select the **C#** or **Visual Basic** programming language. Your selected scripting language determines which generators are available for configuration. (See "Exports" on page 306 for information about individual settings for PaperVision Capture exports and the constant values that you can define for each one.)
    - The **Basic** generator lets you write your own custom code directly in the **Script Editor**. See "Script Editor" on page 295 for more information.
    - The **Match and Merge** generator runs code from the **Match and Merge Wizard**, where you set up connection properties for your SQL Server database. See "Match and Merge Wizard" on page 302 for information about configuring this generator.
    - The **Export Template** generator contains additional pre-defined code that will automatically process batches. See "Exports" on page 306 for information about configuring PaperVision Capture exports.
10. To configure a generator, double-click it to access it's corresponding properties. In the dialog boxes that appear, default values and applicable index fields are provided for your reference, and lists and menus contain only the options specific to your selected generator. You can manually enter file paths or browse to the appropriate directory.
11. After you have configured the appropriate properties, click **OK** to save the generator. **Step Executing** On the **Properties** tab now appears as **Enabled**.

NOTE: The most recent template and programming language that you selected is retained for the next time you create a custom code generator.

Digitech Systems' API

You can access Digitech Systems' API from the **Script Editor**. The API provides classes for reading/writing documents and indexes within the current batch. For more information on the Digitech Systems API, launch the **PVCaptureBatchAPI.chm** help file located in the **Docs** directory where PaperVision Capture is installed. This help file provides Microsoft Developer's Network (MSDN)-style documentation on our DSI.Capture.API namespace, including code samples.

Custom code samples (as text or XML files) are in the **Library\Samples** directory where PaperVision Capture is installed. You can cut and paste the code directly into the **Script Editor** for a **Custom Code** step. The following code samples are included:

- **AddPrefixValuetoBatchDocumentIndexes** iterates through all documents comprising a batch and appends prefixes to index values.

NOTE: This script is intended to be run in an automated custom code step.

- **AutoCreateBatches_Part1** and **AutoCreateBatches_Part2** use the **PaperVision Capture Automation Server** to create and populate batches on the fly through two custom code steps (for example, polling a directory for .TIF files, and then automatically creating batches).

NOTE: Creating and populating batches via automated Custom Code causes the Automation Server to consume a PaperVision Capture Scan license.

- **CalltoCustomAssembly** demonstrates one way to call out to code in your own assembly.
- **CopyIndexValues** duplicates an index value from a source document to one or more subsequent documents.
- **DisplayBatchPageCount** displays the total number of pages in the batch (designed to be run in the Operator Console from a manual custom code execute event).
- **ExportFullTextData** copies full-text OCR data for each document stored in the batch to a specified directory.
- **ImportASCIIwithImages** imports images and index information from a number of other document imaging systems.

NOTE: You must configure constants at the beginning of the script for the operator to successfully run the script.

- **InspectBeforeAddPage** examines the physical dimensions of a scanned image and inserts a document break if the page is detected as an envelope.
- **MatchAndMergeOnIndexValidate** executes custom code that will look up and populate index values when the operator enters a index value and then tabs to the next field.
- **MultiPageTIFFConversion** divides a multiple-page TIFF into separate images (one image per page).
- **OCRFullTextPageStatistics** records Open Text Full-Text OCR statistics per selected output type. Statistics are recorded when the **Open Text Full-Text OCR** step processes a page and converts the page to the selected output format(s).
- **OCRIndexZoneStatistics** records Open Text Zonal OCR statistics when an Open Text OCR zone populates an index value.
- **OCRMarkSenseZoneStatistics** records Open Text Zonal OCR statistics when an Open Text OCR zone inserts an auto document break page between documents.
- **OpenBatchCustomCode** executes custom code when the operator opens a batch in the Operator Console.
- **QCDocumentPageCounts** automatically applies a QC tag to every document in the batch that contains fewer than four and greater than six pages. This script is designed to be run from within a manual job step from the Custom Code Execute event.
- **QCTaggingIndexDocAndPageCustomCode** automatically tags a document containing more than “x” number of pages; pages less than “x” kilobytes; and, index fields containing specific text. For example, to change the maximum number of pages per document to 6, change the following lines to:

```
if (pages.Length > 6)
if(!this.Batch.TryAddDocumentTag(docId, "Document Size", "Document
contains more than 6 pages", out error))
```

- **RecordDailyDocumentAndPageCountStatistics** when used in an automated **Custom Code** step following a **Capture** step, totals the number of documents and pages for batches that flow through a job on a daily basis. Results are available as custom statistics that are viewable/filterable from the **Batch Statistics** screen.
- **Reformat Indexes** attempts to parse all non-text barcode and OCR values that are ingested into index fields without being formatted. The script calls the "TrySetValueFormatted" method after attempting to parse the value, and then tags the index if the value cannot be formatted.
- **SendEmail** runs custom code that sends an email, for example, you could modify this code to send an email when an unclassified document is found in a batch.
- **SetScanDate** automatically sets a scan date index value (document creation date) into the batch for every document. The document's creation date is the date/time the document entered the batch. The date/time value is stored in Universal Time Coordinated (UTC), also known as Greenwich Mean Time (GMT). For example, Denver, Colorado's UTC time at 2:00 PM on April 9, 2009 will display as "04/09/2009 20:00:00." To change the date/time value to your local time zone instead of UTC, change the code in line 46 to:

  ```
  if (!this.Batch.TrySetIndexValue(id, "ScanDate",
  documentCreatedDate.ToLocalTime(), true, out error))
  ```
- **SubmitBatchCustomCode** executes custom code when the operator submits a batch in the Operator Console.
- **ValidateIndex** provides an example of how to validate an index field value.

Batch Property

Within your custom code, you can access the Digitech Systems API via the Batch property. The Batch property is of the type **DSI.Capture.API.Batch** and represents the primary entry point for the Digitech Systems API.

For example, to insert a new document to a batch within your CallHandler method (C# in this case), you can type:

```
this.Batch.TryInsertDocument(/*see API documentation for parameters*/)
```

Another approach is to call out to your own assembly and pass the instance of the Batch object to your code (again, the instance is available as the "Batch" property inside the pre-written "Code" class.) This approach would allow you to use Visual Studio for coding. Then, at run time, you would need to ensure that your assembly is located in the same directory as the PaperVision Capture executable files.

Custom Code Event Arguments

Each custom code event exposes an argument parameter that is specific to the given event type. Within your code, you can access these arguments to read event-specific data and to configure settings. For example, your code can change a property that determines the action that is triggered in the PaperVision Capture Operator Console after the event. The event-specific arguments are listed below.

NOTE: The following classes are derived from the .NET System.Data.DataSet class and support all DataSet properties and functions. Additionally, DataSets are mapped to index values in the Operator Console's Index Manager.

Add Page Event - CCustomCodeNewImageEventArgs

The Add Page event uses the CCustomCodeNewImageEventArgs class to pass every scanned image to the custom code. Use of this argument is illustrated in the InspectBeforeAddPage sample script:

```
CCustomCodeNewImageEventArgs args = base.Parameter as
CCustomCodeNewImageEventArgs;
```

The following properties are located within the custom code:

1. **`Image.Attributes`** (hashtable containing the following image attributes):
   a. **`PageSide`**: **`string`** (indicates the side of the page as "Front" or "Back")
   b. **`DriverName`**: **`string`** (indicates the name of the scanner driver)
2. **`PageTags: TagInfo[]`**

This property can be used to specify one or more page tags to be added after the page has been appended to the batch. Tags added to a break page (based on job configuration settings to delete break pages) will be ignored.

Barcode Detected Event - BarcodeReadEventArgs

The Barcode Detected event uses the BarcodeReadEventArgs class to pass every barcode's data (from each barcode zone) to the custom code. This event is triggered each time a barcode is successfully detected during scanning (multiple barcodes can be detected per page).

The following properties are located within the custom code:

1. **`BarcodeItem Properties`**

   These properties contain all barcode data, including barcode value, location, size, orientation, and barcode type.
2. **`PageTags: TagInfo[]`**

   This property can be used to specify one or more page tags to be added after the page has been appended to the batch. Tags added to a break page (based on job configuration settings to delete break pages) will be ignored.

Custom Code Execution Event - ManualCustomCodeEventArgs

The Custom Code Execution event uses the ManualCustomCodeEventArgs class to pass the operator's index values to the manual custom code event. This event is triggered when the operator triggers the Execute Custom Code operation in the Operator Console.

```
ManualCustomCodeEventArgs args = base.Parameter as
ManualCustomCodeEventArgs;
```

Index Populated Event - IndexPopulateEventArgs

The Index Populated event uses the IndexPopulateEventArgs class to pass the operator's index values to the custom code. This event is triggered when an index value is populated.

```
IndexPopulateEventArgs args = base.Parameter as IndexPopulateEventArgs
```

Index Validate Event - IndexValidateEventArgs

The Index Validate event uses the IndexValidateEventArgs class to pass the operator's index values to the custom code. This event is triggered once the operator proceeds or tabs to the next index field in the Index Manager.

```
IndexValidateEventArgs args = base.Parameter as IndexValidateEventArgs;
```

OCR Statistics Event - OCRFullTextPageProcessedEventArgs

The OCR Statistics custom code event uses the OCRFullTextPageProcessedEventArgs class to pass Open Text full-text data from each page (per selected output format) to the custom code. For each output type, this event is triggered once a page has been converted to PDF, PaperVision Enterprise, PaperFlow, or Text full-text output.

The following properties are located within the custom code:

1. **`DocumentId: string`**
2. **`PageId: Guid`**
3. **`PageIndex: int32`**
4. **`OCRWords: int32`**

   The `OCRWords` property contains the following variables:

   ```
   internal OCRCharacter[] characters = new OCRCharacter[] { };
   internal Int32 line = 0;
   internal System.Drawing.Point location = new System.Drawing.Point();
   internal System.Drawing.Size size = new System.Drawing.Size();
   ```

   The `OCRCharacter` variable contains the following properties:

   ```
   public System.Drawing.Point Location
   {
       get
       {
           return location;
       }
   }
   public System.Drawing.Size Size
   {
       get
       {
           return size;
       }
   ```

```
    }
    public Byte Confidence
    {
        get
        {
            return confidence;
        }
    }
    public Char Code
    {
        get
        {
            return code;
        }
    }
    public bool Rejected
    {
        get
        {
            return rejected;
        }
    }
    public Char[] Alternatives
    {
        get
        {
            return alternatives;
        }
    }
```

5. `RecognitionTime: int32 (milliseconds)`
6. `AdditionalValues: Hashtable`
7. `ConverterName: string`

OCR Statistics Event - OCRIndexZoneProcessedEventArgs

The OCR Statistics custom code event uses the OCRIndexZoneProcessedEventArgs class to pass index values populated by Open Text OCR zones to the custom code. This event is triggered once the contents of an Open Text OCR zone populate an index value.

The following properties are located within the custom code:

1. `DocumentId: string`
2. `PageId: Guid`
3. `PageIndex: int32`
4. `OCRWords: int32`

   The `OCRWords` property contains the following variables:

   ```
   internal OCRCharacter[] characters = new OCRCharacter[] { };
   internal Int32 line = 0;
   internal System.Drawing.Point location = new System.Drawing.Point();
   internal System.Drawing.Size size = new System.Drawing.Size();
   ```

   The `OCRCharacter` variable contains the following properties:

   ```
    public System.Drawing.Point Location
           {
               get
               {
                   return location;
               }
           }
           public System.Drawing.Size Size
           {
               get
               {
                   return size;
               }
           }
           public Byte Confidence
           {
               get
               {
   ```

```
                return confidence;
            }
        }
        public Char Code
        {
            get
            {
                return code;
            }
        }
        public bool Rejected
        {
            get
            {
                return rejected;
            }
        }
        public Char[] Alternatives
        {
            get
            {
                return alternatives;
            }
        }
```

5. **`RecognitionTime: int32 (milliseconds)`**
6. **`AdditionalValues: Hashtable`**
7. **`FieldName: string`**

OCR Statistics Event - OCRMarkSenseZoneProcessedEventArgs

The OCR Statistics custom code event uses the OCRMarkSenseZoneProcessedEventArgs class to pass auto document break zone statistics to the custom code. This event is triggered when an Open Text OCR zone inserts an auto document break page between documents.

The following properties are located within the custom code:

1. **`DocumentId: string`**
2. **`PageId: Guid`**

3. **`PageIndex: int32`**
4. **`OCRWords: int32`**

   The `OCRWords` property contains the following variables:

   ```
   internal OCRCharacter[] characters = new OCRCharacter[] { };
   internal Int32 line = 0;
   internal System.Drawing.Point location = new System.Drawing.Point();
   internal System.Drawing.Size size = new System.Drawing.Size();
   ```

   The `OCRCharacter` variable contains the following properties:

   ```
           public System.Drawing.Point Location
           {
               get
               {
                   return location;
               }
           }
           public System.Drawing.Size Size
           {
               get
               {
                   return size;
               }
           }
           public Byte Confidence
           {
               get
               {
                   return confidence;
               }
           }
           public Char Code
           {
               get
               {
                   return code;
   ```

```
        }
    }
    public bool Rejected
    {
        get
        {
            return rejected;
        }
    }
    public Char[] Alternatives
    {
        get
        {
            return alternatives;
        }
    }
```

5. `RecognitionTime: int32 (milliseconds)`
6. `AdditionalValues: Hashtable`

Saving Indexes Event - IndexSaveEventArgs

The Saving Indexes event uses the IndexSaveEventArgs class to pass the operator's index values to the custom code. The Saving Indexes event is triggered as index values are saved to the batch. This class contains the BatchNavigation enumeration property that determines which document (in the Operator Console) opens immediately after indexes are saved.

```
IndexSaveEventArgs args = base.Parameter as IndexSaveEventArgs;
```

NOTE: By default, the Saving Indexes event proceeds to the next document.

Within your custom code, you can use the following constants to set the BatchNavigation enumeration property:

1. None: Remains on current document
2. NextDoc: Proceeds to next document
3. PreviousDoc: Returns to previous document
4. LastDoc: Proceeds to last document in batch
5. FirstDoc: Returns to first document in batch

For example, you can configure the BatchNavigation enumeration property to remain on the current document after index values are saved:

```
args.BatchNavigation = BatchNavigation.None;
```

Additional API Functions

In addition to the API Functions documented in the **PVCaptureBatchAPI.chm** help file, the API functions described in this section can be used within your custom code.

Custom Code/Export Functions
`protected string[] GetPageFiles(string documentID)` Returns path values for all images contained in a document (from all pages)
`protected Stream GetFileStream(PVFile file)` Returns the stream for a specified PVFile
`protected Stream[]GetDocumentStreams(string documentID)` Returns an array of streams for all files contained in a document (from all pages)
`protected Stream[] GetDocumentStreams(string documentID, string jobStepName, bool bitonal)` Returns streams for all files contained in a document (from all pages) based on job step name and bitonal option
`protected void CopyStreamToDisk(Stream stream, string path)` Copies content of a stream to disk
`public string[] CopyFilesToDisk(string documentID, string rootPath)` Copies all files from a document (from all pages) to a folder and returns an array for all image path values
`protected void SetPersistValue(string key, string value, string rootPath)` Copies all files from a document (from all pages) to a folder based on job step name and bitonal option
`protected string Get PersistValue(string key, string rootPath)` Reads persisted value for a key
`protected string GetNextLockedPath(string root, Int32 maxExportSize, bool exclusive)` Returns the next available path (path is locked before it is returned) **NOTE**: If you set the EXCLUSIVE_EXPORT script constant to **True**, the function will throw an exception if the last available folder is in use. If you set the EXCLUSIVE_ EXPORT script constant to **True**, it is strongly recommended to specify an automation server that will process exports. The automation server can be assigned within each export generator's **Configuration > Options** tab. See "Exports" on page 306 for more information.

`String GetNextLockedPath(string root, Int32 maxExportSize, ExcludePathDelegate excludeFunction, bool exclusive)` Returns the next available path (path is locked before it is returned) **NOTE**: If you set the EXCLUSIVE_EXPORT script constant to **True**, the function will throw an exception if the last available folder is currently is in use. The delegate is used to determine which folders should be skipped. In addition, if you set the EXCLUSIVE_ EXPORT script constant to **True**, it is strongly recommended to specify an automation server that will process exports. The automation server can be assigned within each export generator's **Configuration > Options** tab. See "Exports" on page 306 for more information.
`protected string GetNextLockedPath(string root, Int32 maxExportSize)` Returns the next available path (path is locked before it is returned) **NOTE**: If using this custom code function in conjunction with the EXCLUSIVE_EXPORT script constant (set to **True**), it is strongly recommended to specify an automation server during export configuration. The automation server can be assigned within each export generator's **Configuration > Options** tab. See "Exports" on page 306 for more information.
`protected void UnlockPath(string path)` Deletes lock for a specified path
`void ClearRootPath(string path)` Deletes all folders containing empty subfolders for all folders listed under 'path'
`protected void SetExportComplete(string path)` Flags folder as complete by dropping export.complete file
`protected bool IsExportComplete(string path)` Checks whether export folder is flagged as complete
`protected bool IsExported(string documentID)` Checks whether document was previously exported
`protected bool SetExported(string documentID)` Sets the document's exported status
`protected void DeleteDocument(string documentID)` Deletes document after it has been exported
`protected void SetStatus(string status, Int32 percentage)` Returns percentage of custom code that has been executed
`Protected int GetNonExportedDocumentCount();` Returns the number of non-exported documents

`protected string[] GetPageText (string filePath)` Returns text for each page
`protected string[] GetOCRFiles (string documentID, string stepName, string converterCode)` Returns Full-Text OCR files belonging to a specific converter
`string[] GetOCRFiles (string documentID, string stepName, string converterCode, string path)` Writes Full-Text OCR files belonging to a specific converter to directory 'path' **Important**! The caller is responsible for post-processing clean-up if the files are not required.
`string ConvertImages(string[] sourceFiles, string destinationFile, ConvertFileType convertFileType)` Converts one or more images to a single destination image file and returns the actual path under which the file was saved
`Int32 GetPageCount(string sourceFile)` Returns the number of pages found in a multiple-page image
`string GetPageImage(string sourceFile, Int32 pageIndex, string destinationFile, OutputFileType outputFileType)` Retrieves a specific image referenced by a specific page index in a multiple-page image
`protected string[] GetPageFiles(string documentID)` Returns a path value for all images belonging to a document (from all pages)
`bool IsMultipageFormat(ConvertFileType convertFileType)` Determines if the passed file type supports multiple-page format
`Int32 GetBlankIndexCount()` Returns the number of blank indices
`string[] GetAvailableFields()` Returns the set of fields that can be written to
`string GetIndexValue(string fieldname)` Returns the field value for the specified field name
`void SetIndexValue(string fieldname, string fieldValue)` Assigns a field value for a specified field name **NOTE**: This function cannot be used with a detail set field; otherwise, an exception will result. Also, when called from within an Index Validate event, this function can only be used for the target index.
`string[] GetDetailSetFields()` Returns the field names of the detail set in Match and Merge

`void AssignDetailSet(DataRow row)` Assigns a detail set field in automated match and merge using a single passed DataRow
`void AssignDetailSet(DataSet dataset)` Assigns detail set values from a DataSet (returned from the database) - used in match and merge
`void AssignDetailSet(DataRow row, DataSet indices)` Assigns a detail set from a passed DataRow value (manual match and merge) - detail set is not written to the batch; instead, it is written to the indices DataSet which passed from the UI
`void AssignDetailSet(DataSet dataset, DataSet indices)` Assigns detail set values to passed indices (manual match and merge)
`void UpdateCurrentIndex(DataRow row)` Updates the current index value from the passed DataRow - row is retrieved from a dataset populated by the SQL database (match and merge)
`Bool IsFieldDetailSet(string fieldName)` Checks whether the specified field is a detail set field
`PVIndexMetadata GetIndexMetadata(string fieldName)` Returns metadata for an index
`bool IsFieldEmpty(string fieldName)` Checks whether a field is empty
`string GetMappedColumn(string fieldName)` Returns the mapped column to a specific field name (match and merge)
`DataTable GetMapping()` Returns a mapping table between indices and table columns (match and merge)
`string GetWhereClause()` Generates a WHERE clause to be used in the SQL query (match and merge)
`string GetWhereClause(DataRow row)` Generates a WHERE clause to be used in the SQL query that uses the values in DataRow to add conditions (match and merge)
`string[] GetDocumentIDs()` Returns list of document id values
`PVPage[] GetPages(string documentID)` Returns a list of pages for a specific document
`string GetPath(PVFile file)` Returns a path for a specified file
`PVIndex[] GetIndices(string documentID)` Returns a list of indices for a specific document
`PVDetailSet[] GetDetailSets(string documentID)` Returns the detail set values for a specific document
`PVFile GetPreferredFile(PVPage, string jobStepName, bool bitonal)`

Returns the file that matches the bitonal value (otherwise, first file in array is returned)
`string GetExtension(string imagePath)` Returns the extension of an image path

Enumerations

The enumerations described in this section can be used within your custom code.

public enum ConvertFileType

This enumeration is used by the ConvertImages() function and specifies the conversion types that will be applied to one or more images.

```
 {
        /// <summary>
        /// No file conversion (returns image input path and appends an
extension if not passed in destinationFile variable)
        /// </summary>
        CVT_NO_CONVERSION,
        /// <summary>
        /// TIFF with Group IV and/or medium JPEG compression (single- or
multi-page)
        /// </summary>
        CVT_TIFF_G4_MEDJPG,
        /// <summary>
        /// TIFF with Group IV and/or LZW compression (single- or multi-page)
        /// </summary>
        CVT_TIFF_G4_LZW,
        /// <summary>
        /// TIFF with no compression (single- or multi-page)
        /// </summary>
        CVT_TIFF_NONE,
        /// <summary>
        /// PDF with Group IV and/or medium JPEG compression (single- or multi-
page)
        /// </summary>
        CVT_PDF_G4_MEDJPG,
        /// <summary>
        /// PDF with Group IV and/or LZW compression (single- or multi-page,
and image-only PDFs)
```

```
        /// </summary>
        CVT_PDF_G4_LZW,
        /// <summary>
        /// JPEG with medium JPEG compression (single-page only)
        /// </summary>
        CVT_JPG_MEDJPG,
        /// <summary>
        /// GIF (single-page only)
        /// </summary>
        CVT_GIF,
        /// <summary>
        /// BMP (single-page only)
        /// </summary>
        CVT_BMP,
        /// <summary>
        /// PNG (single-page only)
        /// </summary>
        CVT_PNG
        /// <summary>
        /// JPEG 2000
        /// </summary>
        CVT_JPG2000


    }
```

public enum OutputFileType

This enumeration is used by the GetPageImage() function, and specifies the output file types when single pages are retrieved from a multiple-page image.

```
  {
        /// <summary>
        /// JPEG
        /// </summary>
        OFT_JPG
        /// <summary>
        /// TIFF
        /// </summary>
        OFT_TIFF
        /// <summary>
        /// Bitmap
        /// </summary>
        OFT_BMP
      }
```

public enum UIRefreshLevel

This enumeration synchronizes the Operator Console's user interface with any changes made to the batch via custom code. Setting the UIRefreshLevel in custom code forces the user interface to refresh the selected component specified by the enumeration value (None, Index, CurrentDocumentIndexes, etc.). If you use either the Index Populated or Index Validate Custom Code Event to change an index value, the Operator Console's Index Manager will remain synchronized using the **UIRefreshLevel.Index** value.

```
 {
   /// <summary>
   /// no UI refresh required
   /// </summary>
   None = 0x00,
   /// <summary>
   /// index field needs to be refreshed (i.e., via IndexValidate or
 IndexPopulate
 event)
   /// </summary>
   Index = 0x01,
   /// <summary>
   /// all indexes for current document need to be refreshed (does not apply to
 Match
 and Merge)
   /// </summary>
   CurrentDocumentIndexes = 0x02,
   /// <summary>
   /// current page needs to be refreshed
   /// </summary>
   SinglePage = 0x04,
    /// <summary>
    /// multiple pages need to be refreshed
    /// </summary>
    MultiPage = 0x08
   }
```

Public Properties

The public properties listed in this section can be used within your custom code.

```
        /// <summary>
        /// Batch object
        /// </summary>
        public PVBatch Batch

        /// <summary>
        /// Parent window
        /// </summary>
        public Control Parent

       /// <summary>
       /// Control referencing the current index
       /// </summary>
       public Control Control

       /// <summary>
       /// Used to pass optional parameters
       /// </summary>
       public object Parameter

       /// <summary>
       /// Code result that returns status of custom code execution
       /// </summary>
       public CodeResult CodeResult

       /// <summary>
       /// PDF Resolution used when importing PDF files
```

```
/// </summary>
public Int32 PDFResolution


 /// <summary>
/// PDF Smoothing option used when importing PDF files
/// </summary>
public PDFSmoothing PDFSmoothing
```

Debugging Custom Code

Custom code that you type on the **Script Editor** window is compiled on-the-fly by PaperVision Capture, so there is no way to debug or step through this code at run time. However, if you write code in your own assemblies and call out to these pre-compiled assemblies, then you can debug this code by attaching your debugger to the appropriate capture process.

For code that is run in a manual job step (for example,code running in a "Saving Indexes" event), then you should attach your debugger to the **CaptureClient.exe** process.

To debug code that is executed in an automated custom code step:

1. On the machine where the code is going to be executed, stop the **PaperVision Process Initiator Windows** service.
2. Set your debugger to start an external application for debugging.
3. From the directory where PaperVision Capture is installed, choose the **DSI.PVECommon.PVProcWork.exe** executable and pass a command line argument of "0." When you start this executable, it will execute any pending "Process Batch" operations (including executing custom code steps) that have been appropriately scheduled in the "Automation Service Scheduling" on page 27 screen.
4. When you are finished debugging, restart the **PaperVision Process Initiator Windows** service.

WARNING: Do not attempt to debug code in a production environment. Doing so may adversely impact system performance and have unpredictable impacts on customer data and end-user functionality.

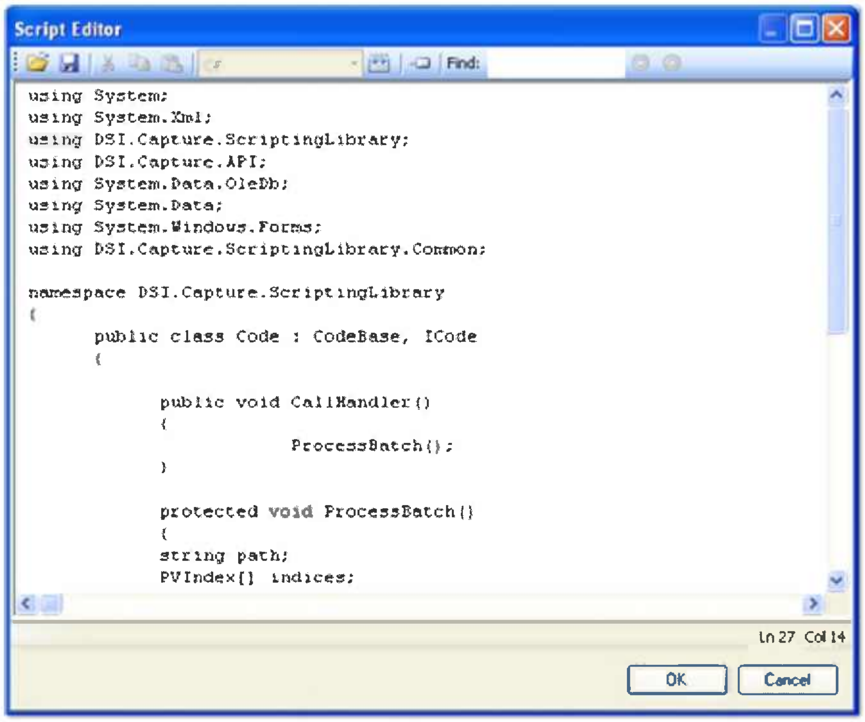
Script Editor

The **Script Editor** launches with pre-written, generic code that you can edit and compile directly in the window. The **Script Editor** window contains the "CallHandler" pre-written method. Although you can add new methods or properties to the "Code" class or call out to other classes (even those defined in your own, separately-compiled assemblies), you should not remove the "CallHandler" method since it is the entry point for executing your custom code. If you call out to other namespaces, remember to add a reference to the necessary assemblies. (See "References" on page 299 for more information.)

Opening the Script Editor

1. After you have logged in to the **PaperVision Capture Administration Console**, expand **Entities**, and then expand *Entity Name*.
2. Click **Capture Jobs.** A listing of jobs appears on the right pane.
3. Select the job you want to edit, and then click **Edit Job** .
4. If necessary, click **Check Out Job** so you can edit it.
5. On the workspace of the **Job Definitions** window, double-click the **Custom Code** job step to display the **Properties** tab on the left pane.
6. On the **Properties** tab, expand **Custom Code Events (Step Level)**.
7. Click **Step Executing**, and then click the ellipsis button to open the **Select Custom Code Generator** dialog box.
8. Ensure that the **Advanced** check box is selected.
9. From the **Language** list, select the **C#** or **Visual Basic** programming language.
10. From the list of generators, select one of the following.
    - Select **Basic** to write your own custom code directly in the **Script Editor**.
    - Select **Export Template** to open a pre-defined custom code script for custom exports that you can edit in the **Script Editor**.

The **Script Editor** appears similar to the following.



```
using System;
using System.Xml;
using DSI.Capture.ScriptingLibrary;
using DSI.Capture.API;
using System.Data.OleDb;
using System.Data;
using System.Windows.Forms;
using DSI.Capture.ScriptingLibrary.Common;

namespace DSI.Capture.ScriptingLibrary
{
        public class Code : CodeBase, ICode
        {

                public void CallHandler()
                {
                                ProcessBatch();
                }

                protected void ProcessBatch()
                {
                string path;
                PVIndex[] indices;
```

Script Editor

Importing Custom Code

The **Import** command lets you load an external custom code XML file into the **Script Editor.**

To import an external XML file

1. If the **Script Editor** window is not open, complete the procedure under "Opening the Script Editor" on page 296.
2. On the toolbar, click **Import** .
3. In the **Open** dialog box, locate, and then select the XML file you want to import.
4. Click **Open.**

Exporting Custom Code

The **Export** command lets you export custom code as an XML file.

To export custom code

1. If the **Script Editor** window is not open, complete the procedure under "Opening the Script Editor" on page 296.
2. On the toolbar, click **Export** .

   NOTE: You cannot export code that does not compile successfully in the **Script Editor**.

3. In the **Save As** dialog box, specify in which folder and under what name you want to save the exported XML file.
4. Click **Save**.

Deleting, Copying, and Moving Custom Code

You can delete, copy, and move sections of the custom code within the **Script Editor** or to another editor.

To delete custom code

1. If the **Script Editor** window is not open, complete the procedure under "Opening the Script Editor" on page 296.
2. In the **Script Editor** window, select the code you want to delete.
3. On the toolbar, click **Cut** .

To copy custom code

1. If the **Script Editor** window is not open, complete the procedure under "Opening the Script Editor" on page 296.
2. In the **Script Editor** window, select the code you want to copy.
3. On the toolbar, click **Copy** .

To move custom code

1. If the **Script Editor** window is not open, complete the procedure under "Opening the Script Editor" on page 296.
2. In the **Script Editor** window, select the code you want to move.
3. From the toolbar, click **Cut** if you want to remove the code, or click **Copy** to copy it.
4. Place your cursor at the new location for the code, and then click **Paste** .

Compiling Custom Code

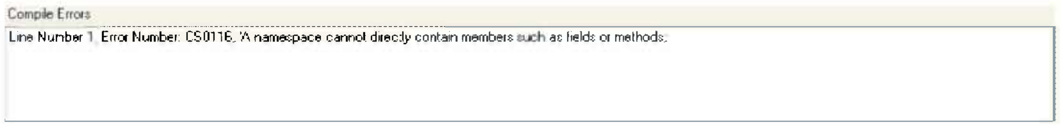
The **Compile** command validates your code.

To compile your code

1. If the **Script Editor** window is not open, complete the procedure under "Opening the Script Editor" on page 296.
2. After writing your custom code in the **Script Editor**, on the toolbar click **Compile** .

   If the code compiles correctly, a "**Code compiled successfully**" message appears.

   If the code does not compile correctly, the **Compile Errors** pane appears at the bottom of the widow similar to the following.

   Compile Errors

   ```
   Line Number 1. Error Number: CS0116. A namespace cannot directly contain members such as fields or methods.
   ```

   Compile Errors

   The **Compile Errors** pane describes the error and its location.
3. Fix any errors that exist, and then compile again.
4. After the "**Code compiled successfully**" message appears, click **OK**.

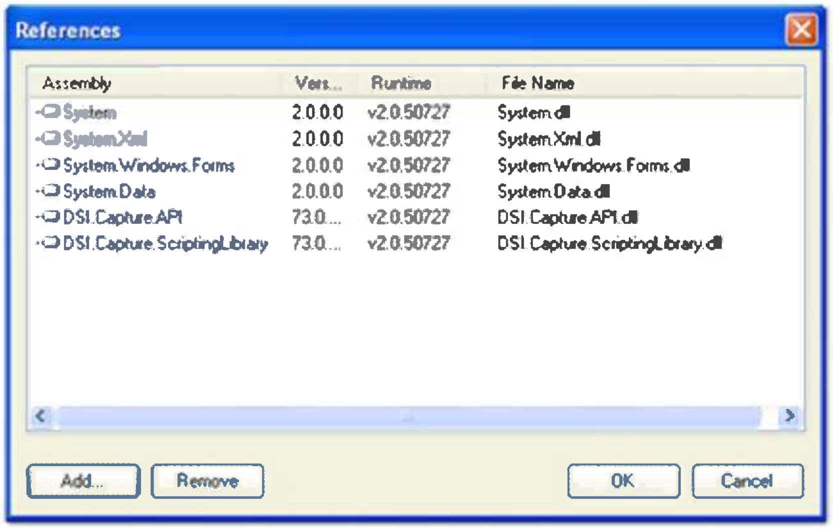
References

References are used to link external assemblies, including standard .NET or custom assemblies that you generate.

To specify references

1. If the **Script Editor** window is not open, complete the procedure under "Opening the Script Editor" on page 296.

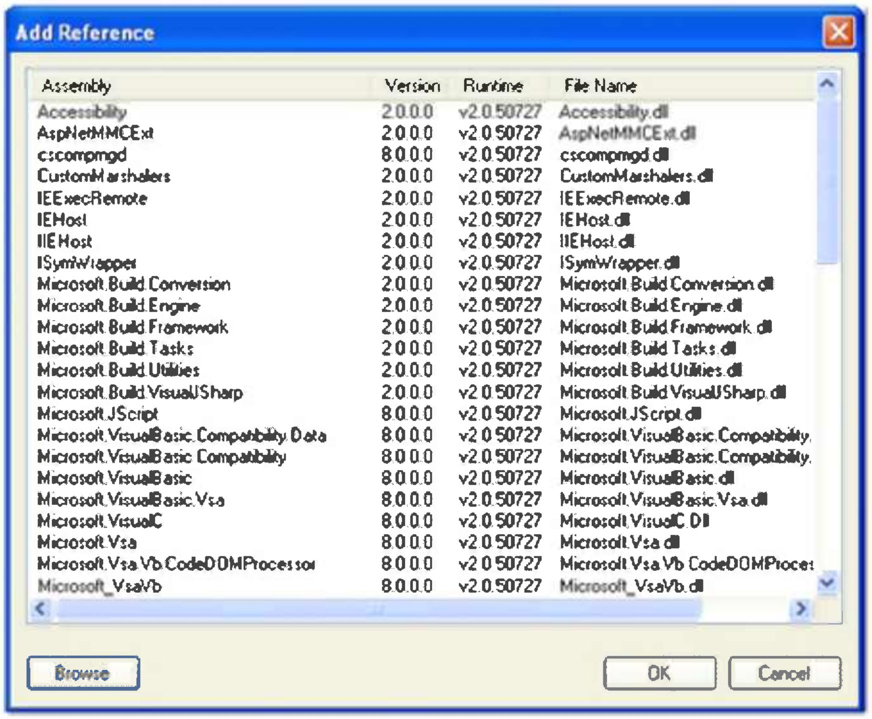
2. On the toolbar, click **References** to open the **References** dialog box where a default listing of assembly files appears. You can add to or remove files from this list.



Assembly	Vers...	Runtime	File Name
System	2.0.0.0	v2.0.50727	System.dll
System.Xml	2.0.0.0	v2.0.50727	System.Xml.dll
System.Windows.Forms	2.0.0.0	v2.0.50727	System.Windows.Forms.dll
System.Data	2.0.0.0	v2.0.50727	System.Data.dll
DSI.Capture.API	7.3.0....	v2.0.50727	DSI.Capture.API.dll
DSI.Capture.ScriptingLibrary	7.3.0....	v2.0.50727	DSI.Capture.ScriptingLibrary.dll

References

3. If you want to add more assembly files, click **Add** to open the **Add Reference** dialog box.



Assembly	Version	Runtime	File Name
Accessibility	2.0.0.0	v2.0.50727	Accessibility.dll
AspNetMMCExt	2.0.0.0	v2.0.50727	AspNetMMCExt.dll
cscompmgd	8.0.0.0	v2.0.50727	cscompmgd.dll
CustomMarshalers	2.0.0.0	v2.0.50727	CustomMarshalers.dll
IEExecRemote	2.0.0.0	v2.0.50727	IEExecRemote.dll
IEHost	2.0.0.0	v2.0.50727	IEHost.dll
IIEHost	2.0.0.0	v2.0.50727	IIEHost.dll
ISymWrapper	2.0.0.0	v2.0.50727	ISymWrapper.dll
Microsoft.Build.Conversion	2.0.0.0	v2.0.50727	Microsoft.Build.Conversion.dll
Microsoft.Build.Engine	2.0.0.0	v2.0.50727	Microsoft.Build.Engine.dll
Microsoft.Build.Framework	2.0.0.0	v2.0.50727	Microsoft.Build.Framework.dll
Microsoft.Build.Tasks	2.0.0.0	v2.0.50727	Microsoft.Build.Tasks.dll
Microsoft.Build.Utilities	2.0.0.0	v2.0.50727	Microsoft.Build.Utilities.dll
Microsoft.Build.VisualJSharp	2.0.0.0	v2.0.50727	Microsoft.Build.VisualJSharp.dll
Microsoft.JScript	8.0.0.0	v2.0.50727	Microsoft.JScript.dll
Microsoft.VisualBasic.Compatibility.Data	8.0.0.0	v2.0.50727	Microsoft.VisualBasic.Compatibility.
Microsoft.VisualBasic.Compatibility	8.0.0.0	v2.0.50727	Microsoft.VisualBasic.Compatibility.
Microsoft.VisualBasic	8.0.0.0	v2.0.50727	Microsoft.VisualBasic.dll
Microsoft.VisualBasic.Vsa	8.0.0.0	v2.0.50727	Microsoft.VisualBasic.Vsa.dll
Microsoft.VisualC	8.0.0.0	v2.0.50727	Microsoft.VisualC.Dll
Microsoft.Vsa	8.0.0.0	v2.0.50727	Microsoft.Vsa.dll
Microsoft.Vsa.Vb.CodeDOMProcessor	8.0.0.0	v2.0.50727	Microsoft.Vsa.Vb.CodeDOMProce:
Microsoft_VsaVb	8.0.0.0	v2.0.50727	Microsoft_VsaVb.dll

Add Reference

4. Do one of the following.
   - Select the file(s) that you want to add, and then click **OK**.
   - Click **Browse** to locate the assembly file you want to use, and then click **Open**.
5. To remove an assembly file from the **References** dialog box, select it, and then click **Remove** .
6. When you are finished adding and removing references, click **OK** .

Finding Code in the Script Editor

You can quickly locate code in the script editor by using the **Find** operation.

To find code in the Script Editor

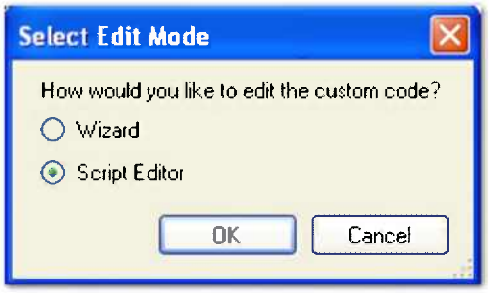
1. If the **Script Editor** window is not open, complete the procedure under "Opening the Script Editor" on page 296.
2. In the **Find** box, enter the code or character you want to find.
3. Press **Enter** to initiate the search. The code or character is selected in the **Script Editor**window.
4. You can click **Find Next** or **Find Previous** on the toolbar to navigate to instances of your specified code or character.

Modifying Exports with the Script Editor

After you have initially configured exports with the Custom Code Generator Wizard, you can use the **Script Editor** to modify export scripts. (See "Exports" on page 306 for information about configuring PaperVision Capture exports.)

To modify exports with the Script Editor

1. After you have logged in to the **PaperVision Capture Administration Console**, expand **Entities**, and then expand *Entity Name*.
2. Click **Capture Jobs**. A listing of jobs appears on the right pane.
3. Select the job you want to edit, and then click **Edit Job** .
4. If necessary, click **Check Out Job** so you can edit it.
5. On the workspace of the **Job Definitions** window, double-click the **Custom Code** job step to display the **Properties** tab on the left pane.
6. On the **Properties** tab, expand **Custom Code Events (Step Level)**.
7. Click **Step Executing**, and then click the ellipsis button to open the **Select Edit Mode** dialog box.



Select Edit Mode

8. Select **Script Editor**, and then click **OK**. The resulting export script appears in the **Script Editor**.

Modifying Export Constants

From the **Script Editor**, you can modify export scripts that you previously created with the Custom Code Generator Wizard. (See "Modifying Exports with the Script Editor" on page 301 for more information.) On the OCR tab, for example, you can change the OCR_CONVERTER_CODE constant in the **Script Editor** so that PDF searchable images are exported (for Nuance Full-Text OCR). To modify the constant, the following line in the XML script would read:

```
private const string OCR_CONVERTER_CODE = "PDFImageOnText";
```

NOTE: For a list of converter codes, see the **PVCaptureBatchAPI.chm** help file's **PVBatch.TryGetOCRFiles Method** topic found within the **Docs** directory where PaperVision Capture is installed.

In another scenario, you can use full-text OCR data from another job step by modifying the OCR_JOB_STEP_NAME constant. This is completed by entering the name of the step between the quotes (for example, "Nuance Full-Text OCR" or "Open Text Full-Text OCR").

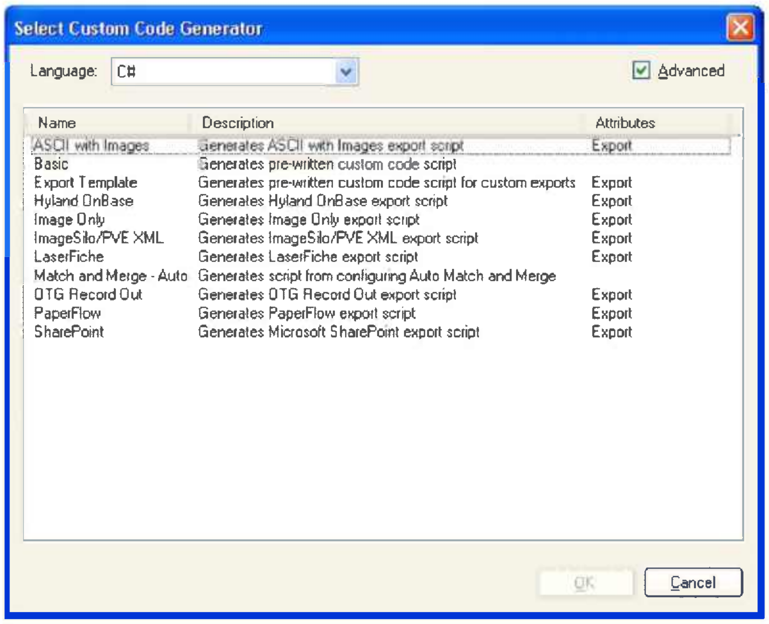
Match and Merge Wizard

The **Match and Merge - Auto** generator launches the **Match and Merge Wizard** where you configure the connection properties, field mapping, and optional Match and Merge settings.

NOTE: Ensure that the lookup table and columns for the database have been configured and indexes have been defined before launching the Match and Merge Wizard.

To Select the Match and Merge Generator

1. After you have logged in to the **PaperVision Capture Administration Console**, expand **Entities**, and then expand *Entity Name*.
2. Click **Capture Jobs**. A listing of jobs appears on the right pane.
3. Select the job you want to edit, and then click **Edit Job** .
4. If necessary, click **Check Out Job** so you can edit it.
5. On the workspace of the **Job Definitions** window, double-click the **Custom Code** job step to display the **Properties** tab on the left pane.
6. On the **Properties** tab, expand **Custom Code Events (Step Level)**.
7. Click **Step Executing**, and then click the ellipsis button to open the **Select Custom Code Generator** dialog box.



Name	Description	Attributes
ASCII with Images	Generates ASCII with Images export script	Export
Basic	Generates pre-written custom code script	
Export Template	Generates pre-written custom code script for custom exports	Export
Hyland OnBase	Generates Hyland OnBase export script	Export
Image Only	Generates Image Only export script	Export
ImageSilo/PVE XML	Generates ImageSilo/PVE XML export script	Export
LaserFiche	Generates LaserFiche export script	Export
Match and Merge - Auto	Generates script from configuring Auto Match and Merge	
OTG Record Out	Generates OTG Record Out export script	Export
PaperFlow	Generates PaperFlow export script	Export
SharePoint	Generates Microsoft SharePoint export script	Export

Select Custom Code Generator

NOTE: To remove existing custom code, on the **Properties** tab, expand **Custom Code Events [Step Level]**. Right-click **Step Executing**, and then click **Reset** . Additionally, to prevent the **Select Scripting Language** dialog box from appearing each time you configure custom code, select **Suppress this dialog when creating new custom code**.

---

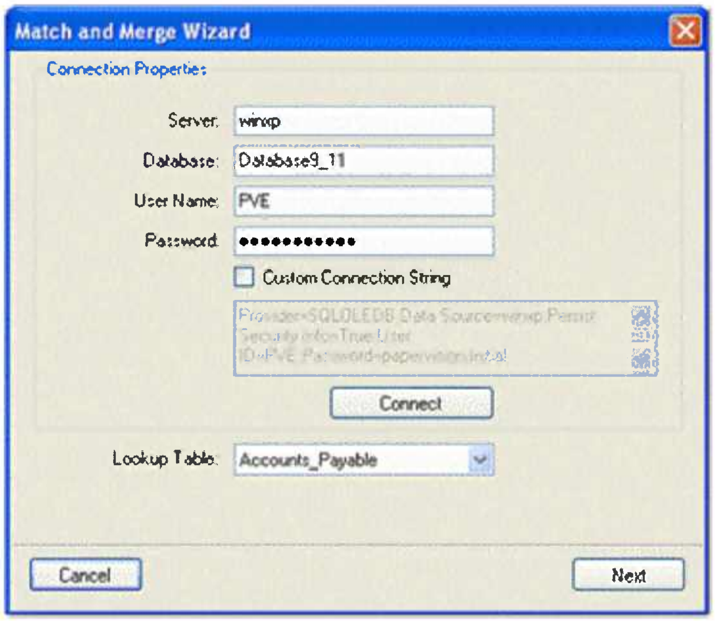
8. From the **Language** list, select the programming language you want to use. You can select **C#** or **Visual Basic**.
9. Double-click **Match and Merge - Auto** to open the **Match and Merge Wizard**.

Configuring the Match and Merge Wizard

The **Connection Properties** area appears when you open the **Match and Merge Wizard**. You can configure the database connection properties including the database server and name, user name and password, and database lookup table.

To configure the Match and Merge Wizard

1. If the **Match and Merge Wizard** is not open, complete the procedure under "To Select the Match and Merge Generator" on page 302.



Server: winxp
Database: Database9_11
User Name: PVE
Password: ●●●●●●●●●●●
☐ Custom Connection String
Provider=SQLOLEDB;Data Source=winxp;Persist Security Info=True;User ID=PVE;Password=papervision;Initial
Lookup Table: Accounts_Payable

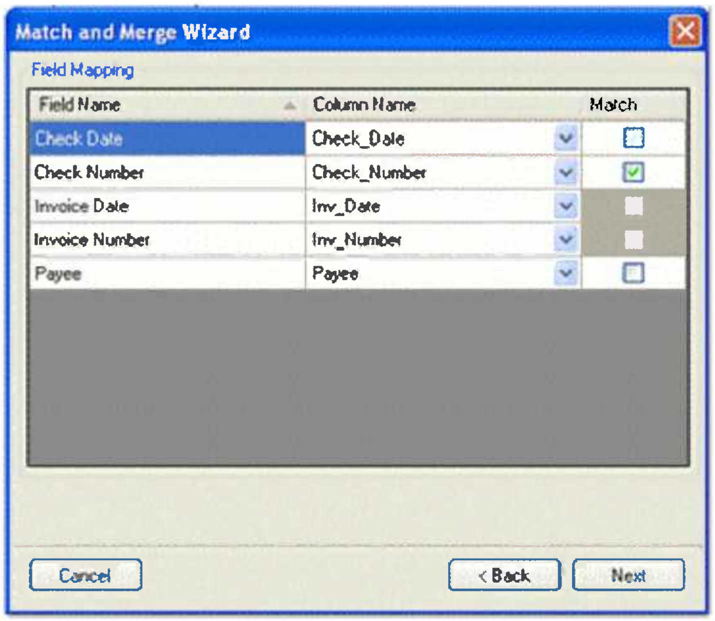
Connection Properties

2. In the **Server** box, type the database server where the match and merge process will be performed.
3. In the **Database** box, type the database name where the match and merge process will be performed.
4. In the **User Name** box, type the user name for the database server connection.
5. In the **Password** box, type the password for the database server connection.

NOTE: If you leave the **User Name** and **Password** fields blank, the database connection will use the Windows Authentication credentials. Entering a user name and password for the database will supersede the Windows Authentication credentials.

---

6. If you want to use a custom connection string, select **Custom Connection String**, and then type the connection information in the box below the option.
7. Click **Connect** to test the connection to the database. After you have connected to the database, values appear in the **Lookup Table** list.
8. From the **Lookup Table** list, select the database table used for lookups.
9. Click **Next**. The **Field Mapping** area appears.



Field Name	Column Name	Match
Check Date	Check_Date	☐
Check Number	Check_Number	✓
Invoice Date	Inv_Date	▨
Invoice Number	Inv_Number	▨
Payee	Payee	☐

Field Mapping

10. The **Field Mapping** area lets you match the columns in the database to the field names (indexes) that you defined. Click the **Column Name** list to select the database column name that will match the associated field name. If one of the index fields should not be matched, do not map it to the Column Name. When the operator executes the Merge Index Values command, only the mapped fields will be populated in the Index Manager.

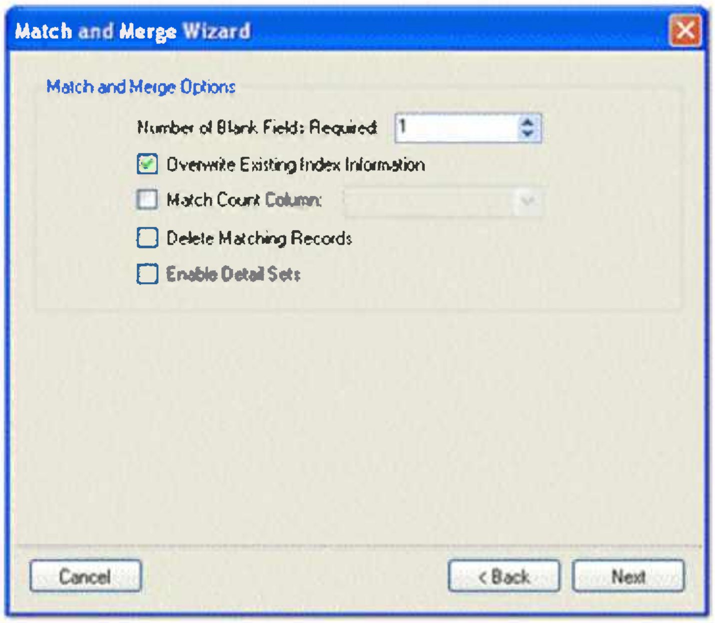
NOTE: Field names are synonymous with indexes that have been defined.

---

11. After selecting the column names, click the **Match** check box(es). Detail fields are indicated by shaded **Match** columns and cannot be selected to match.
    - In the example above, the Check Number index value, entered by the operator, will be matched with the corresponding Check_Number column in the database.

- Once the operator executes the Merge Index Values command, the corresponding Check Date, Invoice Date, Invoice Number, and Payee are populated in the Operator Console Index Manager.
- If the operator does not know the exact index value during hand-key indexing, the operator can insert wildcard characters to perform a partial search against a database. For example, the operator can insert the percent sign (%) to specify any number of unknown characters to search for in a SQL, Sybase, or Oracle database; the operator can insert the asterisk (*) to specify any number of unknown characters to search for within a Microsoft Access database.

12. Click **Next**. The **Match and Merge Options** area appears.



Number of Blank Fields Required: 1
[✓] Overwrite Existing Index Information
[ ] Match Count Column:
[ ] Delete Matching Records
[ ] Enable Detail Sets

Match and Merge Options

13. The **Match and Merge Options** area contains additional parameters that define the match and merge process. In the **Number of Blank Fields Required** box, type or select the number of fields that must be blank for PaperVision Capture to attempt to match during the custom code execution.
    - For example, you set the **Number of Blank Fields Required** to a value of **2**. If only one field is left blank before the match and merge process is run, then PaperVision Capture will not match because at least two fields were not blank.
    - Valid values range from zero to the number of database columns that are defined. For example, if you have five database columns defined, you can enter a value from zero to five.
14. If you select **Overwrite Existing Index Information**, the match and merge values will overwrite the existing index entries already populated in the batch.
15. The **Match Count Column** setting applies only to integer data type columns in the database. Select the **Match Count Column** check box if the match count should increment in the database by one each time a match is encountered. If you enable this setting, choose the database column from the corresponding list.
16. Select **Delete Matching Records** to remove the matching record from the database once it is found during the match and merge process.

NOTE: You can enable only the **Match Count Column** or the **Delete Matching Records** setting, but not both.

17. For manual indexing, select **Enable Detail Sets** if the detail fields should be populated when the operator enters the index fields. See "Configuring Detail Sets" on page 59for more information.
    - If you do not select **Enable Detail Sets**, the operator is presented with a pick list of data that meets the index field criteria. The operator then selects the appropriate record, and the detail fields are populated according to the selected record.

    When you define a Custom Code step to run an automated Match and Merge process

    - If you select **Enable Detail Sets** , all detail fields are automatically populated (for example, if five rows of data meet your criteria, five detail sets are populated).
    - Conversely, if you do not select **Enable Detail Sets**, the detail fields populate with data from the first row of results.
18. Click **Next**, which opens the last screen of the wizard.
19. Click **Finish**, which opens the **Script Editor** where you can make changes to the code if necessary.
20. Click **OK**.

Matching and Merging with Text Files

If you are using custom code to match and merge index fields with a text file, you can control how data is handled in the lookup table. If the text file contains dates, currency, or decimal data, for example, you can manipulate how data is formatted by creating a schema information file (Schema.ini) and placing it in the same directory where the text file resides. If you do not define how date columns are handled, date values will be imported in the DateTime format. You can find information on how to create Schema.ini files on the Microsoft Software Developer's Network:

http://msdn.microsoft.com/en-us/library/ms709353(VS.85).aspx

Exports

PaperVision Capture provides a user interface for export definitions within the **Custom Code** step. Exports can subsequently be imported into ImageSilo or PaperVision Enterprise (ImageSilo/PVE XML), PaperFlow (PaperFlow.xml), and other systems. If you have modified an export script in PaperVision Capture R72 or earlier, the Exports library is located in **Digitech Systems\PaperVision Capture\Library\Exports** where PaperVision Capture was installed. If you have not modified an export script in R72 or earlier, or you are initially installing PaperVision Capture R73, the Exports library will not exist since exports are configured directly in the user interface.

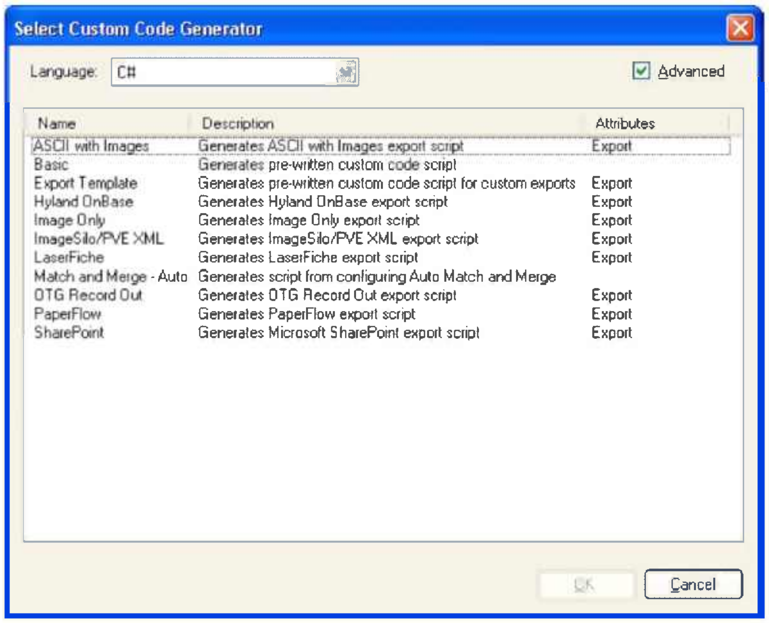
As exports are run, they are appended to the first available destination folder based on the sequence number and maximum export size (as defined by the **MAX_EXPORT_SIZE** script constant). When the maximum export size is reached, exports are appended to the next available folder. If two or more automated processes attempt to execute the same export (in the same destination folder), the first process places an exclusive lock on the folder. As a result, all subsequent processes will append exports to the next available folder. You can overwrite this method by specifying an automation server (in the export's **Configuration > Options** tab) that will process exports.

NOTE: If you are using multiple automation services and you specify multiple values for the **AUTOMATION_SERVER** script constant (or you do not specify a value for the **AUTOMATION_SERVER** script constant), your exported data may output to multiple folders (for example, data groups).

Export Definitions

PaperVision Capture exports contain specific definitions that you can configure. When you configure a PaperVision Capture export from the **Select Custom Code Generator** dialog box, properties specific to that export are shown, and default values that you can modify are included. Use the following procedure to open the **Select Custom Code Generator** dialog box where you can select the export that you want to configure.

1. After you have logged in to the **PaperVision Capture Administration Console**, expand **Entities**, and then expand *Entity Name*.
2. Click **Capture Jobs**. A listing of jobs appears on the right pane.
3. Select the job you want to edit, and then click **Edit Job** .
4. If necessary, click **Check Out Job** so you can edit it.
5. On the workspace of the **Job Definitions** window, double-click the **Custom Code** job step to display the **Properties** tab on the left pane.
6. On the **Properties** tab, expand **Custom Code Events (Step Level)**.
7. Click **Step Executing**, and then click the ellipsis button to open the **Select Custom Code Generator** dialog box. Each custom code generator and corresponding description are listed.



Language: C#

Advanced: ✓

Name	Description	Attributes
ASCII with Images	Generates ASCII with Images export script	Export
Basic	Generates pre-written custom code script	
Export Template	Generates pre-written custom code script for custom exports	Export
Hyland OnBase	Generates Hyland OnBase export script	Export
Image Only	Generates Image Only export script	Export
ImageSilo/PVE XML	Generates ImageSilo/PVE XML export script	Export
LaserFiche	Generates LaserFiche export script	Export
Match and Merge - Auto	Generates script from configuring Auto Match and Merge	
OTG Record Out	Generates OTG Record Out export script	Export
PaperFlow	Generates PaperFlow export script	Export
SharePoint	Generates Microsoft SharePoint export script	Export

Select Custom Code Generator

8. If you want to see only the generators that you can configure using the provided dialog boxes, rather than editing code in the **Script Editor**, then clear the **Advanced** check box.

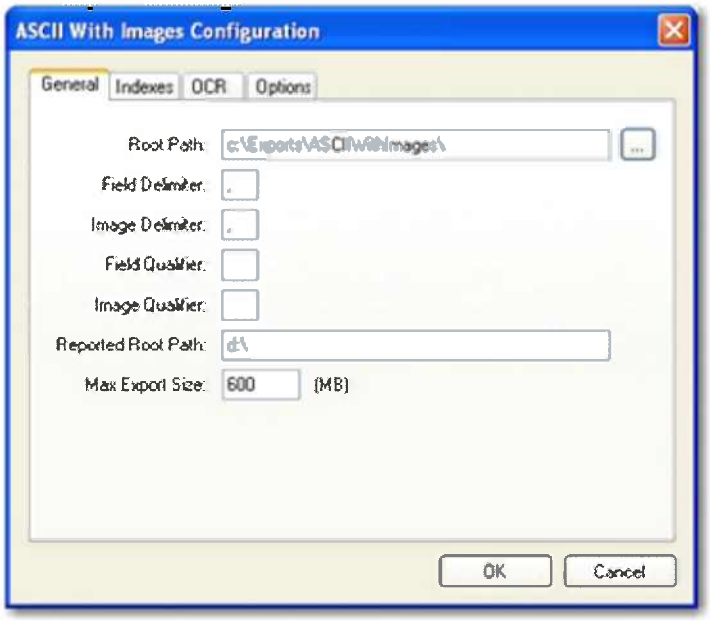
NOTE: To remove existing custom code, on the **Properties** tab, expand **Custom Code Events [Step Level]**. Right-click **Step Executing**, and then click **Reset**. Additionally, to prevent the **Select Scripting Language** dialog box from appearing each time you configure custom code, select **Suppress this dialog when creating new custom code**.

ASCII with Images

The **ASCII with Images** export creates an ASCII text file containing images that can be imported into other systems. The format of the file is completely customizable.

To configure the ASCII with Images export

1. If the **Select Custom Code Generator** dialog box is not open, complete the procedure under "Export Definitions" on page 307.
2. In the **Select Custom Code Generator** dialog box, double-click **ASCII with Images**. The **ASCII with Images Configuration** dialog box appears.



Root Path: c:\Exports\ASCIIwithImages\
Field Delimiter: ,
Image Delimiter: ,
Field Qualifier:
Image Qualifier:
Reported Root Path: d:\
Max Export Size: 600 (MB)

ASCII with Images Configuration - General

Default values that you can modify are provided for your reference, and the available options are specific to the generator you selected. In addition, you can browse to the appropriate directories instead of manually entering file paths.

3. Assign the appropriate properties on the **General**, **Indexes**, **OCR**, and **Options** tabs. Descriptions for constant values that appear in the resulting export script follow.

General

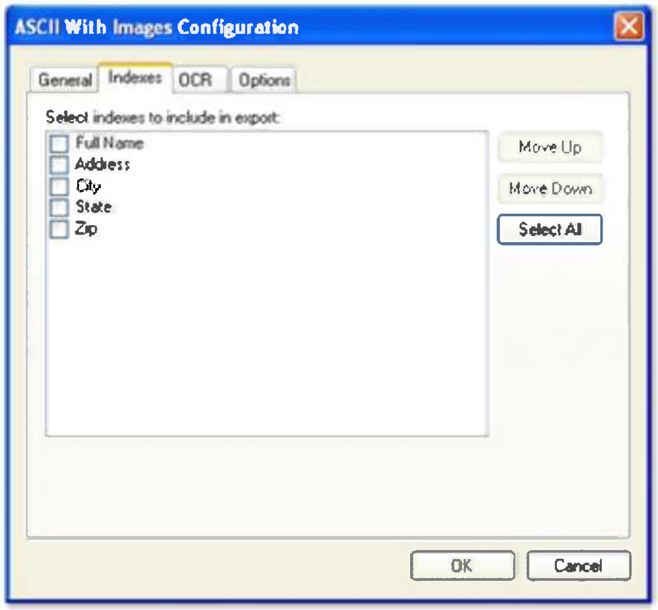
When you configure the properties on the **General** tab, the following constant values appear in the resulting export script.

- **ROOT_PATH**: This is the location where the exports will be created once the automation service processes the step.
- **FIELD_DELIMITER**: This customizable delimiter separates index values, page number/counts, and image sizes.
- **IMAGE_DELIMITER**: This customizable delimiter separates images when exporting using multiple-line indexing and converting to single-page images.
- **FIELD_QUALIFIER**: This constant contains the characters that surround the field values. By default, quotation marks will appear.
- **IMAGE_QUALIFER**: This constant contains the characters that surround the image values. By default, quotation marks will appear.
- **REPORTED_ROOT_PATH**: The path referenced in the export file originates from this location, not the ROOT_PATH.
- **MAX_EXPORT_SIZE**: This constant indicates the maximum export file size in megabytes, which defaults to a value of **600**.

NOTE: If the Root Path is blank, the export is written to the directory where the application is installed (for example, C:\Program Files\Digitech Systems\PaperVision Capture). If the Reported Root Path is blank, the resulting export script displays a blank value for the **REPORTED_ROOT_PATH**.

Indexes

On the **Indexes** tab, you can specify the indexes that will appear in the export by selecting the check box next to the index. To include all of the indexes, click **Select All**. To remove all selections, click **Deselect All**. To change the order in which the indexes appear,select the index you want to move, and then click **Move Up** or **Move Down**.



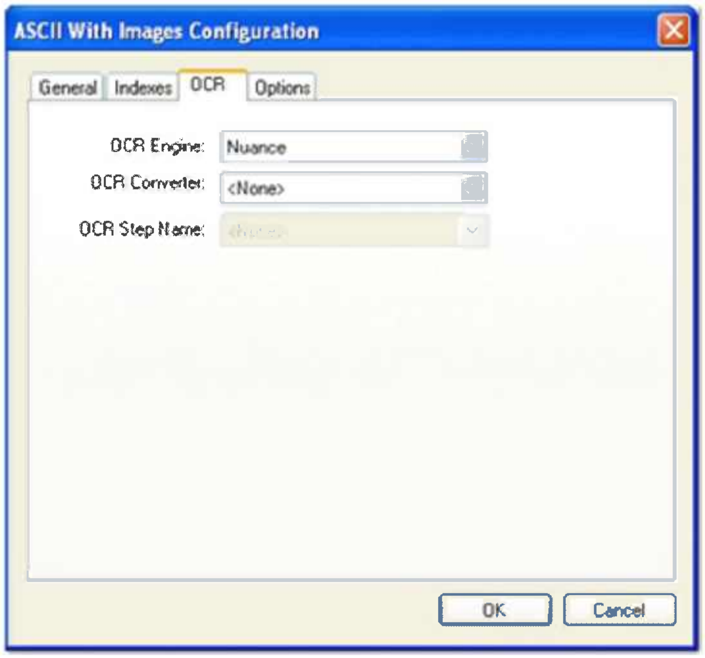
ASCII with Images Configuration - Indexes

To edit the indexes in the resulting export script, you can modify the following **INDICES_TO_INCLUDE** constant.

- **INDICES_TO_INCLUDE**: This constant determines what index values are included in the export file. In the resulting script, you can enter the name of the index value(s) between quotation marks, and separate each index value with a comma. If you leave this array blank, no indices are included.

OCR

When you configure the properties on the **OCR** tab, you can modify constant values that appear in the resulting export script. Descriptions for each constant value follow.



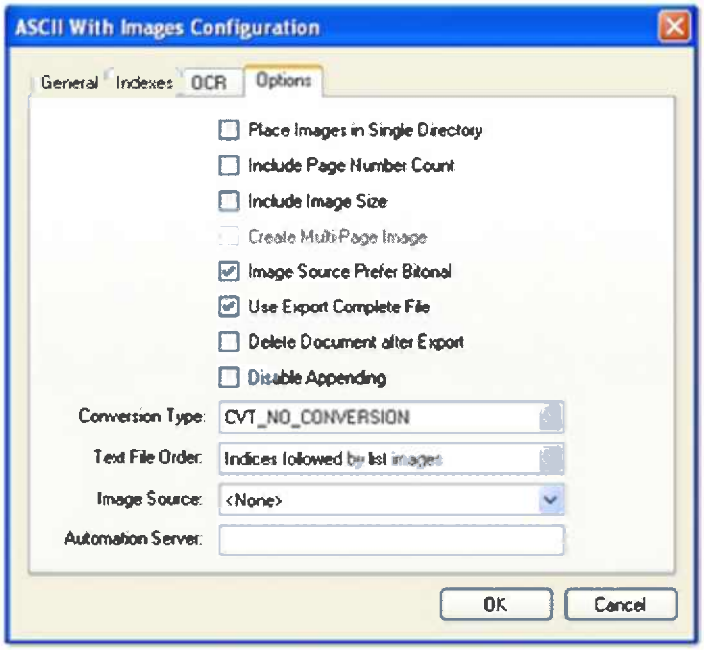
OCR Engine: Nuance
OCR Converter: <None>
OCR Step Name:

ASCII with Images Configuration - OCR

- **OCR_ENGINE**: This constant specifies the OCR engine (Nuance or Open Text) that processes OCR data for the export.
- **OCR_CONVERTER_CODE**: This constant specifies the OCR converter code, such as PDF, Text, XML, etc., whose output format is used to export full-text data. When no value is defined (the default setting), both images and associated full-text data are exported.
- **OCR_JOB_STEP_NAME**: This constant specifies the job step whose full-text data are used for the export. No value is defined by default, so full-text data from the current job step are used for the export.

Options

When you configure properties on the **Options** tab, you can modify constant values that appear in the resulting export script. Descriptions for each constant value follow.



- [ ] Place Images in Single Directory
- [ ] Include Page Number Count
- [ ] Include Image Size
- [ ] Create Multi-Page Image
- [x] Image Source Prefer Bitonal
- [x] Use Export Complete File
- [ ] Delete Document after Export
- [ ] Disable Appending

Conversion Type:	CVT_NO_CONVERSION
Text File Order:	Indices followed by list images
Image Source:	<None>
Automation Server:	

ASCII with Images Configuration - Options

- **PLACE_IMAGES_IN_SINGLE_DIR**: If set to **False**, the images are placed in subdirectories at the **ROOT_PATH** (maximum of 1000 images per directory). If set to **True**, the images are placed directly in the **ROOT_PATH** folder.
- **INCLUDE_PAGE_NUMBER_COUNT**: This determines whether the page number or page count of the document should be added as an additional field in the export. If set to **False**, when exporting in a multi-line format and creating single-page images, this value will match the page number of the document. If set to **True**, the value will match the total number of pages in the document.
- **INCLUDE_IMAGE_SIZE**: This constant determines whether the image file size is added as an additional field in the export. If set to **True**, this value will match the image size referenced on that line of the export file when exporting using a multi-line format and creating single-page images. If set to **False**, this value will match the size of the first page in the document.
- **CREATE_MULTI_PAGE_IMAGE**: Used in conjunction with **CONVERSION_TYPE**, this constant determines whether exported images are single-page or multi-page.
- **IMG_SRC_PREFER_BITONAL_IMAGES**: This constant is applicable to dual-stream scanners and determines whether to export bitonal or color images. When set to **True**, which is the default setting, bitonal images are exported.
- **USE_EXPORT_COMPLETE_FILE:** This constant, set to **True** by default, generates an "export.complete" file once an export has reached its maximum file size, so data will no longer be appended to the export. When set to **False**, the "export.complete" file is not generated, so data may be appended to export folders that have not reached their maximum size. If you set this constant to **False**, for example, and the following four folders are available under the **ROOT_PATH** with the **MAX_EXPORT_SIZE** defined as 600 MB:

  1. Folder_1: 600 MB
  2. Folder_2: 400 MB

3. Folder_3: 600 MB
4. Folder_4: 100 MB

Since the maximum export size has been reached in Folder_1, Folder_2 will be used as the export folder, and the "export.complete" file will not be generated.

TIP: By default, the lockedPath (working directory) for any export is returned by calling GetNextLockedPath(). If an export should contain this constant value, the following line in the **Script Editor**, which is available to use in all exports, can be changed to: `lockedPath = GetNextLockedpath(root, MAX_EXPORT_SIZE, true)`.

- **DELETE_DOCUMENT_AFTER_EXPORT**: This constant specifies whether documents are deleted after they have been exported (set to **False** by default).
- **DISABLE_APPENDING**: This constant is set to **False** by default. When set to **True**, exported images will not be appended to export folders whose maximum file sizes have not been reached.
- **CONVERSION_TYPE**: This constant determines the type of image file created during the export. The default value, **CVT_NO_CONVERSION**, does not convert images during the export. If exporting to a format that supports both single and multi-page images, you must set the **CREATE_MULTI_PAGE_ IMAGE** constant to **True** if you want to create multi-page images; otherwise single page images will result. For example, if you set this to **CVT_TIFF_G4_MEDJPG**, a TIFF image is created during the export. If the source image is binary, it will create a TIFF using Group 4 compression; if the source image is color (JPG or BMP), it will create a TIFF using Medium JPEG compression. (See "Enumerations" on page 291 for more information.)
- **TEXT_FILE_ORDER**: This constant determines how the export file is formatted. You can select from the following options.
  - **IndicesFollowedByListImages**: This option creates a single row for each document with indexes listed first, followed by image files.
  - **ListImagesFollowedByIndices**: This option creates a single row for each document with images listed first, followed by the index values.
  - **MultiLineIndicesFollowedBySingleImage**: This option creates one row of index values for every image created during the export. If multiple image files are created for a single document, multiple rows of identical index values will be created, each referencing a different page of the document. This will be formatted with index values followed by images.
  - **MultiLineImagesFollowedByIndices**: One row of index values for every image created during the export. If multiple image files are created for a single document, multiple rows of identical index values will be created, each referencing a different page of the document. This will be formatted with images followed by index values.
- **IMG_SRC_JOB_STEP_NAME**: This constant determines the job step from which images are used for the export. The default selection, **<None>**, uses the most recent image prior to exporting. To use images from another job step, select the name of the step from the **Image Source** list.
- **AUTOMATION_SERVER**: If you specify an automation server (in the **MACHINENAME_INSTANCE** format), your specified server will process exports one at a time in the **ROOT_PATH** location. When one or

more automation servers are specified, separate folders may be created for multiple exports that are processed simultaneously.

If you leave the **Automation Server** field blank during export configuration, all servers will be used to process the exports. If you are using multiple automation servers, separate each server name with a comma. You can enter wildcard characters in this field and values that you enter are not case-sensitive.

NOTE: If you are using multiple automation services and you specify multiple values for the **AUTOMATION_SERVER** constant (or, if using multiple automation services and you do not specify a value for the **AUTOMATION_SERVER** constant), your exported data may output to multiple folders (for example, data groups).

Hyland OnBase

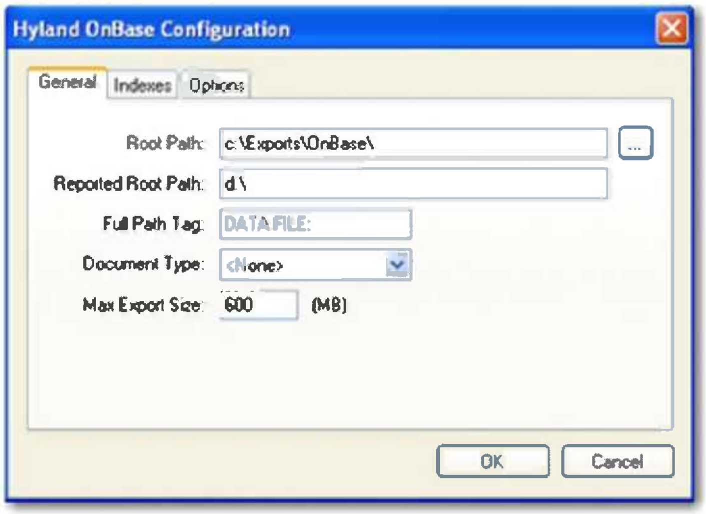
The **Hyland OnBase** export creates an ASCII text file and single-page TIFF images that can be imported into the Hyland OnBase system. The following settings must be configured in the Hyland OnBase system prior to importing any PaperVision Capture exports.

1. The Document Import Processor separator must be set to **New Line**.
2. The field delimiter must be set to **None**.
3. The field type must be set to **Tagged Fields.**

NOTE: If the PaperVision Capture job contains dates, the Hyland OnBase date format settings must match the date field format for that job.

To configure the Hyland OnBase export

1. If the **Select Custom Code Generator** dialog box is not open, complete the procedure under "Export Definitions" on page 307.
2. In the **Select Custom Code Generator** dialog box, double-click **Hyland OnBase**. The **Hyland OnBase Configuration** dialog box appears.



Root Path: c:\Exports\OnBase\
Reported Root Path: d:\
Full Path Tag: DATA FILE:
Document Type: <None>
Max Export Size: 600 (MB)

Hyland OnBase Configuration - General

Default values that you can modify are provided for your reference, and the available options are specific to the generator you selected. In addition, you can browse to the appropriate directories instead of manually entering file paths.

3. Assign the appropriate properties on the **General**, **Indexes**, and **Options** tabs. Descriptions for constant values that appear in the resulting export script follow.

General

When you configure the properties on the **General** tab, the following constant values appear in the resulting export script.

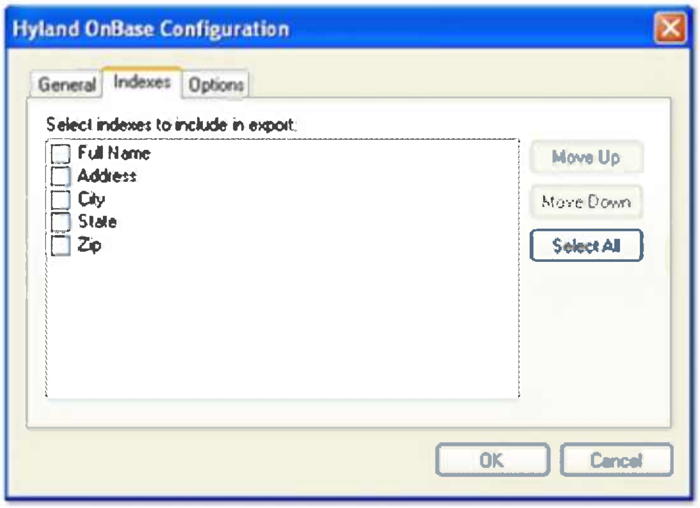
- **ROOT_PATH**: This is the location where the exports are created after the automation service processes the step.
- **REPORTED_ROOT_PATH**: The path referenced in the export file originates from this location, not the **ROOT_PATH**.

NOTE: If the **Root Path** box is blank, the export is written to the directory where the application is installed (for example, C:\Program Files\Digitech Systems\PaperVision Capture). If the **Reported Root Path** box is blank, the resulting export script displays a blank value for the **REPORTED_ROOT_PATH** constant.

- **FULL_PATH_TAG**: This tag precedes the **REPORTED_ROOT_PATH** in the export file.
- **DOCUMENT_TYPE**: This constant indicates index mapping to the document type.
- **MAX_EXPORT-SIZE**: This constant indicates the maximum export file size in megabytes. The default value is **600**.

Indexes

On the **Indexes** tab, you can specify the indexes that will appear in the export by selecting the check box next to the index. To include all of the indexes, click **Select All**. To remove all selections, click **Deselect All**. To change the order in which the indexes appear,select the index you want to move, and then click **Move Up** or **Move Down**.



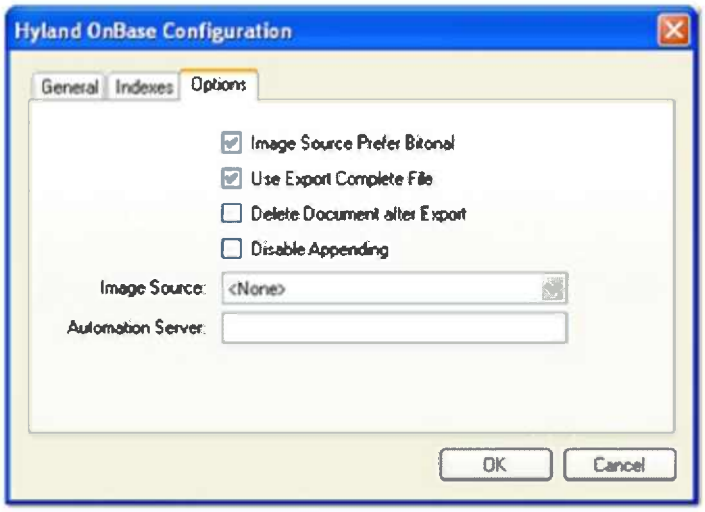
Hyland OnBase Configuration - Indexes

To edit the indexes in the resulting export script, you can modify the following **INDICES_TO_INCLUDE** constant.

- **INDICES_TO_INCLUDE**: This constant determines what index values are included in the export file. In the resulting script, you can enter the name of the index value(s) between quotation marks, and separate each index value with a comma. If you leave this array blank, no indices are included.

Options

When you configure properties on the **Options** tab, you can modify constant values that appear in the resulting export script. Descriptions for each constant value follow.



- [✓] Image Source Prefer Bitonal
- [✓] Use Export Complete File
- [ ] Delete Document after Export
- [ ] Disable Appending

Image Source: <None>
Automation Server:

Hyland OnBase Configuration - Options

- **IMG_SRC_PREFER_BITONAL_IMAGES**: This constant is applicable to dual-stream scanners and determines whether to export bitonal or color images. When set to **True**, which is the default setting, bitonal images are exported.

- **USE_EXPORT_COMPLETE_FILE:** This constant, set to **True** by default, generates an "export.complete" file once an export has reached its maximum file size, so data will no longer be appended to the export. When set to **False**, the "export.complete" file is not generated, so data may be appended to export folders that have not reached their maximum size. If you set this constant to **False**, for example, and the following four folders are available under the **ROOT_PATH** with the **MAX_EXPORT_SIZE** defined as 600 MB:

  1. Folder_1: 600 MB
  2. Folder_2: 400 MB
  3. Folder_3: 600 MB
  4. Folder_4: 100 MB

  Since the maximum export size has been reached in Folder_1, Folder_2 will be used as the export folder, and the "export.complete" file will not be generated.

TIP: By default, the lockedPath (working directory) for any export is returned by calling GetNextLockedPath(). If an export should contain this constant value, the following line in the **Script Editor**, which is available to use in all exports, can be changed to: `lockedPath = GetNextLockedpath(root, MAX_EXPORT_SIZE, true)`.

- **DELETE_DOCUMENT_AFTER_EXPORT**: This constant specifies whether documents are deleted after they have been exported, and is set to **False** by default.
- **DISABLE_APPENDING**: This constant is set to **False** by default. When set to **True**, exported images will not be appended to export folders that have not reached the maximum file size.
- **IMG_SRC_JOB_STEP_NAME**: This constant determines the job step from which images are used for the export. The default selection, **<None>**, uses the most recent image prior to exporting. To use images from another job step, select the name of the step from the **Image Source** list.
- **AUTOMATION_SERVER**: If you specify an automation server (in the **MACHINENAME_INSTANCE** format), your specified server will process exports one at a time in the **ROOT_PATH** location. When one or more automation servers are specified, separate folders may be created for multiple exports that are processed simultaneously.

  If you leave the **Automation Server** box blank during export configuration, all servers will be used to process the exports. If you are using multiple automation servers, separate each server name with a comma. You can enter wildcard characters in this box, and values that you enter are not case-sensitive.

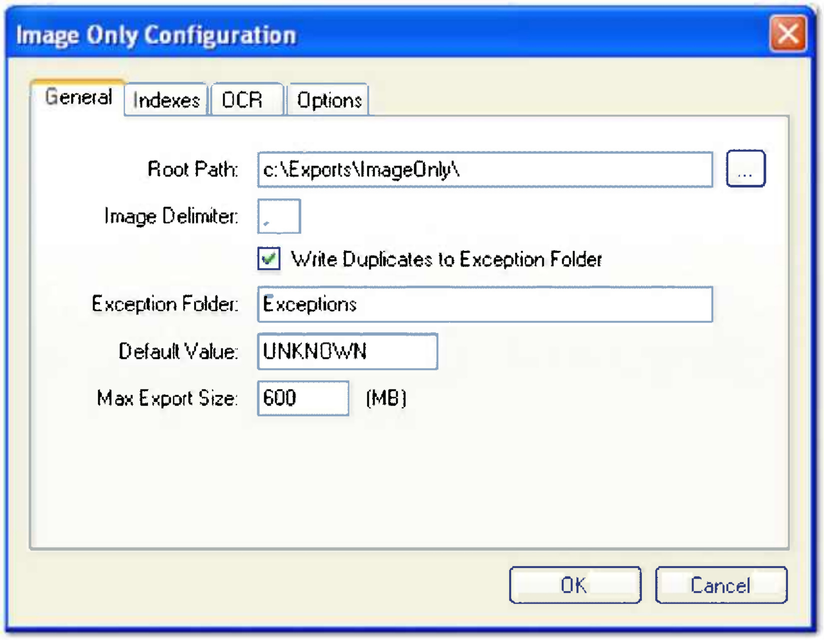
  NOTE: If you are using multiple automation services and you specify multiple values for the **AUTOMATION_SERVER** constant (or, if using multiple automation services and you do not specify a value for the **AUTOMATION_SERVER** constant), your exported data may output to multiple folders (for example, data groups).

Image Only

The **Image Only** export creates image files that are named after a specific index field. Any subdirectories containing those image files are named after other index fields (optional). Single-page image file formats are named with an "-*X*" at the end of the file name where "*X*" denotes the page number.

To configure the Image Only export

1. If the **Select Custom Code Generator** dialog box is not open, complete the procedure under "Export Definitions" on page 307.
2. In the **Select Custom Code Generator** dialog box, double-click **Image Only**. The **Image Only Configuration** dialog box appears.



Root Path: c:\Exports\ImageOnly\
Image Delimiter: -
[✓] Write Duplicates to Exception Folder
Exception Folder: Exceptions
Default Value: UNKNOWN
Max Export Size: 600 (MB)

Image Only Configuration

Default values that you can modify are provided for your reference, and the available options are specific to the generator you selected. In addition, you can browse to the appropriate directories instead of manually entering file paths.

3. Assign the appropriate properties on the **General**, **Indexes**, **OCR**, and **Options** tabs. Descriptions for constant values that appear in the resulting export script follow.

General

When you configure the properties on the **General** tab, the following constant values appear in the resulting export script.

- **ROOT_PATH**: This is the location where the exports are created after the automation service processes the step.

NOTE: If the **Root Path** box is blank, the export is written to the directory where the application is installed (for example, C:\Program Files\Digitech Systems\PaperVision Capture).

- **IMAGE_DELIMITER**: This constant specifies the character that will separate the image file name if multiple index values are combined to create the image file name.
- **WRITE_DUPLICATES_TO_EXCEPTION_FOLDER**: If duplicate files are created in the same directory during the export and this value is set to **False**, PaperVision Capture will not copy the duplicate files into the **EXCEPTION_FOLDER** directory. If this value is set to **True**, duplicate files are placed in the **EXCEPTION_FOLDER** instead.

NOTE: Files in the **EXCEPTION_FOLDER** directory display with "_#" appended to the file name, where "#" is a unique incrementing number starting with "**1**." This appending process prevents the exception files from being overwritten in the directory.

- **EXCEPTION_FOLDER**: If the **WRITE_DUPLICATES_TO_EXCEPTION_FOLDER** value is set to **True**, and multiple images with the same file name are created in the same directory, duplicates will be placed in this folder at the **ROOT_PATH** instead of overwriting the existing file of that name.
- **DEFAULT_VALUE**: As the export script executes, invalid characters are stripped from index fields, possibly resulting in blank fields. By default, the resulting **DEFAULT_VALUE** for these blank fields is defined as "**UNKNOWN**."
- **MAX_EXPORT-SIZE**: This constant indicates the maximum export file size in megabytes. The default value is **600**.

Indexes

On the **Indexes** tab, you can specify the indexes that will appear in the export by selecting the check box next to the index. To include all of the indexes, click **Select All**. To remove all selections, click **Deselect All**. To change the order in which the indexes appear,select the index you want to move, and then click **Move Up** or **Move Down**.

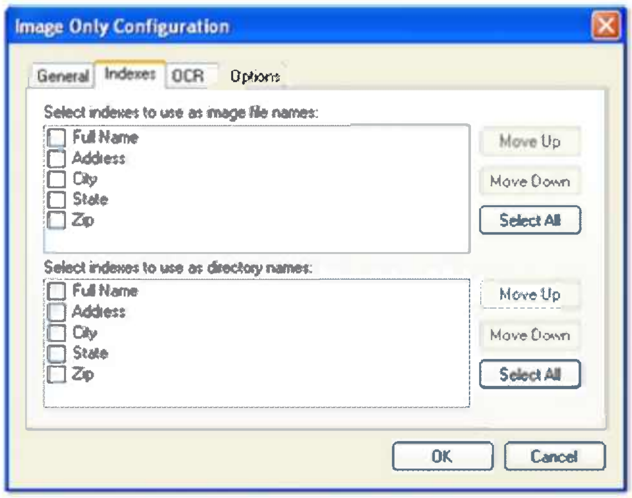


Image Only Configuration - Indexes

To edit the indexes in the resulting export script, you can modify the following constants.

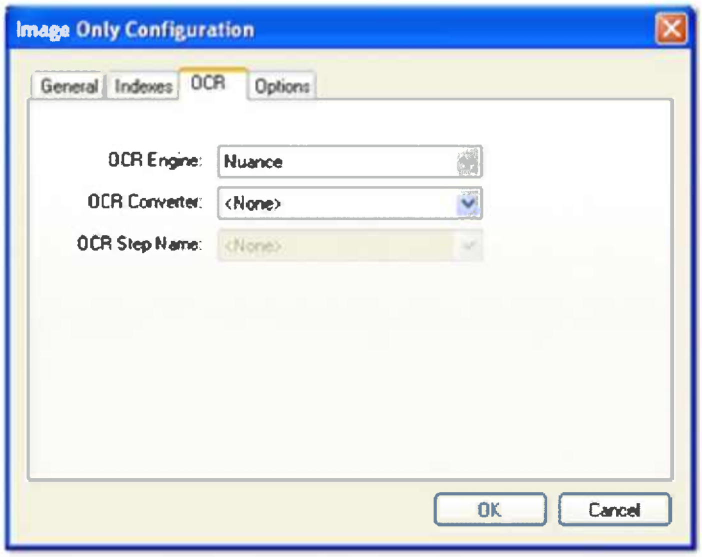
- **IMAGE_INDICES**: Images created during the export are named based on the index fields mapped in the **IMAGE_INDICES** field. If multiple index fields are mapped, the specified **IMAGE_DELIMITER** value is used to separate the fields in the name of the file. If no fields are mapped, a standard eight-digit incrementing file name is used.

NOTE: Image file names are pulled from a single index field configured in the **IMAGE_INDICES** field. Any subdirectories are also configured similarly. Index fields should not contain characters that create invalid file or directory names.

- **FOLDER_INDICES**: Images created during the export are placed in named folders based on the **FOLDER_INDICES**. The first mapped field will match the first folder, the second mapped field will match the name of the subfolder, and so on. If no fields are mapped, the images are placed directly in the **ROOT_PATH**.

OCR

When you configure the properties on the **OCR** tab, you can modify constant values that appear in the resulting export script. Descriptions for each constant value follow.



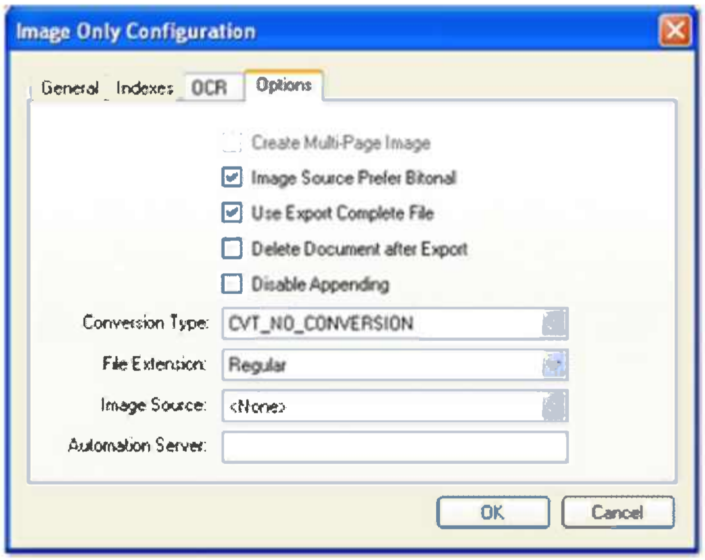
OCR Engine: Nuance
OCR Converter: <None>
OCR Step Name: <None>

Image Only Configuration - OCR

- **OCR_ENGINE**: This constant specifies the OCR engine (Nuance or Open Text) that processes OCR data for the export.
- **OCR_CONVERTER_CODE**: This constant specifies the OCR converter code, such as PDF, Text, XML, etc., whose output format is used to export full-text data. When no value is defined (the default setting), both images and associated full-text data are exported.
- **OCR_JOB_STEP_NAME**: This constant specifies the job step whose full-text data are used for the export. No value is defined by default, so full-text data from the current job step are used for the export.

Options

When you configure properties on the **Options** tab, you can modify constant values that appear in the resulting export script. Descriptions for each constant value follow.



☐ Create Multi-Page Image
☑ Image Source Prefer Bitonal
☑ Use Export Complete File
☐ Delete Document after Export
☐ Disable Appending

Conversion Type: CVT_NO_CONVERSION
File Extension: Regular
Image Source: <None>
Automation Server:

Image Only Configuration - Options

- **CREATE_MULTI_PAGE_IMAGE**: Used in conjunction with **CONVERSION_TYPE**, this constant determines whether exported images are single-page or multi-page.
- **IMG_SRC_PREFER_BITONAL_IMAGES**: This constant is applicable to dual-stream scanners and determines whether to export bitonal or color images. When set to **True**, which is the default setting, bitonal images are exported.
- **USE_EXPORT_COMPLETE_FILE:** This constant, set to **True** by default, generates an "export.complete" file once an export has reached its maximum file size, so data will no longer be appended to the export. When set to **False**, the "export.complete" file is not generated, so data may be appended to export folders that have not reached their maximum size. If you set this constant to **False**, for example, and the following four folders are available under the **ROOT_PATH** with the **MAX_EXPORT_SIZE** defined as 600 MB:

  1. Folder_1: 600 MB
  2. Folder_2: 400 MB
  3. Folder_3: 600 MB
  4. Folder_4: 100 MB

  Since the maximum export size has been reached in Folder_1, Folder_2 will be used as the export folder, and the "export.complete" file will not be generated.

TIP: By default, the lockedPath (working directory) for any export is returned by calling GetNextLockedPath(). If an export should contain this constant value, the following line in the **Script Editor**, which is available to use in all exports, can be changed to: `lockedPath = GetNextLockedpath(root, MAX_EXPORT_SIZE, true)`.

- **DELETE_DOCUMENT_AFTER_EXPORT**: This constant specifies whether documents are deleted after they have been exported (set to **False** by default).
- **DISABLE_APPENDING**: This constant is set to **False** by default. When set to **True**, exported images will not be appended to export folders whose maximum file sizes have not been reached.
- **CONVERSION_TYPE**: This constant determines the type of image file created during the export. The default value, **CVT_NO_CONVERSION**, does not convert images during the export. If exporting to a format that supports both single and multi-page images, you must set the **CREATE_MULTI_PAGE_ IMAGE** constant to **True** if you want to create multi-page images; otherwise single page images will result. For example, if you set this to **CVT_TIFF_G4_MEDJPG**, a TIFF image is created during the export. If the source image is binary, it will create a TIFF using Group 4 compression; if the source image is color (JPG or BMP), it will create a TIFF using Medium JPEG compression. (For file types you can use for conversion during the export process, see "Enumerations" on page 291 for more information.)
- **FILE_EXTENSION**: This constant determines whether the file extension or page number will be assigned to the file type created during the export. You can choose one of the following options.

  Regular: This option uses the original file extension (for example, .tif, .jpg, etc.).

  PageNumberStartingZero: This option uses the page number for the file extension, starting with **0** (for example,**.0**, **.1**, and so on).

  PageNumberStartingOne: This option uses the page number for the file extension, starting with **1** (for example, **.1**, **.2**, and so on).

  PageNumberStartingZeroWithPadding: This option uses the page number for the file extension, starting with **000** (for example, **.000**, **.001**,and so on).

  PageNumberStartingOneWithPadding: This option uses the page number for the file extension, starting with **001** (for example, **.001**, **.002**, and so on).
- **IMG_SRC_JOB_STEP_NAME**: This constant determines the job step from which images are used for the export. The default selection, **<None>**, uses the most recent image prior to exporting. To use images from another job step, select the name of the step from the **Image Source** list.
- **AUTOMATION_SERVER**: If you specify an automation server (in the **MACHINENAME_INSTANCE** format), your specified server will process exports one at a time in the **ROOT_PATH** location. When one or more automation servers are specified, separate folders may be created for multiple exports that are processed simultaneously.

  If you leave the **Automation Server** box blank during export configuration, all servers will be used to process the exports. If you are using multiple automation servers, separate each server name with a comma. You can enter wildcard characters in this box, and values that you enter are not case-sensitive.

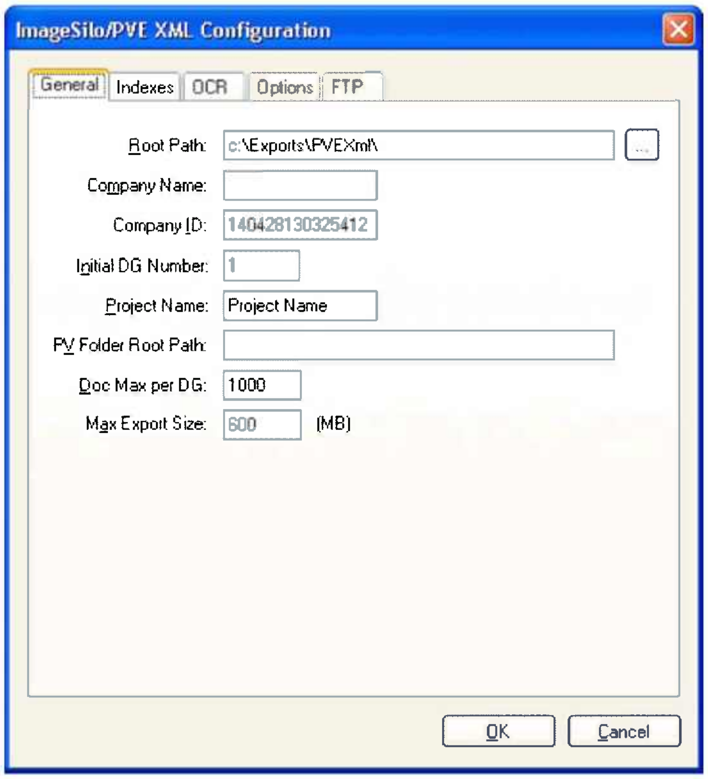
NOTE: If you are using multiple automation services and you specify multiple values for the **AUTOMATION_SERVER** constant (or, if using multiple automation services and you do not specify a value for the **AUTOMATION_SERVER** constant), your exported data may output to multiple folders (for example, data groups).

ImageSilo/PVE XML

The **ImageSilo/PVE XML** export creates an export that can be used to import batches into ImageSilo or PaperVision Enterprise.

To configure the ImageSilo/PVE XML export

1. If the **Select Custom Code Generator** dialog box is not open, complete the procedure under "Export Definitions" on page 307.
2. In the **Select Custom Code Generator** dialog box, double-click **ImageSilo/PVE XML**. The **ImageSilo/PVE XML Configuration** dialog box appears.



Root Path:	c:\Exports\PVEXml\
Company Name:	
Company ID:	140428130325412
Initial DG Number:	1
Project Name:	Project Name
PV Folder Root Path:	
Doc Max per DG:	1000
Max Export Size:	600 (MB)

ImageSilo/PVE XML Configuration - General

Default values that you can modify are provided for your reference, and the available options are specific to the generator you selected. In addition, you can browse to the appropriate directories instead of manually entering file paths.

3. Assign the appropriate properties on the **General**, **Indexes**, **OCR**, **Options** and **FTP** tabs. Descriptions for constant values that appear in the resulting export script follow.

General

When you configure the properties on the **General** tab, the following constant values appear in the resulting export script.

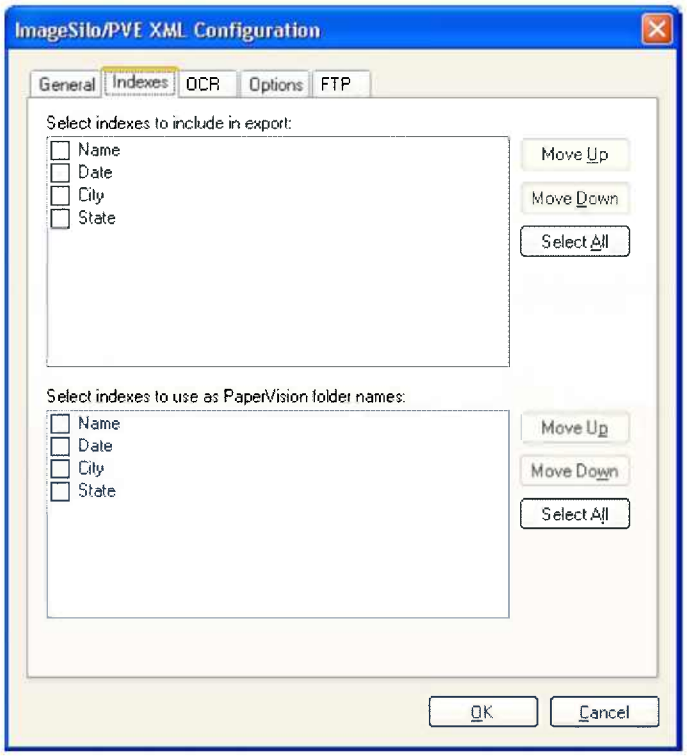
- **ROOT_PATH**: This is the location where the exports are created after the automation service processes the step.

NOTE: If the **Root Path** box is blank, the export is written to the directory where the application is installed (for example, C:\Program Files\Digitech Systems\PaperVision Capture).

- **COMPANY_NAME:** This constant is the name of your company or department and has a blank default value. The Company Name is required.
- **COMPANY_ID**: This constant is the ID of your company or department. The default value is set to the identifier, "yymmddhhnnssms".
- **INITIAL_DATA_GROUP_NUMBER**: This constant represents the initial Data Group number used by ImageSilo or PaperVision Enterprise. The default value is **1**.
- **PROJECT_NAME**: This constant indicates the name of your project. The default value is set to **Project Name**.
- **PV_FOLDER_ROOT_PATH**: This constant specifies the root path containing all folders (used in the Folder view in ImageSilo or PaperVision Enterprise). Type the root path between the quotes (for example, C:\\Exports\\PVEXml\\FolderRootPath\\).
- **DOCUMENT_MAX_PER_DATAGROUP**: This constant indicates the maximum number of documents per data group. The default value is **1000**, which is the recommended value for XML files.
- **MAX_EXPORT-SIZE**: This constant indicates the maximum export file size in megabytes. The default value is **600**.

Indexes

On the **Indexes** tab, you can specify the indexes that will appear in the export by selecting the check box next to the index. To include all of the indexes, click **Select All**. To remove all selections, click **Deselect All**. To change the order in which the indexes appear,select the index you want to move, and then click **Move Up** or **Move Down**.



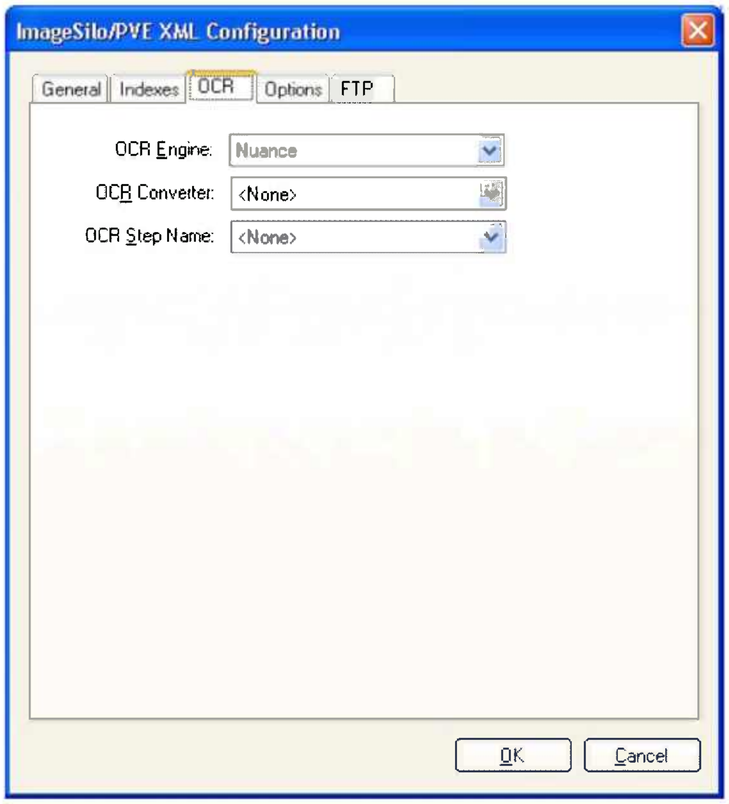
ImageSilo/PVE XML Configuration - Indexes

To edit the indexes in the resulting export script, you can modify the following constants.

- **INDICES_TO_INCLUDE**: This constant determines the index values included in the export file. To include all indices, leave the array blank.
- **PV_FOLDER_INDICES**: This constant determines the index value(s) representing each folder (used in the Folder view in ImageSilo or PaperVision Enterprise). If you leave the array blank, no index values will be included.

OCR

When you configure the properties on the **OCR** tab, you can modify constant values that appear in the resulting export script. Descriptions for each constant value follow.



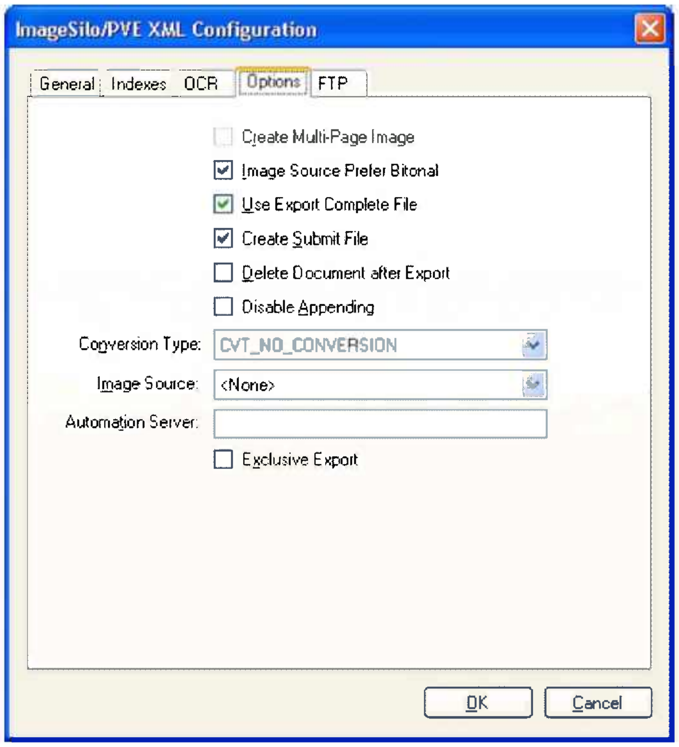
OCR Engine: Nuance
OCR Converter: <None>
OCR Step Name: <None>

ImageSilo/PVE XML Configuration - OCR

- **OCR_ENGINE**: This constant specifies the OCR engine (Nuance or Open Text) that processes OCR data for the export.
- **OCR_CONVERTER_CODE**: This constant specifies the OCR converter code, such as PDF, Text, XML, etc., whose output format is used to export full-text data. When no value is defined (the default setting), both images and associated full-text data are exported. If you select the PaperVision Full-Text OCR converter, only full-text data will be exported (associated images will not be exported).
- **OCR_JOB_STEP_NAME**: This constant specifies the job step whose full-text data are used for the export. No value is defined by default, so full-text data from the current job step are used for the export.

Options

When you configure properties on the **Options** tab, you can modify constant values that appear in the resulting export script. Descriptions for each constant value follow.



☐ Create Multi-Page Image
☑ Image Source Prefer Bitonal
☑ Use Export Complete File
☑ Create Submit File
☐ Delete Document after Export
☐ Disable Appending
Conversion Type: CVT_NO_CONVERSION
Image Source: <None>
Automation Server:
☐ Exclusive Export

ImageSilo/PVE XML Configuration - Options

- **CREATE_MULTI_PAGE_IMAGE**: Used in conjunction with **CONVERSION_TYPE**, this constant determines whether exported images are single-page or multi-page.
- **IMG_SRC_PREFER_BITONAL_IMAGES**: This constant is applicable to dual-stream scanners and determines whether to export bitonal or color images. When set to **True**, which is the default setting, bitonal images are exported.
- **USE_EXPORT_COMPLETE_FILE:** This constant, set to **True** by default, generates an "export.complete" file once an export has reached its maximum file size, so data will no longer be appended to the export. When set to **False**, the "export.complete" file is not generated, so data may be appended to export folders that have not reached their maximum size. If you set this constant to **False**, for example, and the following four folders are available under the **ROOT_PATH** with the **MAX_EXPORT_SIZE** defined as 600 MB:

  1. Folder_1: 600 MB
  2. Folder_2: 400 MB
  3. Folder_3: 600 MB
  4. Folder_4: 100 MB

  Since the maximum export size has been reached in Folder_1, Folder_2 will be used as the export folder, and the "export.complete" file will not be generated.

TIP: By default, the lockedPath (working directory) for any export is returned by calling GetNextLockedPath(). If an export should contain this constant value, the following line in the **Script Editor**, which is available to use in all exports, can be changed to: `lockedPath = GetNextLockedpath(root, MAX_EXPORT_SIZE, true)`.

- **CREATE_SUBMIT_FILE**: Enable this option to automatically generate a DATAGRP.SUBMIT file. If you are importing the data group into PaperVision Enterprise via a Monitored Import Path or via Data Transfer Manager, this file is required before the import can run in ImageSilo or PaperVision Enterprise.
- **DELETE_DOCUMENT_AFTER_EXPORT**: This constant specifies whether documents are deleted after they have been exported (set to **False** by default).
- **DISABLE_APPENDING**: This constant is set to **False** by default. When set to **True**, exported images will not be appended to export folders whose maximum file sizes have not been reached.
- **CONVERSION_TYPE**: This constant determines the type of image file created during the export. The default value, **CVT_NO_CONVERSION**, does not convert images during the export. If exporting to a format that supports both single and multi-page images, you must set the **CREATE_MULTI_PAGE_IMAGE** constant to **True** if you want to create multi-page images; otherwise single page images will result. For example, if you set this to **CVT_TIFF_G4_MEDJPG**, a TIFF image is created during the export. If the source image is binary, it will create a TIFF using Group 4 compression; if the source image is color (JPG or BMP), it will create a TIFF using Medium JPEG compression. (See "Enumerations" on page 291 for more information.)
- **IMG_SRC_JOB_STEP_NAME**: This constant determines the job step from which images are used for the export. The default selection,**<None>**, uses the most recent image prior to exporting. To use images from another job step, select the name of the step from the **Image Source** list.
- **AUTOMATION_SERVER**: If you specify an automation server (in the **MACHINENAME_INSTANCE** format), your specified server will process exports one at a time in the **ROOT_PATH** location. When one or more automation servers are specified, separate folders may be created for multiple exports that are processed simultaneously.

  If you leave the **Automation Server** field blank during export configuration, all servers will be used to process the exports. If you are using multiple automation servers, separate each server name with a comma. You can enter wildcard characters in this field and values that you enter are not case-sensitive.

NOTE: If you are using multiple automation services and you specify multiple values for the **AUTOMATION_SERVER** constant (or, if using multiple automation services and you do not specify a value for the **AUTOMATION_SERVER** constant), your exported data may output to multiple folders (for example, data groups).

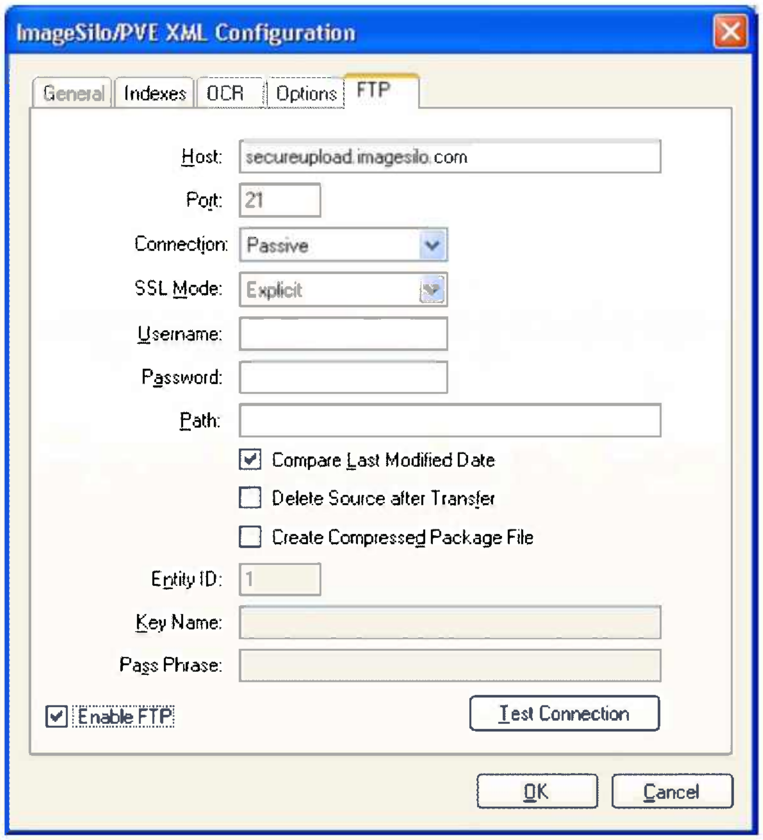
- **EXCLUSIVE_EXPORT:** This constant determines whether to create separate folders for multiple exports that are processed simultaneously. When set to **True**, the default setting, only one export will be processed at a time in the **ROOT_PATH** location. If two or more exports access the same **ROOT_PATH** location, an error message will appear in the Windows Event Viewer, indicating the export folder is already in use.

IMPORTANT!

- If you set the former **EXCLUSIVE_EXPORT** constant to **True** in PaperVision Capture R72 and earlier:
- If you will regenerate an export script in R73 or later, you must specify the automation server when you configure the export.
- If you will use an export script from R72 or earlier and you will not regenerate the script in R73 or later, it is not required to specify the automation server.

FTP

The **FTP** tab contains settings that let you securely transfer data to an **FTP** site. You can transfer data files in their original state, or they can be placed in a compressed package file. When you configure the properties on the **FTP** tab, you can modify constant values that appear in the resulting export script. Descriptions for each constant value follow. To make the options on the **FTP** tab available, you must select the **Enable FTP** check box on the bottom of the tab.



Host: secureupload.imagesilo.com
Port: 21
Connection: Passive
SSL Mode: Explicit
Username:
Password:
Path:
[✓] Compare Last Modified Date
[ ] Delete Source after Transfer
[ ] Create Compressed Package File
Entity ID: 1
Key Name:
Pass Phrase:
[✓] Enable FTP

ImageSilo/PVE XML Configuration - FTP

- **FTP_HOST:** This constant specifies the FTP host site name used for the export.
- **FTP_PORT:** This constant specifies the command port number that will be used to connect to the remote FTP server. FTP communications are typically initiated on port 21.

- **FTP_CONNECTION**: This constant specifies the type of connection that will be created. During an active connection, the remote FTP server specifies the data port number that will be used. During a passive connection, PaperVision Capture specifies the data port number that will be used.
- **FTP_ENCRYPTION**: This export supports fully encrypted FTP communications using SSL (also known as FTPS). The remote FTP server must also support this feature to take advantage of the export's capabilities. You can select one of the following from the **SSL Mode** list.
    - **Automatic** indicates the server will use SSL encryption, but will attempt to automatically determine whether to use Implicit or Explicit SSL.
    - **Implicit** indicates the SSL negotiation will start immediately after the FTP connection is established.
    - **Explicit** indicates the connection will be established in plain text and then explicitly starts the SSL negotiation.
    - **None** (no SSL encryption) indicates a standard FTP, non-encrypted session connection will be used.
- **FTP_USERNAME**: This constant specifies the user name that will be used to authenticate to the remote FTP server.
- **FTP_PASSWORD**: This constant specifies the password that will be used to authenticate to the remote FTP server. If desired, you can expose the password in the **Script Editor** by inserting the tilde character (**~**) prefix before the password (for example, **~password**).
- **FTP_PATH**: This constant specifies the folder name on the FTP site that stores the exported data. By default, this field is blank, and will write data to the user's home directory as specified by the FTP server.

  For example, other possible paths include the following:

  1. / (root)
  2. FolderA (subdirectory under home directory)
  3. /FolderA (subfolder under root path)
- **FTP_COMPARE_LAST_MODIFIED_DATE**: For an operation type related to data groups or package files, the agent will automatically record the last modified date of the file that is being processed. When the same job is processed (and potentially the same file), the last modified date of the previous run is compared to the current, last modified date. If the file has not changed, it will not be processed again.

  For data group processing, this will also allow users to perform incremental data group processing. After the data group has been changed, any data group files (that is, images) that have a modified date/time greater than or equal to the previous run's database (that is, DATAGRP.MDB or DATAGRP.XML) last modified date/time will be processed.
- **FTP_DELETE_SOURCE_AFTER_EXPORT**: Once the data has been successfully transferred, this constant allows the agent to delete the source data.
- **FTP_ENABLE_PACKAGE:** When pushing data groups or files to a remote site, you can increase transfer speed by sending a single, large file rather than hundreds or thousands of small files. This option causes the agent to create a compressed package file that increases transfer speeds and security (if encryption is enabled).

- **FTP_ENTITY_ID**: When the export is configured to create compressed package files, the Entity ID and Encryption values are placed into the package file to allow the remote PaperFlow system to decrypt the data. This constant specifies the ID of the remote entity whose encryption key will be used to decrypt the package file.
- **FTP_KEY_NAME:** This constant specifies the name of the encryption key used to decrypt the package file.
- **FTP_PASS_PHRASE**: For compressed package files, this constant specifies a user-defined pass phrase that is passed through a SHA-2 algorithm (Secure Hashing Algorithm) to generate a 256-bit hash.
- **FTP_ENABLE**: This constant specifies whether FTP has been enabled for the export.

Testing FTP Connections

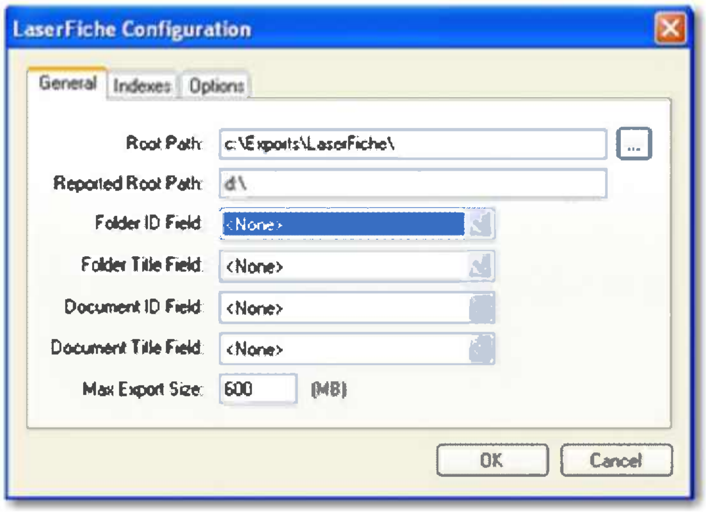
After you have configured the FTP settings, click **Test Connection** to ensure that the connection is valid. If you successfully connected to the site, click **OK** to the **Success** prompt.

LaserFiche

The **LaserFiche** export creates an ASCII text file and single-page TIFF images that can be imported into the LaserFiche system using the LaserFiche List Import Feature.

To configure the LaserFiche export

1. If the **Select Custom Code Generator** dialog box is not open, complete the procedure under "Export Definitions" on page 307.
2. In the **Select Custom Code Generator** dialog box, double-click **LaserFiche**. The **LaserFiche Configuration** dialog box appears.



Root Path: c:\Exports\LaserFiche\
Reported Root Path: d:\
Folder ID Field: <None>
Folder Title Field: <None>
Document ID Field: <None>
Document Title Field: <None>
Max Export Size: 600 (MB)

LaserFiche Configuration - General

Default values that you can modify are provided for your reference, and the available options are specific to the generator you selected. In addition, you can browse to the appropriate directories instead of manually entering file paths.

3. Assign the appropriate properties on the **General**, **Indexes**, and **Options** tabs. Descriptions for constant values that appear in the resulting export script follow.

General

When you configure the properties on the **General** tab, the following constant values appear in the resulting export script.

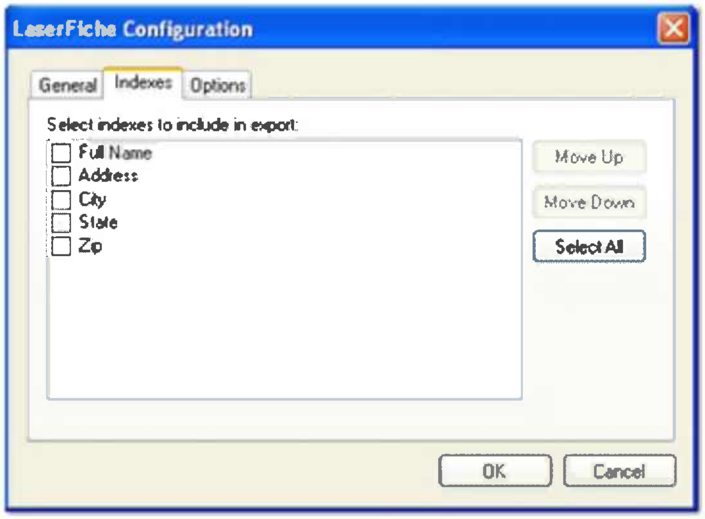
- **ROOT_PATH**: This is the location where the exports are created after the automation service processes the step.
- **REPORTED_ROOT_PATH**: The path referenced in the export file originates from this location, not the **ROOT_PATH**.

NOTE: If the **Root Path** box is blank, the export is written to the directory where the application is installed (for example, C:\Program Files\Digitech Systems\PaperVision Capture). If the **Reported Root Path** box is blank, the resulting export script displays a blank value for the **REPORTED_ROOT_PATH** constant.

- **FOLDER_ID_FIELD_NAME**: This field name specifies the index value that populates the FOLDER ID field in the export.
- **FOLDER_TITLE_FIELD_NAME**: This field name specifies the index value that populates the FOLDER TITLE field in the export.
- **DOCUMENT_ID_FIELD_NAME**: This field name specifies the index value that populates the DOCUMENT ID field in the export.
- **DOCUMENT_TITLE_FIELD_NAME**: This field name specifies the index value that populates the DOCUMENT TITLE field in the export.
- **MAX_EXPORT-SIZE**: This constant indicates the maximum export file size in megabytes. The default value is **600**.

Indexes

On the **Indexes** tab, you can specify the indexes that will appear in the export by selecting the check box next to the index. To include all of the indexes, click **Select All**. To remove all selections, click **Deselect All**. To change the order in which the indexes appear,select the index you want to move, and then click **Move Up** or **Move Down**.



Select indexes to include in export:

- [ ] Full Name
- [ ] Address
- [ ] City
- [ ] State
- [ ] Zip

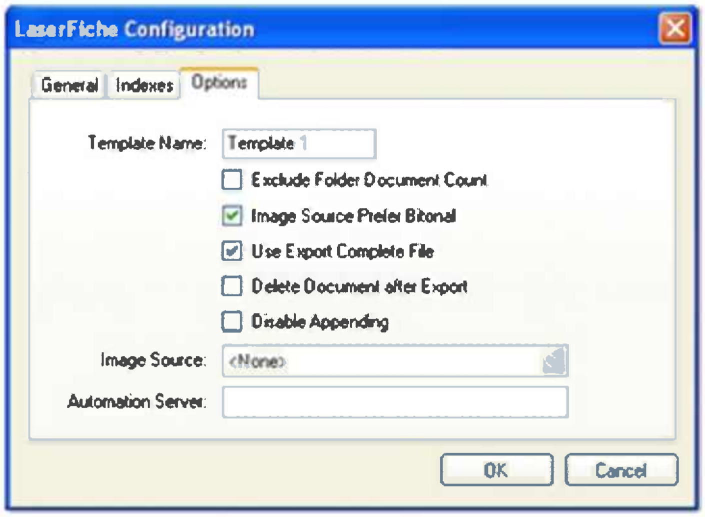
LaserFiche Configuration - Indexes

To edit the indexes in the resulting export script, you can modify the following **INDICES_TO_INCLUDE** constant.

- **INDICES_TO_INCLUDE**: This constant determines what index values are included in the export file. In the resulting script, you can enter the name of the index value(s) between quotation marks, and separate each index value with a comma.

Options

When you configure properties on the **Options** tab, you can modify constant values that appear in the resulting export script. Descriptions for each constant value follow.



Template Name: Template 1

- [ ] Exclude Folder Document Count
- [x] Image Source Prefer Bitonal
- [x] Use Export Complete File
- [ ] Delete Document after Export
- [ ] Disable Appending

Image Source: <None>

Automation Server:

LaserFiche Configuration - Options

- **TEMPLATE_NAME**: This specified value will populate the TEMPLATE NAME field in the export.
- **EXCLUDE_FOLDER_DOCUMENT_COUNT**: When set to **True**, an incrementing number can be appended to the FOLDER line of the export. It will increment from 1 to 2, and so on, for each new document. If set to **False**, no numbers are appended to the FOLDER line of the export.

- **IMG_SRC_PREFER_BITONAL_IMAGES**: This constant is applicable to dual-stream scanners and determines whether to export bitonal or color images. When set to **True**, which is the default setting, bitonal images are exported.
- **USE_EXPORT_COMPLETE_FILE:** This constant, set to **True** by default, generates an "export.complete" file once an export has reached its maximum file size, so data will no longer be appended to the export. When set to **False**, the "export.complete" file is not generated, so data may be appended to export folders that have not reached their maximum size. If you set this constant to **False**, for example, and the following four folders are available under the **ROOT_PATH** with the **MAX_EXPORT_SIZE** defined as 600 MB:

  1. Folder_1: 600 MB
  2. Folder_2: 400 MB
  3. Folder_3: 600 MB
  4. Folder_4: 100 MB

  Since the maximum export size has been reached in Folder_1, Folder_2 will be used as the export folder, and the "export.complete" file will not be generated.

TIP: By default, the lockedPath (working directory) for any export is returned by calling GetNextLockedPath(). If an export should contain this constant value, the following line in the **Script Editor**, which is available to use in all exports, can be changed to: `lockedPath = GetNextLockedpath(root, MAX_EXPORT_SIZE, true)`.

- **DELETE_DOCUMENT_AFTER_EXPORT**: This constant specifies whether documents are deleted after they have been exported (set to **False** by default).
- **DISABLE_APPENDING**: This constant is set to **False** by default. When set to **True**, exported images will not be appended to export folders whose maximum file sizes have not been reached.
- **CONVERSION_TYPE**: This constant determines the type of image file created during the export. The default value, **CVT_NO_CONVERSION**, does not convert images during the export. If exporting to a format that supports both single and multi-page images, you must set the **CREATE_MULTI_PAGE_ IMAGE** constant to **True** if you want to create multi-page images; otherwise single page images will result. For example, if you set this to **CVT_TIFF_G4_MEDJPG**, a TIFF image is created during the export. If the source image is binary, it will create a TIFF using Group 4 compression; if the source image is color (JPG or BMP), it will create a TIFF using Medium JPEG compression. (See "Enumerations" on page 291 for more information.)
- **IMG_SRC_JOB_STEP_NAME**: This constant determines the job step from which images are used for the export. The default selection,**<None>**, uses the most recent image prior to exporting. To use images from another job step, select the name of the step from the **Image Source** list.
- **AUTOMATION_SERVER**: If you specify an automation server (in the **MACHINENAME_INSTANCE** format), your specified server will process exports one at a time in the **ROOT_PATH** location. When one or more automation servers are specified, separate folders may be created for multiple exports that are processed simultaneously.

  If you leave the **Automation Server** field blank during export configuration, all servers will be used to process the exports. If you are using multiple automation servers, separate each server name with a comma. You can enter wildcard characters in this field and values that you enter are not case-sensitive.

NOTE: If you are using multiple automation services and you specify multiple values for the **AUTOMATION_SERVER** constant (or, if using multiple automation services and you do not specify a value for the **AUTOMATION_SERVER** constant), your exported data may output to multiple folders (for example, data groups).

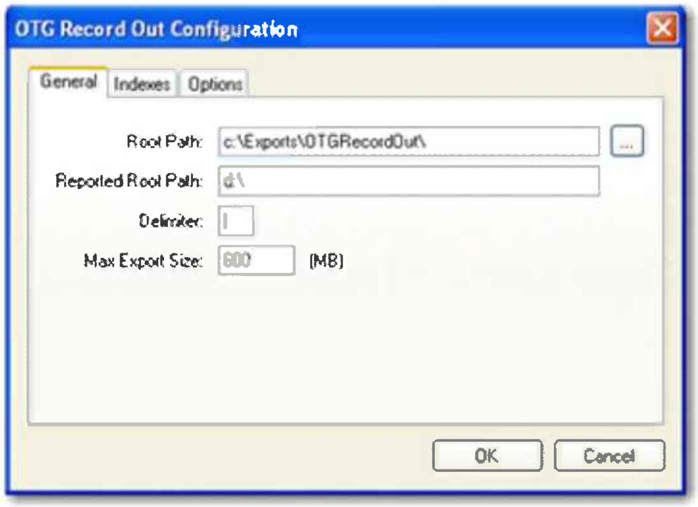
OTG Record Out

The **OTG Record Out** export creates a valid OTG Record-Out file and its associated images. This can be imported into the OTG Application Extender system using the OTG RDS.

NOTE: Ensure that date formats for the PaperVision Capture job correspond with date formats configured in OTG and that all appropriate index values have been defined.

To configure the OTG Record Out export

1. If the **Select Custom Code Generator** dialog box is not open, complete the procedure under "Export Definitions" on page 307.
2. In the **Select Custom Code Generator** dialog box, double-click **OTG Record Out**. The **OTG Record Out Configuration** dialog box appears.



Root Path: c:\Exports\OTGRecordOut\
Reported Root Path: d:\
Delimiter: |
Max Export Size: 600 (MB)

OTG Record Out Configuration - General

Default values that you can modify are provided for your reference, and the available options are specific to the generator you selected. In addition, you can browse to the appropriate directories instead of manually entering file paths.

3. Assign the appropriate properties on the **General**, **Indexes**, and **Options** tabs. Descriptions for constant values that appear in the resulting export script follow.

General

When you configure the properties on the **General** tab, the following constant values appear in the resulting export script.

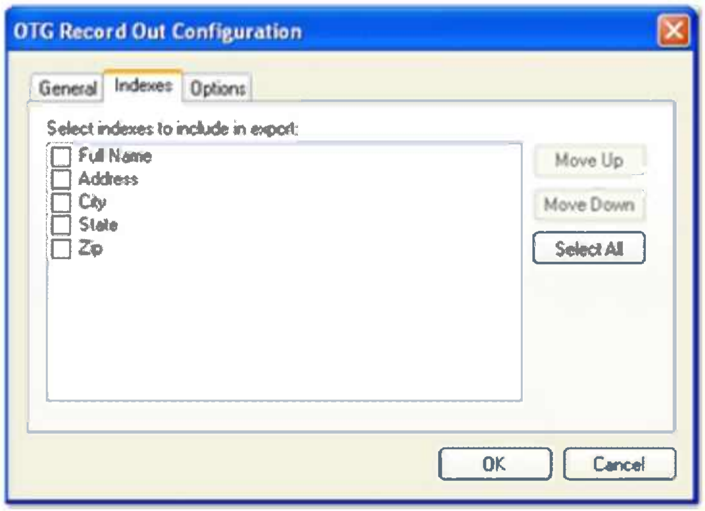
- **ROOT_PATH**: This is the location where the exports are created after the automation service processes the step.
- **REPORTED_ROOT_PATH**: The path referenced in the export file originates from this location, not the **ROOT_PATH**.

NOTE: If the **Root Path** box is blank, the export is written to the directory where the application is installed (for example, C:\Program Files\Digitech Systems\PaperVision Capture). If the **Reported Root Path** box is blank, the resulting export script displays a blank value for the **REPORTED_ROOT_PATH** constant.

- **DELIMITER**: This constant specifies the character that will delimit index values in the export file.
- **MAX_EXPORT-SIZE**: This constant indicates the maximum export file size in megabytes. The default value is **600**.

Indexes

On the **Indexes** tab, you can specify the indexes that will appear in the export by selecting the check box next to the index. To include all of the indexes, click **Select All**. To remove all selections, click **Deselect All**. To change the order in which the indexes appear, select the index you want to move, and then click **Move Up** or **Move Down**.



Select indexes to include in export:

- [ ] Full Name
- [ ] Address
- [ ] City
- [ ] State
- [ ] Zip

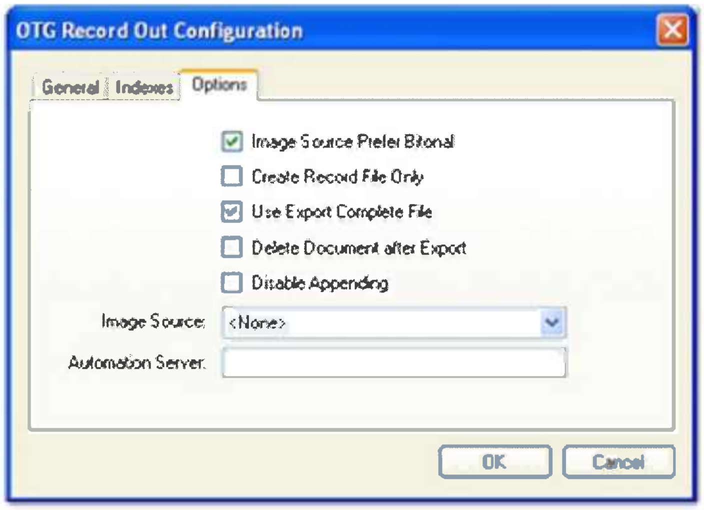
OTG Record Out Configuration - Indexes

To edit the indexes in the resulting export script, you can modify the following **INDICES_TO_INCLUDE** constant.

- **INDICES_TO_INCLUDE**: This constant determines what index values are included in the export file. In the resulting script, you can enter the name of the index value(s) between quotation marks, and separate each index value with a comma.

Options

When you configure properties on the **Options** tab, you can modify constant values that appear in the resulting export script. Descriptions for each constant value follow.



[✓] Image Source Prefer Bitonal
[ ] Create Record File Only
[✓] Use Export Complete File
[ ] Delete Document after Export
[ ] Disable Appending
Image Source: <None>
Automation Server:

OTG Record Out Configuration - Options

- **IMG_SRC_PREFER_BITONAL_IMAGES**: This constant is applicable to dual-stream scanners and determines whether to export bitonal or color images. When set to **True**, which is the default setting, bitonal images are exported.
- **CREATE_RECORD_FILE_ONLY**: If set to **True**, a RECORD.TXT file will be created, but no images will be created during the export.
- **USE_EXPORT_COMPLETE_FILE:** This constant, set to **True** by default, generates an "export.complete" file once an export has reached its maximum file size, so data will no longer be appended to the export. When set to **False**, the "export.complete" file is not generated, so data may be appended to export folders that have not reached their maximum size. If you set this constant to **False**, for example, and the following four folders are available under the **ROOT_PATH** with the **MAX_EXPORT_SIZE** defined as 600 MB:

  1. Folder_1: 600 MB
  2. Folder_2: 400 MB
  3. Folder_3: 600 MB
  4. Folder_4: 100 MB

  Since the maximum export size has been reached in Folder_1, Folder_2 will be used as the export folder, and the "export.complete" file will not be generated.

TIP: By default, the lockedPath (working directory) for any export is returned by calling GetNextLockedPath(). If an export should contain this constant value, the following line in the **Script Editor**, which is available to use in all exports, can be changed to: `lockedPath = GetNextLockedpath(root, MAX_EXPORT_SIZE, true)`.

- **DELETE_DOCUMENT_AFTER_EXPORT**: This constant specifies whether documents are deleted after they have been exported (set to **False** by default).
- **DISABLE_APPENDING**: This constant is set to **False** by default. When set to **True**, exported images will not be appended to export folders whose maximum file sizes have not been reached.
- **IMG_SRC_JOB_STEP_NAME**: This constant determines the job step from which images are used for the export. The default selection,**<None>**, uses the most recent image prior to exporting. To use images from another job step, select the name of the step from the **Image Source** list.
- **AUTOMATION_SERVER**: If you specify an automation server (in the **MACHINENAME_INSTANCE** format), your specified server will process exports one at a time in the **ROOT_PATH** location. When one or more automation servers are specified, separate folders may be created for multiple exports that are processed simultaneously.

  If you leave the **Automation Server** field blank during export configuration, all servers will be used to process the exports. If you are using multiple automation servers, separate each server name with a comma. You can enter wildcard characters in this field and values that you enter are not case-sensitive.

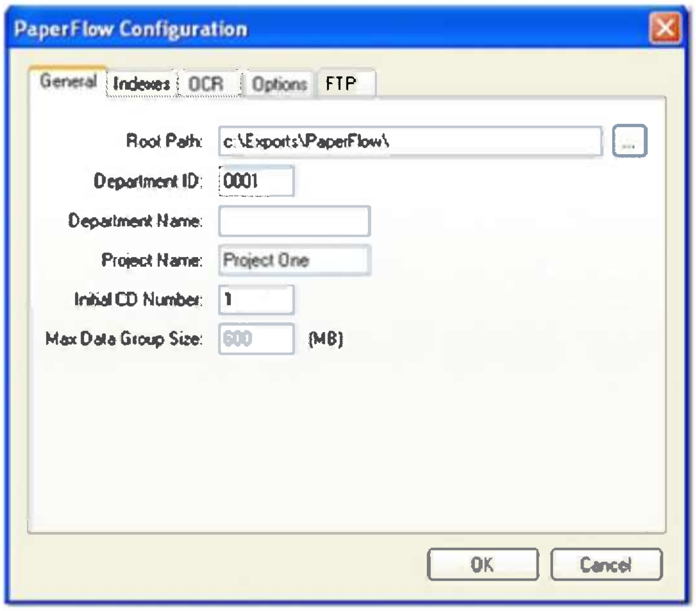
  NOTE: If you are using multiple automation services and you specify multiple values for the **AUTOMATION_SERVER** constant (or, if using multiple automation services and you do not specify a value for the **AUTOMATION_SERVER** constant), your exported data may output to multiple folders (for example, data groups).

PaperFlow

The **PaperFlow** export can be used to import batches into PaperFlow, OCRFlow, or QCFlow.

To configure the PaperFlow export

1. If the **Select Custom Code Generator** dialog box is not open, complete the procedure under "Export Definitions" on page 307.
2. In the **Select Custom Code Generator** dialog box, double-click **PaperFlow**. The **PaperFlow** dialog box appears.



Root Path: c:\Exports\PaperFlow\
Department ID: 0001
Department Name:
Project Name: Project One
Initial CD Number: 1
Max Data Group Size: 600 (MB)

PaperFlow Configuration - General

Default values that you can modify are provided for your reference, and the available options are specific to the generator you selected. In addition, you can browse to the appropriate directories instead of manually entering file paths.

3. Assign the appropriate properties on the **General**, **Indexes**, **OCR**, **Options**, and **FTP** tabs. Descriptions for constant values that appear in the resulting export script follow.

General

When you configure the properties on the **General** tab, the following constant values appear in the resulting export script.

- **ROOT_PATH**: This is the location where the exports are created after the automation service processes the step.

NOTE: If the **Root Path** box is blank, the export is written to the directory where the application is installed (for example, C:\Program Files\Digitech Systems\PaperVision Capture).

- **DEPT_ID**: This value is uniquely assigned to each client for which the export is generated. The default value is **0001**.
- **DEPT_NAME**: This value is uniquely assigned to each client or department and is a required field. The default value is blank.
- **PROJECT_NAME**: This value is uniquely assigned to each client or department. The default value is **Project One**.
- **INITIAL_CD_NUMBER**: This value can be used to export to a CD. The default value is **1**.

If you change this value after you have already performed a PaperFlow export, the new value will not be reflected in exported data groups unless you remove the "//" comment codes. The "Reset CD Number?" code should appear as follows in the export script:

```
if (!PVUtilities.TrySetCustomCounter(DEPT_ID + "_" + PROJECT_NAME,
INITIAL_CD_NUMBER, out error))

throw (new Exception("Unable to reset custom counter: " + error.Message));
```

After you remove the comment codes, you must run the export to reset the counter. The next data group that is created will reflect your new **INITIAL_CD_NUMBER** value. Lastly, to ensure that new data groups increment properly from the new **INITIAL_CD_NUMBER**, you must insert the "\\" comment codes once again:

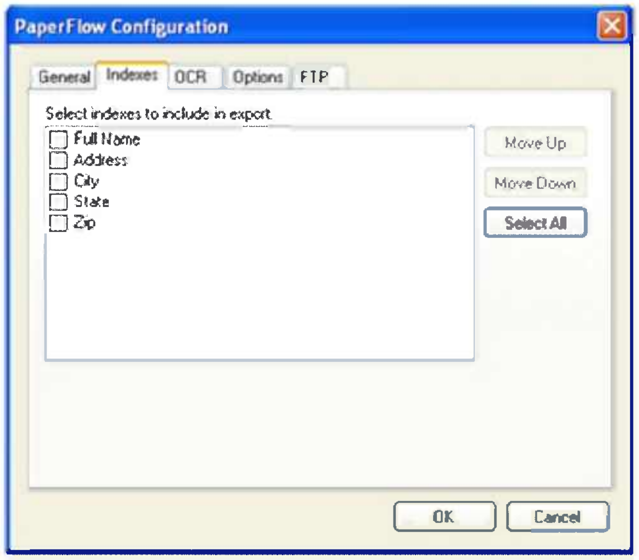
```
//if (!PVUtilities.TrySetCustomCounter(DEPT_ID + "_" + PROJECT_NAME,
INITIAL_CD_NUMBER, out error))

//throw (new Exception("Unable to reset custom counter: " +
error.Message));
```

NOTE: You must export to a directory that does not contain existing data groups. Otherwise, the system will attempt to append to data groups whose maximum size has not been reached, and the new **INITIAL_CD_NUMBER** value may be ignored or other unexpected results may occur.

- **MAX_DATAGROUP_SIZE**: This indicates the maximum size (in MB) that a data group can reach before a new data group begins. The default value is **600**, the standard CD size.

Indexes

On the **Indexes** tab, you can specify the indexes that will appear in the export by selecting the check box next to the index. To include all of the indexes, click **Select All**. To remove all selections, click **Deselect All**. To change the order in which the indexes appear, select the index you want to move, and then click **Move Up** or **Move Down**.



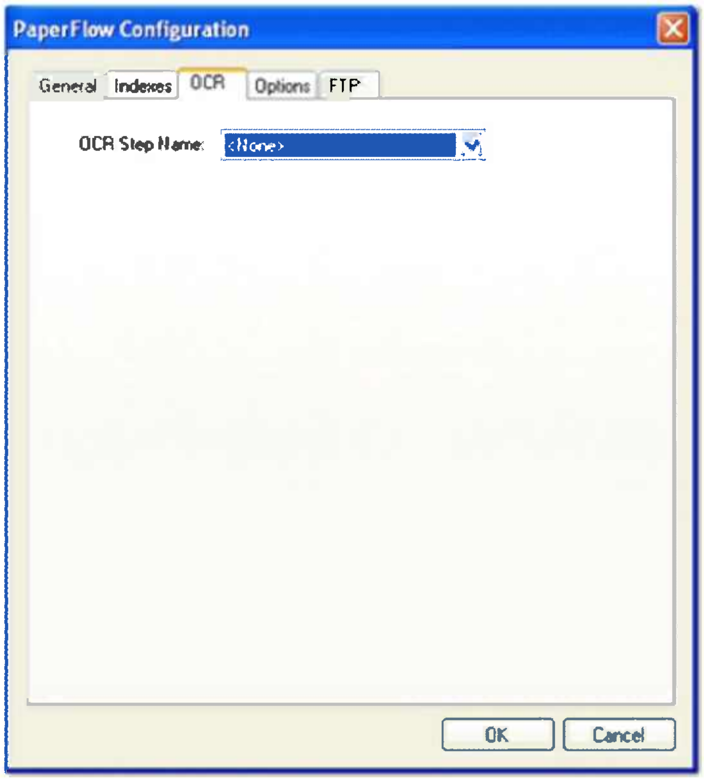
PaperFlow Configuration - Indexes

To edit the indexes in the resulting export script, you can modify the following **INDICES_TO_INCLUDE** constant.

- **INDICES_TO_INCLUDE**: This constant determines what index values are included in the export file. In the resulting script, you can enter the name of the index value(s) between quotation marks, and separate each index value with a comma.

OCR

When you configure the properties on the **OCR** tab, you can modify constant values that appear in the resulting export script. Descriptions for each constant value follow.



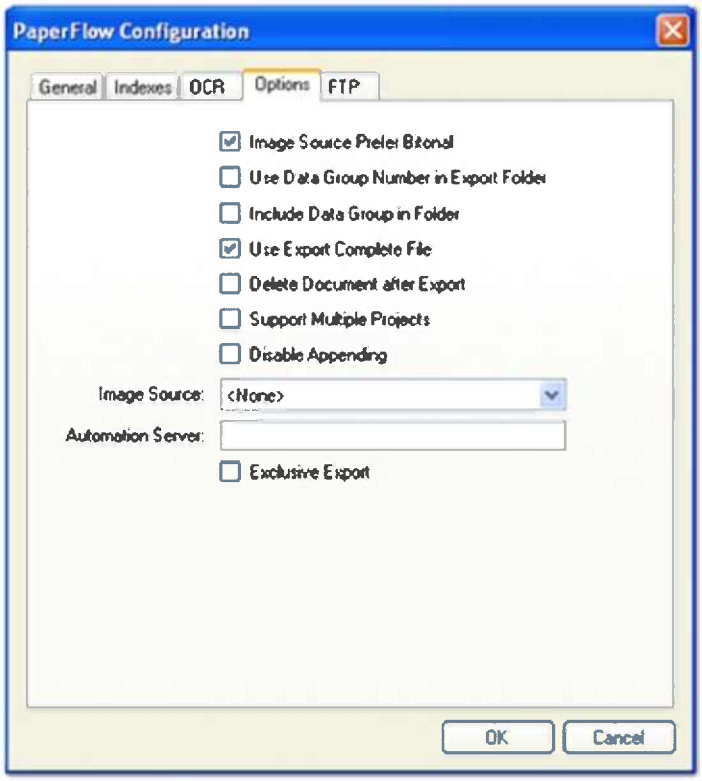
OCR Step Name: <None>

PaperFlow Configuration - OCR

- **OCR_JOB_STEP_NAME**: This constant specifies the job step whose full-text data are used for the export. No value is defined by default, so full-text data from the current job step are used for the export.

Options

When you configure properties on the **Options** tab, you can modify constant values that appear in the resulting export script. Descriptions for each constant value follow.



- [✓] Image Source Prefer Bitonal
- [ ] Use Data Group Number in Export Folder
- [ ] Include Data Group in Folder
- [✓] Use Export Complete File
- [ ] Delete Document after Export
- [ ] Support Multiple Projects
- [ ] Disable Appending

Image Source: <None>

Automation Server:

- [ ] Exclusive Export

PaperFlow Configuration - Options

- **IMG_SRC_PREFER_BITONAL_IMAGES**: This constant is applicable to dual-stream scanners and determines whether to export bitonal or color images. When set to **True**, which is the default setting, bitonal images are exported.
- **USE_DATAGROUP_NUMBER_IN_EXPORT_FOLDER**: When set to **True**, the parent export directory will be organized by data group name instead of export number.
- **INCLUDE_DATAGROUP_IN_FOLDER**: When set to **True**, a folder named "DATAGRP" is created under the directory in which the export data is copied (for example, <root>\<export#>\DATAGRP\<export data>). When set to **False** (the default setting), the "DATAGRP" folder is not created.
- **USE_EXPORT_COMPLETE_FILE:** This constant, set to **True** by default, generates an "export.complete" file once an export has reached its maximum file size, so data will no longer be appended to the export. When set to **False**, the "export.complete" file is not generated, so data may be appended to export folders that have not reached their maximum size. If you set this constant to **False**, for example, and the following four folders are available under the **ROOT_PATH** with the **MAX_EXPORT_SIZE** defined as 600 MB:

  1. Folder_1: 600 MB
  2. Folder_2: 400 MB
  3. Folder_3: 600 MB
  4. Folder_4: 100 MB

  Since the maximum export size has been reached in Folder_1, Folder_2 will be used as the export folder, and the "export.complete" file will not be generated.

TIP: By default, the lockedPath (working directory) for any export is returned by calling GetNextLockedPath(). If an export should contain this constant value, the following line in the **Script Editor**, which is available to use in all exports, can be changed to: `lockedPath = GetNextLockedpath(root, MAX_EXPORT_SIZE, true)`.

- **DELETE_DOCUMENT_AFTER_EXPORT**: This constant specifies whether documents are deleted after they have been exported (set to **False** by default).
- **SUPPORT_MULTIPLE_PROJECTS**: When set to **True**, multiple Department IDs will be exported to the same folder, creating a single MDB file. When set to **False** (the default setting), one Department ID will be exported to a single folder.
- **DISABLE_APPENDING**: This constant is set to **False** by default. When set to **True**, exported images will not be appended to export folders whose maximum file sizes have not been reached.
- **IMG_SRC_JOB_STEP_NAME**: This constant determines the job step from which images are used for the export. The default selection,**<None>**, uses the most recent image prior to exporting. To use images from another job step, select the name of the step from the **Image Source** list.
- **AUTOMATION_SERVER**: If you specify an automation server (in the **MACHINENAME_INSTANCE** format), your specified server will process exports one at a time in the **ROOT_PATH** location. When one or more automation servers are specified, separate folders may be created for multiple exports that are processed simultaneously.

  If you leave the **Automation Server** field blank during export configuration, all servers will be used to process the exports. If you are using multiple automation servers, separate each server name with a comma. You can enter wildcard characters in this field and values that you enter are not case-sensitive.

  NOTE: If you are using multiple automation services and you specify multiple values for the **AUTOMATION_SERVER** constant (or, if using multiple automation services and you do not specify a value for the **AUTOMATION_SERVER** constant), your exported data may output to multiple folders (for example, data groups).

- **EXCLUSIVE_EXPORT:** This constant determines whether to create separate folders for multiple exports that are processed simultaneously. When set to **True**, the default setting, only one export will be processed at a time in the **ROOT_PATH** location. If two or more exports access the same **ROOT_PATH** location, an error message will appear in the Windows Event Viewer, indicating the export folder is already in use.

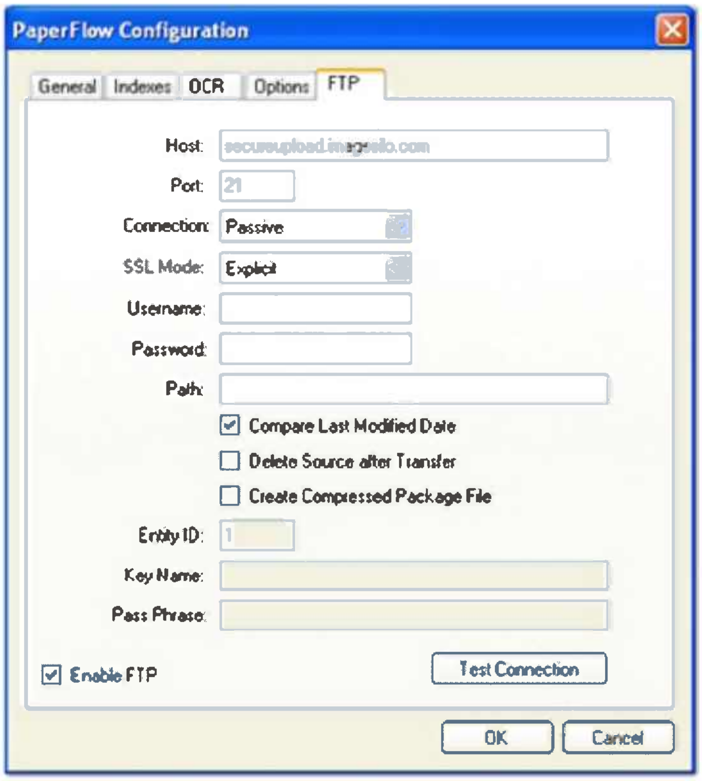
IMPORTANT!

If you set the former **EXCLUSIVE_EXPORT** constant to **True** in PaperVision Capture R72 and earlier:

- If you will regenerate an export script in R73 or later, you must specify the automation server when you configure the export.
- If you will use an export script from R72 or earlier and you will not regenerate the script in R73 or later, it is not required to specify the automation server.

FTP

The **FTP** tab contains settings that let you securely transfer data to an FTP site. You can transfer data files in their original state, or they can be placed in a compressed package file. When you configure the properties on the **FTP** tab, you can modify constant values that appear in the resulting export script. Descriptions for each constant value follow. To make the options on the **FTP** tab available, you must select the **Enable FTP** check box on the bottom of the tab.



Host: secureupload.imagesilo.com
Port: 21
Connection: Passive
SSL Mode: Explicit
Username:
Password:
Path:
[✓] Compare Last Modified Date
[ ] Delete Source after Transfer
[ ] Create Compressed Package File
Entity ID: 1
Key Name:
Pass Phrase:
[✓] Enable FTP

PaperFlow Configuration - FTP

- **FTP_HOST:** This constant specifies the FTP host site name used for the export.
- **FTP_PORT:** This constant specifies the command port number that will be used to connect to the remote FTP server. FTP communications are typically initiated on port 21.
- **FTP_CONNECTION**: This constant specifies the type of connection that will be created. During an active connection, the remote FTP server specifies the data port number that will be used. During a passive connection, PaperVision Capture specifies the data port number that will be used.
- **FTP_ENCRYPTION**: This export supports fully encrypted FTP communications using SSL (also known as FTPS). The remote FTP server must also support this feature to take advantage of the export's capabilities. You can select one of the following from the **SSL Mode** list.
  - **Automatic** indicates the server will use SSL encryption, but will attempt to automatically determine whether to use Implicit or Explicit SSL.
  - **Implicit** indicates the SSL negotiation will start immediately after the FTP connection is established.

  - **Explicit** indicates the connection will be established in plain text and then explicitly starts the SSL negotiation.
  - **None** (no SSL encryption) indicates a standard FTP, non-encrypted session connection will be used.
- **FTP_USERNAME**: This constant specifies the user name that will be used to authenticate to the remote FTP server.
- **FTP_PASSWORD**: This constant specifies the password that will be used to authenticate to the remote FTP server. If desired, you can expose the password in the **Script Editor** by inserting the tilde character (**~**) prefix before the password (for example, **~password**).
- **FTP_PATH**: This constant specifies the folder name on the FTP site that stores the exported data. By default, this field is blank, and will write data to the user's home directory as specified by the FTP server.

  For example, other possible paths include the following:

  1. / (root)
  2. FolderA (subdirectory under home directory)
  3. /FolderA (subfolder under root path)
- **FTP_COMPARE_LAST_MODIFIED_DATE**: For an operation type related to data groups or package files, the agent will automatically record the last modified date of the file that is being processed. When the same job is processed (and potentially the same file), the last modified date of the previous run is compared to the current, last modified date. If the file has not changed, it will not be processed again.

  For data group processing, this will also allow users to perform incremental data group processing. After the data group has been changed, any data group files (that is, images) that have a modified date/time greater than or equal to the previous run's database (that is, DATAGRP.MDB or DATAGRP.XML) last modified date/time will be processed.
- **FTP_DELETE_SOURCE_AFTER_EXPORT**: Once the data has been successfully transferred, this constant allows the agent to delete the source data.
- **FTP_ENABLE_PACKAGE:** When pushing data groups or files to a remote site, you can increase transfer speed by sending a single, large file rather than hundreds or thousands of small files. This option causes the agent to create a compressed package file that increases transfer speeds and security (if encryption is enabled).
- **FTP_ENTITY_ID**: When the export is configured to create compressed package files, the Entity ID and Encryption values are placed into the package file to allow the remote PaperFlow system to decrypt the data. This constant specifies the ID of the remote entity whose encryption key will be used to decrypt the package file.
- **FTP_KEY_NAME:** This constant specifies the name of the encryption key used to decrypt the package file.
- **FTP_PASS_PHRASE**: For compressed package files, this constant specifies a user-defined pass phrase that is passed through a SHA-2 algorithm (Secure Hashing Algorithm) to generate a 256-bit hash.
- **FTP_ENABLE**: This constant specifies whether FTP has been enabled for the export.

Testing FTP Connections

After you have configured the FTP settings, click **TestConnection** to ensure that the connection is valid. If you successfully connected to the site, click **OK** to the **Success** prompt.

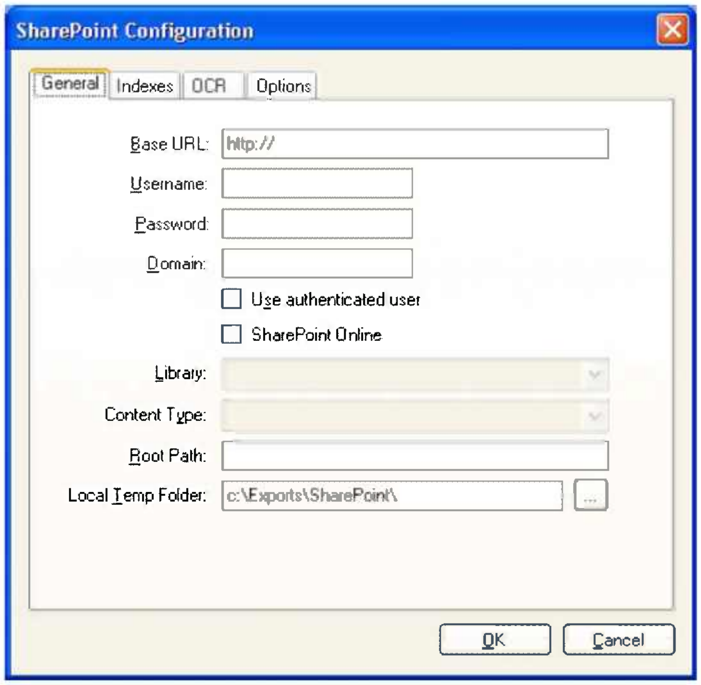
SharePoint

The **SharePoint** export creates a file that can be used to import PaperVision Capture data into a Microsoft® SharePoint® site.

NOTE: Only Microsoft SharePoint 2007 (on Windows Server 2003 or 2008) or Microsoft SharePoint 2010 (on Windows Server 2008) are supported for this export.

To configure the SharePoint export

1. If the **Select Custom Code Generator** dialog box is not open, complete the procedure under "Export Definitions" on page 307.
2. In the **Select Custom Code Generator** dialog box, double-click **SharePoint**. The **SharePoint Configuration** dialog box appears.



Base URL: http://
Username:
Password:
Domain:
Library:
Content Type:
Root Path:
Local Temp Folder: c:\Exports\SharePoint\

SharePoint Configuration - General

3. You must configure all properties on the **General** tab. Descriptions for each property follow this procedure.
4. Proceed to the **Indexes** tab. If you entered valid SharePoint data, you can map PaperVision Capture index field names to SharePoint columns.

NOTE: An error message will inform you when you have entered invalid SharePoint data.

---

5. If applicable, map the appropriate index field names to SharePoint columns. See "Indexes" on page 346 for more information.
6. Proceed to the **OCR** and **Options** tabs to modify the appropriate properties. See "OCR" on page 348 and "Options" on page 348 for information about each property.

General

When you configure the properties on the **General** tab, the following constant values appear in the resulting export script.

- **SHAREPOINT_BASE_URL**: This constant specifies the Microsoft SharePoint host site name and port used for the export.
- **SHAREPOINT_USERNAME**: This constant specifies the Microsoft SharePoint user name.
- **SHAREPOINT_PASSWORD**: This constant specifies the Microsoft SharePoint user's password. By default, the SharePoint password is encrypted in the **Script Editor**. If desired, you can expose the password in the **Script Editor** by inserting the tilde character (~) prefix before the password, for example, **~password**.
- **SHAREPOINT_DOMAIN**: This constant specifies the Microsoft SharePoint domain name.

NOTE: If you select the **Use Authenticated User** option, the database connection will use Windows Authentication credentials. Entering a user name and password for the database will supersede the Windows Authentication credentials.

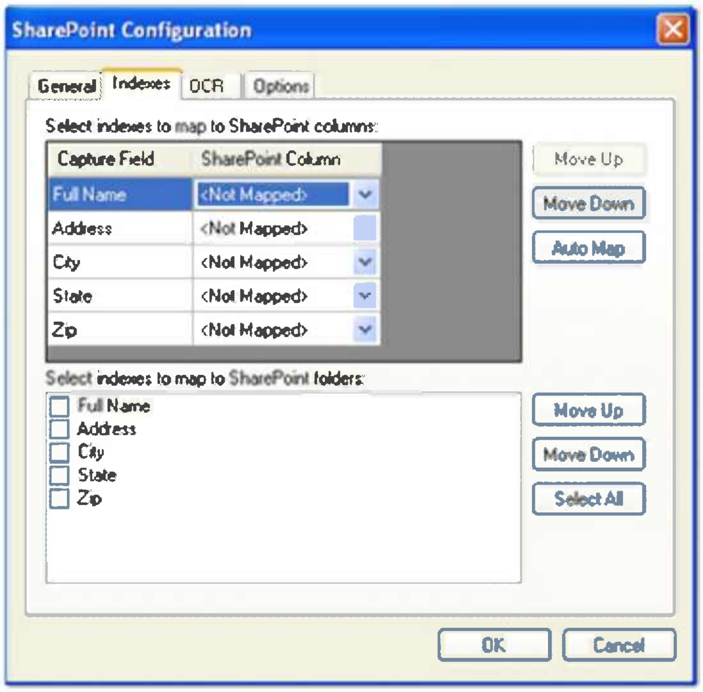
- If you select the **SharePoint Online** option, it enables your hosted SharePoint solution.
- **SHAREPOINT_LIBRARY**: This constant specifies the Microsoft SharePoint library.
- **CONTENT_TYPE**: If applicable, select the SharePoint content type. If content types have been created in the SharePoint library, they will appear in this list. See "SharePoint Content Types" on page 350 for more information.
- **ROOT_PATH**: This is the location on your SharePoint Server where the folders will be created once the automation service processes the step. If you do not specify a value for the **ROOT_PATH** property, no folders will be created on the SharePoint Server.
- **LOCAL_TEMP_FOLDER:**This constant specifies the local folder path where the Microsoft SharePoint export is temporarily stored on your local machine prior to moving to the Microsoft SharePoint site.

Indexes

On the **Indexes** tab, you can map PaperVision Capture index field names to SharePoint column names. PaperVision Capture index field names appear in the left column. From the **SharePoint Column** list, select the column name that maps to the PaperVision Capture index field name. To automatically map a PaperVision Capture index field to a similarly-named Microsoft SharePoint column, click **Auto Map**. To edit the indexes in the resulting export script, you can modify the **INDICES_TO_INCLUDE** constant, which is described below.

NOTE: Some PaperVision Capture index field types may not be supported in Microsoft SharePoint. Therefore, some index fields may not be mapped to SharePoint columns in the export.

Alternatively, if a SharePoint column does not exist, you can create a new column that will be mapped to the corresponding index field. To do this, select **<Create New>** from the **SharePointColumn** list.



Capture Field	SharePoint Column
Full Name	<Not Mapped>
Address	<Not Mapped>
City	<Not Mapped>
State	<Not Mapped>
Zip	<Not Mapped>

- [ ] Full Name
- [ ] Address
- [ ] City
- [ ] State
- [ ] Zip

SharePoint Configuration - Indexes

- **INDICES_TO_INCLUDE**: This constant determines the index values mapped from PaperVision Capture to Microsoft SharePoint columns. These columns must already be defined in your Microsoft SharePoint list. To provide a mapping between fields, the following format is required:

  ```
  <Capture Field>:<SharePoint>
  ```

  ```
  Example 1: "Field1", "Field 2", "Field 3", etc.
  ```

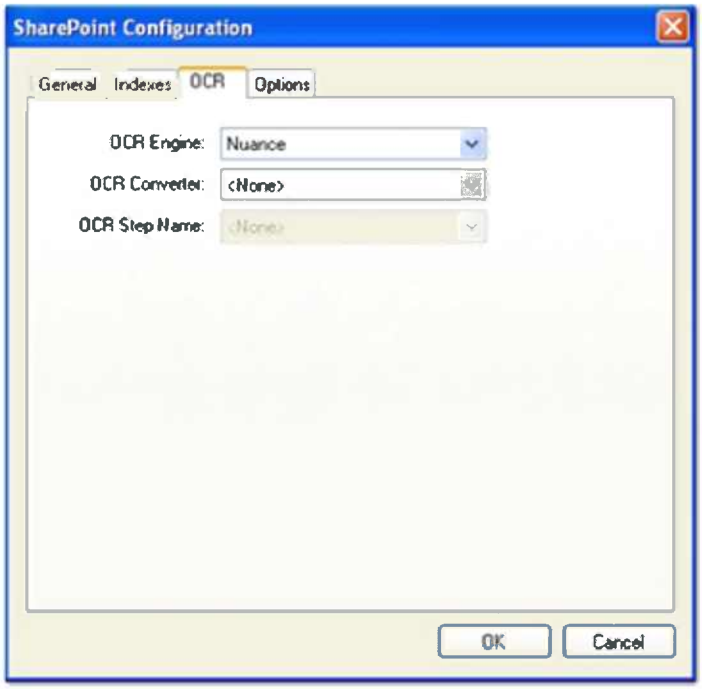
  NOTE: This format can be used when the same field names exist in both PaperVision Capture and your Microsoft SharePoint site.

  ```
  Example 2: "Field1:Field1", "Field2:Field2:", etc.
  ```

  NOTE: This constant is optional, so when an empty array is assigned to INDICES_TO_INCLUDE, Microsoft SharePoint's metadata is not populated.

OCR

When you configure the properties on the **OCR** tab, you can modify constant values that appear in the resulting export script. Descriptions for each constant value follow.



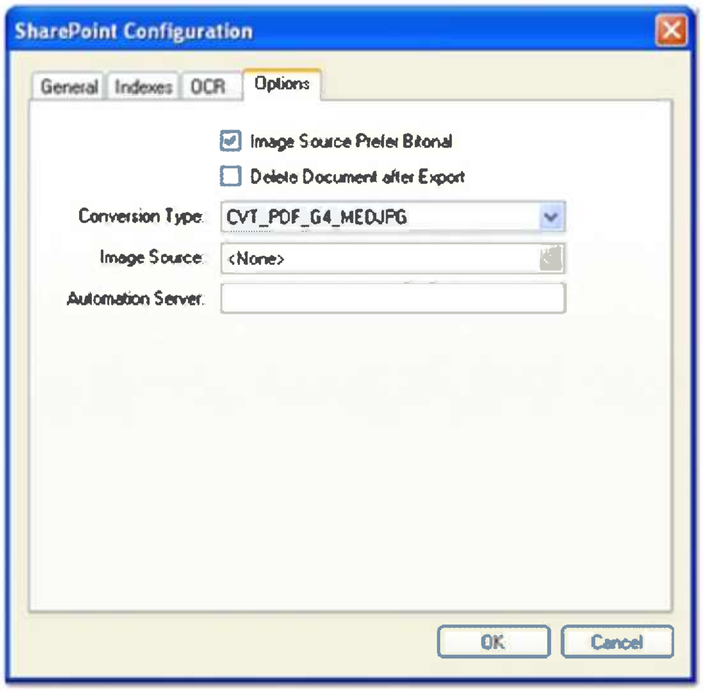
OCR Engine: Nuance
OCR Converter: <None>
OCR Step Name: <None>

SharePoint Configuration - OCR

- **OCR_ENGINE**: This constant specifies the OCR engine (Nuance or Open Text) that processes OCR data for the export.
- **OCR_CONVERTER_CODE**: This constant specifies the OCR converter code, such as PDF, Text, XML, etc., whose output format is used to export full-text data. When no value is defined (the default setting), both images and associated full-text data are exported. If you select the PaperVision Full-Text OCR converter, only full-text data will be exported (associated images will not be exported).
- **OCR_JOB_STEP_NAME**: This constant specifies the job step whose full-text data are used for the export. No value is defined by default, so full-text data from the current job step are used for the export.

Options

When you configure properties on the **Options** tab, you can modify constant values that appear in the resulting export script. Descriptions for each constant value follow.



☑ Image Source Prefer Bitonal

☐ Delete Document after Export

Conversion Type: CVT_PDF_G4_MEDJPG

Image Source: <None>

Automation Server:

SharePoint Configuration - Options

- **IMG_SRC_PREFER_BITONAL_IMAGES**: This constant is applicable to dual-stream scanners and determines whether to export bitonal or color images. When set to **True**, which is the default setting, bitonal images are exported.
- **DELETE_DOCUMENT_AFTER_EXPORT**: This constant specifies whether documents are deleted after they have been exported (set to **False** by default).
- **CONVERSION_TYPE**: This constant determines the type of image file created during the export. The default value, **CVT_NO_CONVERSION**, does not convert images during the export. If exporting to a format that supports both single and multi-page images, you must set the **CREATE_MULTI_PAGE_ IMAGE** constant to **True** if you want to create multi-page images; otherwise single page images will result. For example, if you set this to **CVT_TIFF_G4_MEDJPG**, a TIFF image is created during the export. If the source image is binary, it will create a TIFF using Group 4 compression; if the source image is color (JPG or BMP), it will create a TIFF using Medium JPEG compression. (See "Enumerations" on page 291 for more information.)
- **IMG_SRC_JOB_STEP_NAME**: This constant determines the job step from which images are used for the export. The default selection,**<None>**, uses the most recent image prior to exporting. To use images from another job step, select the name of the step from the **Image Source** list.
- **AUTOMATION_SERVER**: If you specify an automation server (in the **MACHINENAME_INSTANCE** format), your specified server will process exports one at a time in the **ROOT_PATH** location. When one or more automation servers are specified, separate folders may be created for multiple exports that are processed simultaneously.

  If you leave the **Automation Server** field blank during export configuration, all servers will be used to process the exports. If you are using multiple automation servers, separate each server name with a comma. You can enter wildcard characters in this field and values that you enter are not case-sensitive.

NOTE: If you are using multiple automation services and you specify multiple values for the **AUTOMATION_SERVER** constant (or, if using multiple automation services and you do not specify a value for the **AUTOMATION_SERVER** constant), your exported data may output to multiple folders (for example, data groups).

SharePoint Content Types

When exporting documents to a SharePoint site, you can optionally link documents to content types. Content types contain limited subsets of index fields in a SharePoint library. For example, a Financial Documents SharePoint library can contain three content types including Purchase Orders, Invoices, and Expense Reports. Each content type can be associated with a specific subset of index fields. Document content types, the default selection, include all index fields in the library. Content types are independent of file types, so one content type can be applied to multiple file types, such as Microsoft Word documents, Excel spreadsheets, and PowerPoint presentations.

For example, Purchase Orders, Invoices, and Expense Reports content types in a Financial Documents library can be associated with the following PaperVision Capture index fields:

Content Type	Check Number	Check Date	Company Name	PO Number	PO Date	Invoice Number	Invoice Date	Amount
Purchase Orders			x	x	x			x
Invoices	x	x	x	x	x	x	x	x
Expense Reports			x			x		x

Information on SharePoint 2007 and 2010 content types, respectively, can be found on the following sites:

http://technet.microsoft.com/en-us/library/cc262735(office.12).aspx

http://technet.microsoft.com/en-us/library/cc262735.aspx

Job Configuration

The following instructions describe how to configure a job that will process a PaperFlow export that can be used to import batches into PaperFlow, OCRFlow, or QCFlow. The following job contains a **Capture**, **Indexing**, and a **Custom Code** step with the export that handles index and detail fields.

Configuring a Job to Process a PaperFlow Export

1. After inserting a **Capture**, **Indexing**, and **Custom Code** job step, respectively, onto the workspace of the **Job Definitions** window, double-click the **Indexing** step to display the **Properties** tab on the left pane.
2. On the **Properties** tab, expand **Indexes**.
3. Click **Indexes**, and then click the ellipsis button to open the **Index Configuration** dialog box similar to the following.