

Perceptive Intelligent Capture

Scripting Guide

Version: 5.7.x

Written by: Product Development, R&D
Date: April 2022

© 2016 Lexmark. All rights reserved.

Lexmark is a trademark of Lexmark International Inc., registered in the U.S. and/or other countries. All other trademarks are the property of their respective owners. No part of this publication may be reproduced, stored, or transmitted in any form without the prior written permission of Lexmark.

Table of Contents

Overview	7
Script Event Reference.....	7
<i>VerifierFormLoadEvent</i>	7
<i>Usage</i>	9
ScriptModule.....	9
<i>Cedar ScriptModule Event Interface</i>	9
<i>Create a script for document export</i>	9
<i>Methods and Properties</i>	10
Document Events	35
<i>Cedar DocClass Event Interface</i>	35
<Field _n > (<i>Cedar FieldDef Event Interface</i>).....	40
Workdoc Object Reference (SCBCdrWorkdocLib).....	51
SCBCdrWorkdoc	51
<i>Description</i>	51
<i>Type Definitions</i>	51
<i>Methods and Properties</i>	55
SCBCdrFields	85
<i>Description</i>	85
<i>Methods and Properties</i>	85
SCBCdrField.....	89
<i>Description</i>	89
<i>Type Definitions</i>	89
<i>Methods and Properties</i>	89
SCBCdrCandidate	104
<i>Description</i>	104
<i>Methods and Properties</i>	104
SCBCdrTable	109
<i>Descriptions</i>	109
<i>Type Definitions</i>	109
<i>Methods and Properties</i>	110
SCBCdrTextblock.....	133
<i>Description</i>	133
<i>Methods and properties</i>	133

SCBCdrWord.....	135
<i>Description</i>	135
<i>Methods and Properties</i>	135
SCBCdrDocPage.....	137
<i>Description</i>	137
<i>Type Definitions</i>	137
<i>Methods and Properties</i>	138
SCBCdrFolder.....	143
<i>Description</i>	143
<i>Methods and Properties</i>	143
Cedar Project Object Reference (SCBCdrPROJLib)	146
Description.....	146
Type Definitions.....	146
<i>Methods and Properties</i>	154
SCBCdrDocClasses.....	165
<i>Description</i>	165
<i>Methods and Properties</i>	165
SCBCdrDocClass.....	167
<i>Description</i>	167
<i>Type Definitions</i>	167
<i>Methods and Properties</i>	169
SCBCdrFieldDefs.....	176
<i>Description</i>	176
<i>Methods and Properties</i>	176
SCBCdrFieldDef.....	178
<i>Description</i>	178
<i>Type Definitions</i>	178
<i>Methods and Properties</i>	180
SCBCdrSettings.....	186
<i>Description</i>	186
<i>Methods and Properties</i>	186
SCBCdrScriptModule.....	190
<i>Description</i>	190
<i>Methods and Properties</i>	190
SCBCdrScriptProject.....	192

<i>Description</i>	192
<i>Methods and Properties</i>	192
SCBCdrScriptAccess	194
<i>Description</i>	194
<i>Methods and Properties</i>	194
CDRADSLib	196
SCBCdrSupExSettings	196
<i>Description</i>	196
<i>Methods and Properties</i>	196
Analysis Engines Object Reference	199
SCBCdrAssociativeDbExtractionSettings	199
<i>Description</i>	199
<i>Type Definitions</i>	199
<i>Method and Properties</i>	199
SCBCdrParaCheckLib	211
SCBCdrParaCheckAnalysisSettings	211
<i>Description</i>	211
<i>Methods and Properties</i>	211
PayeeVocEntries	215
<i>Description</i>	215
<i>Methods and Properties</i>	215
SCBCdrFormatEngine	217
CdrFormatSettings	217
Methods and Properties	219
<i>SrchFlag</i>	219
<i>TestString</i>	220
StringComp Object Reference (SCBCdrSTRCOMPLib)	222
SCBCdrStringComp	222
<i>Description</i>	222
<i>Type Definitions</i>	222
<i>Methods and Properties</i>	223
SCBCdrEmailProperties	225
<i>Description</i>	225
<i>Properties</i>	226
Disable moving downloaded messages to the deleted items folder	226

SCBCdrLicenseInfoAccess	227
<i>Description</i>	227
<i>Methods</i>	227
Cedar Verifier Component Library	231
SCBCdrVerificationForm	231
<i>Description</i>	231
<i>Methods and Properties</i>	231
SCBCdrVerificationField.....	233
<i>Description</i>	233
<i>Type Definitions</i>	233
<i>Methods and Properties</i>	233
SCBCdrVerificationTable.....	241
<i>Description</i>	241
<i>Methods and Properties</i>	241
SCBCdrVerificationButton	242
<i>Description</i>	242
<i>Methods and Properties</i>	242
SCBCdrVerificationLabel.....	243
<i>Description</i>	243
<i>Properties</i>	243
Password Encryption for Database Connection Strings	248
Master Project Side (Project Primary Developer)	248
<i>About encryption keys</i>	248
Advanced troubleshooting.....	250

Overview

This guide provides a reference to the events you can use to implement custom projects and business solutions using Perceptive Intelligent Capture products.

Script Event Reference

VerifierFormLoadEvent

To implement the script handler of this event, complete the following steps.

In Perceptive Intelligent Capture Designer, load the project file.

In **Definition** mode, select the project node and open the **Script Editor**.

In the **Script View for Project** dialog box, in the **Object** list, select **Script Module**, and in the **Proc** list, select **VerifierFormLoad**.

The following simple implementation of the `VerifierFormLoad` event, (in this simple case non-optionally) replaces the standard form `Form_Invoices_1` with a custom `Form_Invoices_2` defined for the same document class.

Example

```
Option Explicit
'Project Level Script Code
Private Sub ScriptModule_VerifierFormLoad(pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc,
FormClassName As String, FormName As String)
    FormClassName = "Invoices"
    FormName = "Form_Invoices_2"
End Sub
```

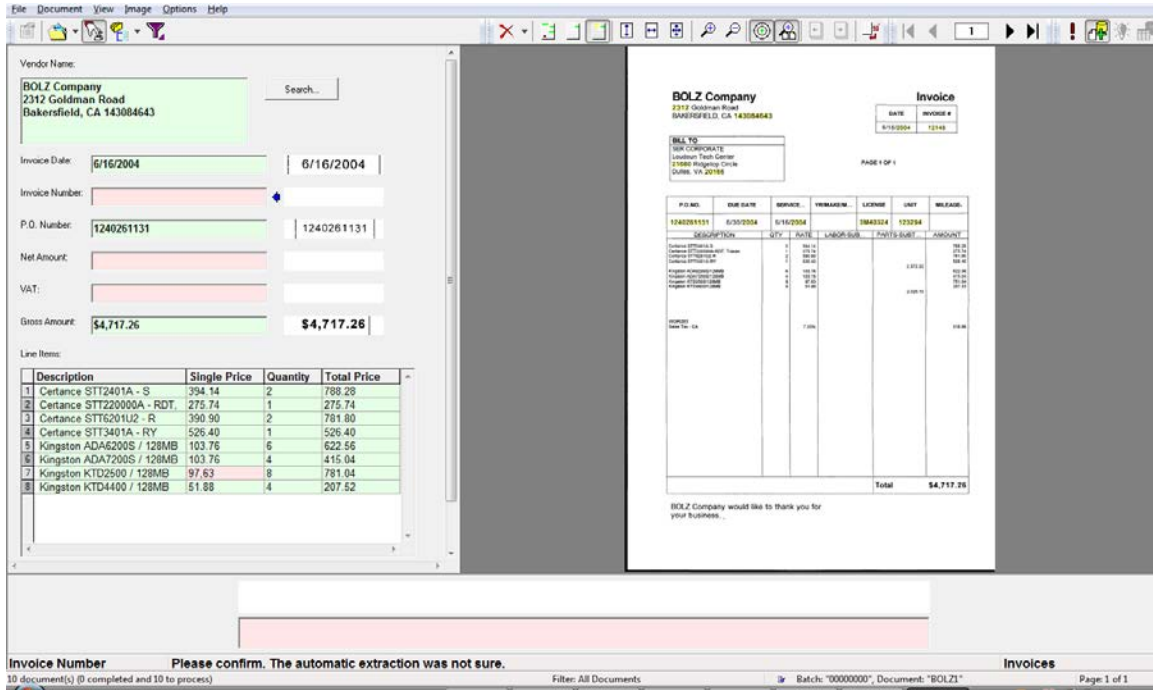
As a result, Verifier application always loads the simple second form specified in the script.

If the script modifies the form and the form's class references incorrectly, a warning message displays to the Verifier user. For example, an incorrect script modification can occur when a reference is made to a non-existing verification form of a class or when the form does not exist in the specified class.

Example

```
Private Sub ScriptModule_VerifierFormLoad(pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc,
FormClassName As String, FormName As String)
    FormClassName = "Non-existing class name"
    FormName = "Non-existing form name"
End Sub
```

Then the application loads the standard verification form, the one that the application would load anyway if the script handler of `VerifierFormLoad` event did not exist, instead of the incorrect one proposed by the custom script.



If you enable **Allow firing of VerifierFormLoad event when in Verifier Test/Train modes**, the system triggers this event in Designer Verifier Test and Train modes.

In Designer, if you enable **Allow firing of FocusChanged event when loading the verification form**, the system fires the document-class level `FocusChanged` event with the `Reason` parameter set to `CdrBeforeFormLoaded` before the desired verification form loads, but after the `VerificationFormLoad` event described above.

Below is an example of a script that shows how you can implement the handler of this `Reason` in the Perceptive Intelligent Capture custom script.

Example

```
Private Sub Document_FocusChanged(pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc, ByVal Reason As SCBCdrPROJLib.CdrFocusChangeReason, ByVal OldFieldIndex As Long, pNewFieldIndex As Long)
    If Reason = CdrBeforeFormLoaded Then
        MsgBox "The form has not been loaded yet"
    End If
End Sub
```


Usage

You can use the features described in this section for many different purposes, including the following examples.

- To optionally load a non-standard verification form in accordance with some parameters of the processed document.
- To translate the content of verification forms dynamically into a different language or simply to load the required verification form in accordance with the current system Regional settings.
- To display a specific page of a document instead of the first one.

ScriptModule


Cedar ScriptModule Event Interface

Project events are specific for one Perceptive Intelligent Capture Project. However, within a Perceptive Intelligent Capture Project, all documents and fields share the same implementation of these events. This means that they are document class (DocClass) independent. As the Project events belong to the “sheet” ScriptModule, all events start with the prefix ScriptModule.

Create a script for document export

The following script writes classified images to subdirectories in the export directory. Each subdirectory holds documents from one class only; the subdirectory name corresponds to the class name.

To create the script, complete the following steps.

1. In Designer, switch to **Definition Mode**.
2. On the toolbar, click **Show/hide script**  button.
3. In the **Script View for Project** dialog box, from the **Object** list, select **ScriptModule**.
4. From the **Proc** list, select **ExportDocument**.
Note This generates the outline of a subroutine.
5. Add the following code.

```
Dim sNewPath As String
Dim MyImage As SCBCroImage
Dim NewFileName As String
sNewPath=ExportPath & "\" & pWorkdoc.DocClassName 'Set directory name
Set MyImage=pWorkdoc.Image(0) 'Access the image file to the current WorkDoc
On Error GoTo Skip 'Skip next step if directory exists
MkDir sNewPath 'Create directory
Skip:
NewFileName=Mid(MyImage.FileName, InStrRev(MyImage.FileName,"\")) 'Set file name
MyImage.SaveFile sNewPath & NewFileName 'Save file to directory
```

6. Close the dialog box.
7. Switch to **Runtime Mode** and test the script.

Methods and Properties

AppendWorkdoc

This event is fired when processing a document separation workflow step in Runtime Server. It can be used to append a given Workdoc after the last Workdoc on the base of `CdrMPTType`.

Description	Definition	
Syntax	<code>ScriptModule_AppendWorkdoc (pLastWorkdoc As ISCBCdrWorkdoc, pCurrentWorkdoc As ISCBCdrWorkdoc, pAppendType As CdrMPTType)</code>	
Parameters	<i>pLastWorkdoc</i>	Last Workdoc
	<i>pCurrentWorkdoc</i>	Current Workdoc
	<i>pAppendType</i>	Defines the possible results of the multipage classification. For example, <code>CdrAttachmentPage</code> would mean that the engine has classified the page as an attachment or <code>CdrFirstPage</code> would mean that the engine has classified this page as the start of a new document.

BatchClose

This event launches when the Verifier user exits a batch in one of the following methods.

- When verifying a batch and selecting Return to batch list.
- Batch Verification Completion.
- Partial Batch verification completion.
- The user quits the verifier applications while in a batch.

The event is triggered in the Verifier Thick Client and the Web Verifier applications.

Description	Definition	
Syntax	<code>ScriptModule_BatchClose (ByVal UserName As String, ByVal BatchDatabaseID As Long, ByVal ExternalGroupID As Long, ByVal ExternalBatchID As String, ByVal TransactionID As Long, ByVal WorkflowType As SCBCdrPROJLib.CDRDatabaseWorkflowTypes, BatchState As Long, BatchReleaseAction As SCBCdrPROJLib.CDRBatchReleaseAction)</code>	
Parameters	<i>UserName:</i>	The currently logged in user name who has closed the batch.
	<i>BatchDatabaseID:</i>	The unique Batch ID within the database. For the File System, this batch ID is not used.

Description	Definition
	<p><i>ExternalGroupID:</i> The Group ID that can be assigned to a batch.</p> <p>The Group ID that can be used with the Scripting security methods that enable the developer to assign a batch to a security group. Only those users belonging to the same Group ID are able to access batches.</p> <p>For example, a batch belonging to Group ID 80 is only accessible by a user who is assigned to group 80.</p> <p>Read or Write Parameter that can be modified to any long value.</p>
	<p><i>ExternalBatchID:</i> The External Batch ID can be assigned to a batch.</p> <p>The External Batch ID allows the developer to synchronize a newly created batch of documents with another external system. For example, an archiver or a storage box ID.</p> <p>Read or Write Parameter that can be modified to any long value.</p>
	<p><i>TransactionID:</i> The Transaction ID can be assigned to a batch.</p> <p>The Transaction ID allows the developer to synchronize a newly created batch of documents with another external system, such as an archiver or a storage box ID.</p> <p>Read or Write Parameter that can be modified to any long value.</p>
	<p><i>WorkflowType:</i> Corresponds to CDRDatabaseWorkflowTypes data type.</p>
	<p><i>BatchState:</i> The current status of the batch being opened, such as status 550 (Extraction Verification).</p>
	<p><i>BatchReleaseAction:</i> Batch Release Action represents the action taken when the last document of the batch has been verified. The parameter can be set or read from script. By default, it is always set to CDRBatchReleaseActionUserDefined (as the user always makes a selection). If registry value is used to hide the batch release dialog box in Verifier thick client, then the last action taken prior to the dialog box being hidden is the one showing in this parameter.</p> <p>The scripter can set an override value to this parameter, such as every time batch verification completes, always goes to the next invalid batch.</p> <p><i>CdrBatchReleaseActionCancel</i> –return to current batch and last document verified.</p> <p><i>CdrBatchReleaseActionReturnToBatchList</i> – return to batch list.</p> <p><i>CdrBatchReleaseActionUndefined</i> – unknown.</p> <p><i>CdrBatchReleaseActionUserDefined</i> – default, user makes a selection on next action to take on batch release.</p> <p><i>CdrBatchReleaseActionVerifyNextInvalidBatch</i> – open next batch to verify.</p> <p><i>CdrBatchReleaseActionVerifyNextInvalidState</i> – open current batch to verify in the next invalid state.</p>

Description	Definition
Example	<pre>Private Sub ScriptModule_BatchClose(ByVal UserName As String, ByVal BatchDatabaseID As Long, ByVal ExternalGroupID As Long, ByVal ExternalBatchID As String, ByVal TransactionID As Long, ByVal WorkflowType As SCBCdrPROJLib.CDRDatabaseWorkflowTypes, BatchState As Long, BatchReleaseAction As SCBCdrPROJLib.CDRBatchReleaseAction) Call LogMessage(BatchDatabaseID & ", " & UserName, "C:\EventTrace.Log") End Sub</pre>

For additional information, see `BatchOpen`, `Project Event`, `PostImportBatch`, and `CDRDatabaseWorkflowTypes`.

BatchOpen

This event triggers when the user opens a batch.

Description	Definition	
Syntax	<pre>ScriptModule_BatchOpen(ByVal UserName As String, ByVal BatchDatabaseID As Long, ByVal ExternalGroupID As Long, ByVal ExternalBatchID As String, ByVal TransactionID As Long, ByVal WorkflowType As SCBCdrPROJLib.CDRDatabaseWorkflowTypes, BatchState As Long)</pre>	
Parameters	<i>UserName:</i>	The Username currently logged in who has opened the batch.
	<i>BatchDatabaseID:</i>	<p>The unique Batch ID within the database. For the File System, this batch ID is not used.</p> <p>The Batch ID displays as a numeric value. For example, Batch 00000061 the value 61 is returned.</p>
	<i>ExternalGroupID:</i>	<p>The Group ID that can be assigned to a batch.</p> <p>The Group ID that can be used with the Scripting security methods that enable the developer to assign a batch a security group. Only those users belonging to the same Group ID are able to access batches.</p> <p>For example, a batch belonging to Group ID 80 is only accessible by a user who is assigned to group 80.</p> <p>Read or Write Parameter that can be modified to any long value.</p>
	<i>ExternalBatchID:</i>	<p>The External Batch ID can be assigned to a batch.</p> <p>The External Batch ID allows the developer to synchronize a newly created batch of documents with another external system, such as archive ID or a storage box ID.</p> <p>Read or Write Parameter that can be modified to any long value.</p>

Description	Definition	
	<i>TransactionID:</i>	<p>The Transaction ID can be assigned to a batch.</p> <p>The Transaction ID allows the developer to synchronize a newly created batch of documents with another external system. For example, an archive ID or a storage box ID.</p> <p>Read or Write Parameter that can be modified to any long value.</p>
	<i>WorkflowType:</i>	Corresponds to CDRDatabaseWorkflowTypes data type.
	<i>BatchState:</i>	The current status of the batch being opened, such as status 550 (Extraction Verification).
Example	<p>The following example logs the Batch ID and User name that opened a batch with date and time.</p> <p>LogMessage is a custom function that writes a text line into a log file with Date/Time as a prefix.</p> <pre>Private Sub ScriptModule_BatchOpen(ByVal UserName As String, ByVal BatchDatabaseID As Long, ByVal ExternalGroupID As Long, ByVal ExternalBatchID As String, ByVal TransactionID As Long, ByVal WorkflowType As SCBCdrPROJLib.CDRDatabaseWorkflowTypes, BatchState As Long) Call LogMessage(BatchDatabaseID & ", " & UserName, "C:\EventTrace_Log") End Sub</pre>	

For additional information, see `BatchClose`, `Project Event`, `PostImportBatch`, and `CDRDatabaseWorkflowTypes`.

ExportDocument

ExportDocument allows you to implement a customer specific export of all extracted data. This provides the ability to implement a customer-specific export of all extracted data.

Description	Definition	
Syntax	<code>ScriptModule_ExportDocument (pWorkdoc As ISCBCdrWorkdoc, ExportPath As String, pCancel As Boolean)</code>	
Parameters	<i>pWorkdoc</i>	Workdoc that should be exported
	<i>ExportPath</i>	Export path, which was configured within the Runtime Server settings (no changes possible)
	<i>pCancel</i>	Set this variable to TRUE to cancel the export.

ForceClassificationReview

In the application, the PostClassify event is extended so that it can force a manual classification review even if the classification succeeded.

Description	Definition
Attribute	Read/Write
Example	<p>The following script sample shows how the manual classification process can be forced from custom script event PostClassify.</p> <pre>Private Sub ScriptModule_PostClassify(pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc) If pWorkdoc.DocClassName = "VeryImportantClass" Then pWorkdoc.ForceClassificationReview = True End If End Sub</pre>

For additional information, see [PostClassify](#).

Initialize

The initialize event is called when a batch is opened for processing.

Description	Definition		
Syntax	ScriptModule_Initialize (ModuleName As String)		
Parameters	<table border="1"> <tr> <td><i>ModuleName:</i></td> <td>Name of the current module, allowed values: 'Server', 'Designer', 'Verifier', Thin Client Verifier.</td> </tr> </table>	<i>ModuleName:</i>	Name of the current module, allowed values: 'Server', 'Designer', 'Verifier', Thin Client Verifier.
<i>ModuleName:</i>	Name of the current module, allowed values: 'Server', 'Designer', 'Verifier', Thin Client Verifier.		
Example	<pre>Public Sub ScriptModule_Initialize(ByVal ModuleName As String) DBname=Project.FileName DBname=Left(DBname, InStrRev(DBname, '\\')) & 'InvoiceBestellNo.mdb' Set DB=OpenDatabase(DBname) End Sub</pre>		

MoveDocumentThis event is launched when the Verifier / Web Verifier User places a document in Exception and the document is moved out of the batch.

The ScriptModule provides the following event information: Old Batch ID, New Batch ID, Reason, Document state. For the event to be triggered, the condition must be set within the application settings that a new exception batch is created when a user places a document to exception.

The event triggers for each document that is placed into exception within a single batch.

After placing a document to Exception, the event is triggered if:

- Batch Verification is completed and all other documents have been verified or placed in exception.
- The user returns to the batch list after placing the document into Exception.

Description	Definition	
Syntax	<pre>ScriptModule_MoveDocument(pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc, ByVal OldBatchID As String, ByVal NewBatchID As String, ByVal Reason As SCBCdrPROJLib.CDRMoveDocumentReason)</pre>	
Parameters	pWorkdoc:	The Workdoc Object that is being used. No changes can be made to the workdoc within this event.
	OldBatchID:	The batch ID to which the document belonged prior to placing a document to exception.
	NewBatchID:	The new batch ID to which the document is moving after the document is placed in Exception.
	Reason:	The reason the event is triggered. The only reason implemented at this point is for the document moved to exception(CDRMoveDocumentToExceptionBatch).
	DocState:	The workflow state of the document
Example	<p>The following example logs a general message for each document placed into exception, showing the old batch ID and the new batch ID.</p> <pre>Private Sub ScriptModule_MoveDocument(pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc, ByVal OldBatchID As String, ByVal NewBatchID As String, ByVal Reason As SCBCdrPROJLib.CDRMoveDocumentReason) If Reason = CDRMoveDocumentToExceptionBatch Then Project.LogScriptMessageEx CDRTYPEINFO, CDRSeveritySystemMonitoring, " Document [" & pWorkdoc.Filename & "] has been moved from Verifier batch [" & OldBatchID & "] to exception batch [" & NewBatchID & "]" Project.LogScriptMessageEx CDRTYPEINFO, CDRSeveritySystemMonitoring, " Current document state is [" & CStr(pWorkdoc.CurrentBatchState) & "]" End If End Sub</pre>	

For more information, see `ProjectEvent`.

PostClassify

The PostClassify event is called after all defined classification methods are executed by the Cedar Project.

Description	Definition	
Syntax	ScriptModule_PostClassify (pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc)	
Parameters	pWorkdoc:	Workdoc object that has been classified
Example	<pre> Private Sub ScriptModule_PostClassify(pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc) Dim imgDocument As SCBCroImage Dim lngTagCount As Long 'Imprint number is stored as a TiffTag in the image file - the following code extracts the TiffTag information and sets the field value. 'NOTE: this will only work if there is a single TiffTag - would require modification for more! Set imgDocument = pWorkdoc.Image(0) lngTagCount = imgDocument.TiffTagCount 'Check that there is at least 1 tiffTag. If (lngTagCount > 0) Then Dim intImageCount As Integer Dim intImageCounter As Integer intImageCount=pWorkdoc.PageCount'Get the number of pages in TIF Dim imgCollection() As SCBCroImage ReDim imgCollection(intImageCount) 'Set an image collection variable to store all the pages of the image 'Store all pages of TIF image onto a temporary image collection array For intImageCounter=0 To intImageCount-1 Set imgCollection(intImageCounter)=pWorkdoc.Image(intImageCounter) Next Dim strTag As String strTag = CStr(Format(Now(), "yyyymmddhhMMss")) & "123456" 'Set the Info to place into TIF Tag imgCollection(0).TiffTagClearAll 'Clear All TIF Tags imgCollection(0).TiffTagAddASCII 33601, strTag 'Add the TIF Tag imgCollection(0).SaveFile(pWorkdoc.DocFileName(0)) 'Save modified image collection with TIF Tag and overwrite existing image </pre>	

Description	Definition
	<pre> 'Reset the collection to the new image in workdoc For intImageCounter=1 To intImageCount-1 imgCollection(intImageCounter).AppendToMultiImageFile(pWorkdoc.DocFileName(0)) Next MsgBox("Tag = " & imgDocument.TiffTagString(lngTagCount)) 'Message box to show TIF Tag Else ' If there is no Tifftag, can set the field to false - no Tifftag means that something ' has gone wrong with scanning. Generate a new Doc ID. MsgBox("No Tag") End If End Sub </pre>
<p>Example</p>	<pre> Private Sub ScriptModule_PostClassify(pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc) Dim imgDocument As SCBCroImage Dim lngTagID as long lngTagID = 12345 Set imgDocument = pWorkdoc.Image(0) Call fnCreateTiffTag(imgDocument, lngTagID, "Test") End Sub </pre>

PostImportBatch

This is an event that is triggered by the Runtime Server after it has finished importing the batch. Use this event to set batch properties such as batch name and external group ID.

Description	Definition	
<p>Syntax</p>	<pre> ScriptModule_PostImportBatch(ByVal BatchDatabaseID As Long, BatchName As String, Priority As Long, State As Long, ExternalGroupID As Long, ExternalBatchID As String, TransactionID As Long, TransactionType As Long) </pre>	
<p>Parameters</p>	<p>BatchDatabaseID:</p>	<p>The unique Batch ID from the database. This would be a numeric ID corresponding to the BatchID within the database tables. Read Only Parameter that cannot be modified.</p>

Description	Definition	
	BatchName:	<p>The Batch Name that is assigned by the Runtime Server instance. The name is taken from the Import settings of the Runtime Server instance.</p> <p>Read or Write Parameter that can be modified to any string value.</p>
	Priority:	<p>The Batch priority that is assigned by the Runtime Server instance. The priority is taken from the Import settings of the Runtime Server instance.</p> <p>Read or Write Parameter that can be modified to any long value between 1 to 9.</p>
	State:	<p>The Batch State that is assigned by the Runtime Server instance. The status is taken from the Workflow settings of the Runtime Server instance – Import Success.</p> <p>Read or Write Parameter that can be modified to any long value between 100 and 999.</p>
	ExternalGroupID:	<p>The Group ID that can be assigned to a batch.</p> <p>The Group ID can be used with the new Scripting security methods that enable the developer to assign a batch to a security group. Only those users belonging to the same Group ID are able to access batches.</p> <p>For example, a batch belonging to Group ID 80 is only accessible by a user who is assigned to group 80.</p> <p>Read or Write Parameter that can be modified to any numeric value.</p>
	ExternalBatchID:	<p>The External Batch ID can be assigned to a batch.</p> <p>The External Batch ID allows the developer to synchronize a newly created batch of documents with another external system. For example, a storage box ID.</p> <p>Read or Write Parameter that can be modified to any numeric value.</p>
	TransactionID:	<p>The Transaction ID can be assigned to a batch.</p> <p>The Transaction ID allows the developer to synchronize a newly created batch of documents with another external system. It can be used to identify originators of batch of documents.</p> <p>Read or Write Parameter that can be modified to any long value.</p>
	TransactionType:	<p>The Transaction Type can be assigned to a batch.</p> <p>The Transaction Type allows the developer to synchronize a newly created batch of documents with another external system. It can used to identify the types of documents (Invoices, Claim forms etc.) in batches or source of document (Email, Scanned etc.)</p> <p>Read or Write Parameter that can be modified to any long value.</p>

Description	Definition
<p>Example</p>	<p>The example below updates batch priorities after import has been done. It changes the name, state and add a group ID as well as transaction type and ID.</p> <pre> Private Sub ScriptModule_PostImportBatch(ByVal BatchDatabaseID As Long, BatchName As String, Priority As Long, State As Long, ExternalGroupID As Long, ExternalBatchID As String, TransactionID As Long, TransactionType As Long) 'Set batch priorities after import BatchName = "AP Batch_" & CStr(BatchDatabaseID) Priority = 2 State = 102 ExternalGroupID = 777 TransactionType = 10 TransactionID = 2 End Sub </pre>

For additional information, see `ScriptModuleEvents`, `SecurityUpdateStart`, `SecurityUpdateAddUserGroup` and `SecurityUpdateCommit`.

PreClassify

This event is called before any defined classification method is executed by the Cedara Project. During this event, it is possible to apply an existing name of a `DocClass` to the `Workdoc`.

Description	Definition	
Syntax	<code>ScriptModule_PreClassify (pWorkdoc As SCBCdrWorkdoc)</code>	
Parameters	<i>pWorkdoc:</i>	Workdoc object that should be classified
Example	<pre> Private Sub ScriptModule_PreClassify(pWorkdoc As SCBCdrWorkdoc) if (DoSomeMagic(pWorkdoc) = TRUE) then 'assign "Invoice" as result of the classification pWorkdoc.DocClassName = "Invoice" else 'do nothing and continue with normal classification end if End Sub </pre>	

PreClassifyAnalysis

This event is fired between the `PreClassify` and `PostClassify` events that identify the beginning and end of the classification workflow step for a particular document. Using this event the custom script can clean-up and extend classification results before the final decision is made by the system and before the final classification matrix is built.

ProcessBatch

The `ProcessBatch` event is launched when the Runtime Server instance begins processing during the Custom Processing workflow step.

Description	Definition	
Syntax	<code>ScriptModule_ProcessBatch(pBatch As SCBCdrPROJLib.ISCBCdrBatch, ByVal InputState As Long, DesiredOutputStateSucceeded As Long, DesiredOutputStateFailed As Long)</code>	
Parameters	<i>pBatch:</i>	The Batch Object that is being processed.
	<i>InputState:</i>	The input state of the batch when Custom Processing was activated on it.
	<i>DesiredOutputStateSucceeded:</i>	The output state of the batch if the workflow step succeeds.
	<i>DesiredOutputStateFailed:</i>	The output state of the batch if the workflow step failed.
Example	<p>The following script should be added to the very beginning of the <code>ProcessBatch</code> event: This script helps to stop an indefinite looping process of state 0 batches.</p> <p>This script does not set batches to special state 987. The script repairs a batch and stops looping of the custom processing step. Note that it is not possible to set the batch state to something other than zero for a batch with no documents because batch state is by definition the lowest state of all enclosed documents. If the number of documents is zero, the application just uses the default value, which is zero.</p>	

Description	Definition
(Continued)	<p>Enhanced recovery script sample:</p> <pre> Private Sub ScriptModule_ProcessBatch(pBatch As SCBCdrPROJLib.ISCBCdrBatch, ByVal InputState As Long, DesiredOutputStateSucceeded As Long, DesiredOutputStateFailed As Long) Dim lFolderIndex As Long Dim lDocIndex As Long Dim theWorkdoc As SCBCdrWorkdoc Dim vLoadingCompletenessStatus As Variant Dim lStatus As Long Dim bNeedSafetyRestart As Boolean Dim strWorkdocName As String Dim theImage As SCBCroImage On Error GoTo LABEL_ERROR pBatch.BatchPriority = 3 '[AE] [2012-03-27] Boost priority for state zero documents Project.LogScriptMessageEx CDRTypeInfo, CDRSeveritySystemMonitoring, "ScriptModule_ProcessBatch starting, batch <" & CStr(pBatch.BatchID) & ">, new state <" & CStr(DesiredOutputStateSucceeded) & ">" If ScriptModule.ModuleName <> "Server" Then Exit Sub For lFolderIndex = pBatch.FolderCount - 1 To 0 Step -1 If pBatch.FolderDocCount (lFolderIndex) = 0 Then Project.LogScriptMessageEx CDRTypeWarning, CDRSeveritySystemMonitoring, "Removed folder with zero documents from batch [" & pBatch.BatchID & "]" pBatch.DeleteFolder(lFolderIndex, False) End If Next lFolderIndex If pBatch.FolderCount = 0 Then Project.LogScriptMessageEx CDRTypeWarning, CDRSeveritySystemMonitoring, "Detected batch with zero folders: [" & pBatch.BatchID & "]" pBatch.BatchState = 987 End If On Error Resume Next For lFolderIndex = 0 To pBatch.FolderCount-1 Step 1 For lDocIndex = pBatch.FolderDocCount(lFolderIndex) - 1 To 0 Step -1 If pBatch.FolderDocState(lFolderIndex, lDocIndex) = InputState Then Err.Clear bNeedSafetyRestart = False strWorkdocName = pBatch.FolderWorkdocFileName (lFolderIndex, lDocIndex, False) Set theWorkdoc = pBatch.LoadWorkdoc(lFolderIndex, lDocIndex) Project.LogScriptMessageEx CDRTypeInfo, CDRSeveritySystemMonitoring, "Loading of zero state Workdoc [" & strWorkdocName & "]" proceeded with error number [" & CStr(Err.Number) & "]" and error description [" & Err.Description & "]" "Detected batch with zero folders: [" & pBatch.BatchID & "]" pBatch.BatchState = 987 End If On Error Resume Next </pre>

Description	Definition
(Continued)	<pre> For lDocIndex = pBatch.FolderDocCount(lFolderIndex) - 1 To 0 Step -1 If pBatch.FolderDocState(lFolderIndex, lDocIndex) = InputState Then Err.Clear bNeedSafetyRestart = False strWorkdocName = pBatch.FolderWorkdocFileName (lFolderIndex, lDocIndex, False) Set theWorkdoc = pBatch.LoadWorkdoc(lFolderIndex, lDocIndex) Project.LogScriptMessageEx CDRTYPEINFO, CDRSeveritySystemMonitoring, "Loading of zero state Workdoc [" & strWorkdocName & "] proceeded with error number [" & CStr(Err.Number) & "] and error description [" & Err.Description & "]" lStatus = 1001 If Err.Number = 0 Then vLoadingCompletenessStatus = theWorkdoc.NamedProperty("LoadingCompletenessStatus") lStatus = vLoadingCompletenessStatus End If For lFolderIndex = 0 To pBatch.FolderCount-1 Step 1 If Err.Number <> 0 Or lStatus > 0 Then bNeedSafetyRestart = True Project.LogScriptMessageEx CDRTYPEWARNING, CDRSeverityEmailNotification, "True corruption case detected for Workdoc [" & strWorkdocName & "] with stream exit code [" & CStr(lStatus) & "]" End If Project.LogScriptMessageEx CDRTYPEINFO, CDRSeveritySystemMonitoring, "PreErrorChecks: Loading return code is {" & CStr(Err.Number) & "} and loading status is {" & CStr(lStatus) & }" If (lStatus > 0 And lStatus <= 700) Then ' if this value is > 700 but <= 790, then re-OCR is required, if it is greater than 790, then re-importing is needed - extend the script below to set a different output state, other than the standard "DesiredOutputStateSucceeded" one Project.LogScriptMessageEx CDRTYPEINFO, CDRSeveritySystemMonitoring, "Loading return code is {" & CStr(Err.Number) & "} and loading status is {" & CStr(lStatus) & }" Project.LogScriptMessageEx CDRTYPEINFO, CDRSeveritySystemMonitoring, "Ignoring internal error when loading Workdoc [" & theWorkdoc.FileName & "]" Err.Clear theWorkdoc.DocClassName = "" theWorkdoc.Fields.Clear theWorkdoc.RebuildBasicObjects If Err.Number <> 0 Then Project.LogScriptMessageEx CDRTYPEWARNING, CDRSeveritySystemMonitoring, "Recovery script: RebuildBasicObjects failed with error code [" & CStr(Err.Number) & "] and error description [" & Err.Description & "]" Err.Clear Project.LogScriptMessageEx CDRTYPEWARNING, CDRSeveritySystemMonitoring, "Recovery script: Proceeding with attempt to redirecting document to re-OCR state" ' [AE] [2012-02-27] DesiredOutputStateSucceeded = 100 ' [AE] [2012-02-27] theWorkdoc.DocState = CDRDocStateHaveDocs ' [AE] [2012-02-28] This call internally triggeres invoking of ".InternalClear(false,true) End If pBatch.FolderDocState(lFolderIndex, lDocIndex) = DesiredOutputStateSucceeded </pre>

Description	Definition
(Continued)	<pre> If Err.Number <> 0 Then Project.LogScriptMessageEx CDRTYPEError, CDRSeveritySystemMonitoring, "Recovery script: put_FolderDocState failed with error code [" & CStr(Err.Number) & "] and error description [" & Err.Description & "]" Err.Clear End If pBatch.UpdateDocument(theWorkdoc, lFolderIndex, lDocIndex) If Err.Number <> 0 Then Project.LogScriptMessageEx CDRTYPEError, CDRSeveritySystemMonitoring, "Recovery script: UpdateDocument failed with error code [" & CStr(Err.Number) & "] and error description [" & Err.Description & "]" Err.Clear End If End If If Err.Number <> 0 Or (lStatus > 700 And lStatus <= 790) Then ' if this value is > 700 but <= 790, then re-OCR is required, if it is greater than 790, then re-importing is needed - extend the script below to set a different output state, other than the standard "DesiredOutputStateSucceeded" one Project.LogScriptMessageEx CDRTYPEInfo, CDRSeveritySystemMonitoring, "Loading return code is {" & CStr(Err.Number) & "} and loading status is {" & CStr(lStatus) & "}" Project.LogScriptMessageEx CDRTYPEInfo, CDRSeveritySystemMonitoring, "Ignoring internal error when loading Workdoc [" & theWorkdoc.Filename & "]" Err.Clear DesiredOutputStateSucceeded = 100 theWorkdoc.DocState = CDRDocStateHaveDocs ' [AE] [2012-02-28] This call internally triggeres invoking of ".InternalClear(false,true) pBatch.FolderDocState(lFolderIndex, lDocIndex) = DesiredOutputStateSucceeded If Err.Number <> 0 Then Project.LogScriptMessageEx CDRTYPEError, CDRSeveritySystemMonitoring, "Recovery script: put_FolderDocState failed with error code [" & CStr(Err.Number) & "] and error description [" & Err.Description & "]" Err.Clear End If pBatch.UpdateDocument(theWorkdoc, lFolderIndex, lDocIndex) If Err.Number <> 0 Then Project.LogScriptMessageEx CDRTYPEError, CDRSeveritySystemMonitoring, "Recovery script: UpdateDocument failed with error code [" & CStr(Err.Number) & "] and error description [" & Err.Description & "]" Err.Clear End If End If ' [AE] [2012-03-05] Test that recovery has been succeeded and the Workdoc can now be loaded with no issues. This is one extra safety solution: "Load document one more time to "test" and recover for (from) real document file corruptions". If lStatus > 0 And lStatus <= 790 Then Set theWorkdoc = Nothing Err.Clear Set theWorkdoc = pBatch.LoadWorkdoc(lFolderIndex, lDocIndex) vLoadingCompletenessStatus = theWorkdoc.NamedProperty("LoadingCompletenessStatus") lStatus = vLoadingCompletenessStatus If Err.Number <> 0 Or lStatus > 0 Then lStatus = 799 End If End If End If </pre>

Description	Definition
(Continued)	<pre> ' [AE] [2012-03-27] Additional check for consistency of loaded document files If lStatus = 0 Then Err.Clear Set theImage = theWorkdoc.Pages(0).Image(0) If Err.Number <> 0 Or theImage Is Nothing Then lStatus = 999 bNeedSafetyRestart = True End If End If If Err.Number <> 0 Or (lStatus > 790) Then ' if this value is > 700 but <= 790, then re-OCR is required, if it is greater than 790, then re-importing is needed - extend the script below to set a different output state, other than the standard "DesiredOutputStateSucceeded" one Project.LogScriptMessageEx CDRTypeInfo, CDRSeveritySystemMonitoring, "Loading return code is {" & CStr(Err.Number) & "} and loading status is {" & CStr(lStatus) & "}" Project.LogScriptMessageEx CDRTypeInfo, CDRSeveritySystemMonitoring, "Ignoring internal error when loading Workdoc [" & theWorkdoc.Filename & "]" Project.LogScriptMessageEx CDRTypeWarning, CDRSeverityEmailNotification, "Document [" & strWorkdocName & "] with stream exit code [" & CStr (lStatus) & "] will be redirected to manual processing state" Err.Clear DesiredOutputStateSucceeded = 850 pBatch.FolderDocState(lFolderIndex, lDocIndex) = DesiredOutputStateSucceeded If Err.Number <> 0 Then Project.LogScriptMessageEx CDRTypeError, CDRSeveritySystemMonitoring, "Recovery script: put_FolderDocState failed with error code [" & CStr(Err.Number) & "] and error description [" & Err.Description & "]" Err.Clear End If ' [AE] [2012-03-27] Do not call update document in case of 850 type recovery - just update the document state via the call above ` pBatch.UpdateDocument(theWorkdoc, lFolderIndex, lDocIndex) ` If Err.Number <> 0 Then ` Project.LogScriptMessageEx CDRTypeError, CDRSeveritySystemMonitoring, ` "Recovery script: UpdateDocument failed with error code [" & ` CStr(Err.Number) & "] and error description [" & Err.Description & "]" ` Err.Clear ` End If End If Set theWorkdoc = Nothing ' [AE] [2012-03-05] Auto-apply the RTS instance restart after recovering every single case of true document loading failure. This is to ensure that corruption's side effects are not cumulated across multiple auto-recovered documents and clean documents are not negatively affected by attempts to load a corrupted one. If bNeedSafetyRestart = True Then Project.PerformScriptCommandRTS(1, 0, 0, "Applying safety recovery restart") GoTo LABEL_SUCCESS End If End If </pre>

Description	Definition
(Continued)	<pre> Next lDocIndex Next lFolderIndex LABEL_SUCCESS: Project.LogScriptMessageEx CDRTTypeInfo, CDRSeveritySystemMonitoring, "ScriptModule_ProcessBatch finished successfully, batch <" & CStr(pBatch.BatchID) & ">, new state <" & CStr(DesiredOutputStateSucceeded) & ">, old state <" & CStr(InputState) & ">" Exit Sub LABEL_ERROR: Project.LogScriptMessageEx CDRTTypeError, CDRSeveritySystemMonitoring, "ScriptModule_ProcessBatch, finished with Error: " & Err.Description End Sub </pre>
	<p>Use the corresponding Terminate Event script instead to delete these empty batches. Do not use both scripts within one project, because the Terminate Event script will make it impossible to load the ProcessBatch script.</p>

For additional information, see `Project Event`.

RouteDocument

This event is launched when a document has been extracted.

Definition	Description	
Syntax	<code>ScriptModule_RouteDocument (pWorkdoc As ISCBCdrWorkdoc, State As Single)</code>	
Parameters	<i>pWorkdoc:</i>	Workdoc object that was classified and extracted
	<i>State:</i>	This parameter contains the current state that is assigned to the Workdoc. Value can be changed from the script.

Definition	Description
<p>Example</p>	<pre> Private Sub ScriptModule_RouteDocument(pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc, State As Integer) If pWorkdoc.Fields("Field1").Valid = FALSE then 'route to 500 if Field1 is not valid State = 500 Exit sub End if If pWorkdoc.Fields("Field2").Valid = FALSE then 'route to 520 if Field2 is not valid State = 520 Exit sub End if 'else use default state End Sub </pre> <p>For example, in an environment where the Batch folder is shared between multiple organisations (either country groups, or departments), it is possible to allocate verifiers their own workflow configurations.</p> <p>The following script automatically sets the Batch status after extraction to a status that is country based (such as, GB is status 550, Germany is status 551).</p> <pre> Private Sub ScriptModule_RouteDocument(pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc, State As Integer) 'If the batch state is 550 and document is not in verifier If State = 550 And Not fnIsVerifier() Then 'Check country code and set batch status Select Case CountryCode Case "GB" State = 550 Case "DE" State = 551 Case "BENL" State = 552 Case "IE" State = 553 Case "RU" State = 554 Case "US" State = 555 Case Else State = 550 End Select 'Save the work doc after changing document status pWorkdoc.Save(pWorkdoc.Filename, "") End If End Sub </pre>

SecurityUpdateAddUserGroup and SecurityUpdateAddUserGroupPwd

This method is used to update or add the database security credentials. This script call is used in creating or updating the Perceptive Intelligent Capture users, roles, and groups.

When updating the security policy of Perceptive Intelligent Capture through a custom script, only the database tables are updated. The project security is not modified after a script update.

- Use the *SecurityUpdateAddUserGroupPwd* method to import user accounts with predefined passwords.
- Use this method between *SecurityUpdateStart* and *SecurityUpdateCommit*.

Note If a user existing in the DB is NOT presented in *SecurityUpdate*, then the user is considered as being deleted from the system and marked as “**deleted = true**”.

The user would be recovered and marked as “**deleted = false**” as soon as the user is present in *SecurityUpdate*.

The password is updated only at creation or recovering of a user. If an administrator needs to change the password for a script imported user, he would need to first exclude the user from the *SecurityUpdate* call so the user is deleted, and then re-add him with a new password into the next iteration of *SecurityUpdate*.

Definition	Description	
Syntax	<p><i>SecurityUpdateAddUserGroup</i> (UserName As String, ExternalGroupID As Long, UserRole As String, UserDomain String)</p> <p><i>SecurityUpdateAddUserGroupPwd</i> (UserName As String, UserPassword as String, ExternalGroupID As Long, UserRole As String, UserDomain String)</p>	
Parameters	<i>UserName:</i>	The Username to create or update within the database. These are the user credentials to enter to log into the system. If Domain is populated, the user must enter MyDomain\UserName for logging into the verification application.
	<i>UserPassword</i>	<p>This password is applied only when creating or recovering a user. For those auto-imported users that were previously imported into Perceptive Intelligent Capture, the password remains unchanged.</p> <p>Use case rules:</p> <p>Auto-imported users with empty password are required to change their password upon first login.</p> <p>Auto-imported users with NON-empty password are NOT required to change their password upon first login.</p> <p>Auto-imported users who already changed their password upon first login will not be required to change their password anymore.</p>
	<i>ExternalGroupID:</i>	The external group ID security number. A batch and a user can be assigned a group ID that enables the user to verify only batches that fall under the same group ID assigned to that user.

Definition	Description	
	<p><i>UserRole:</i></p>	<p>The user role assigned to the Verifier user. The role can be one or a logical combination of the following text strings:</p> <p>SET – Can access settings.</p> <p>VER - Verifier user.</p> <p>SLV - Verifier supervisor (learnset nomination)</p> <p>SLM - Learnset Manager (global learnset manager)</p> <p>ADM - Administrator.</p> <p>FLT - Filtering</p>
	<p><i>UserDomain:</i></p>	<p>The user domain is left blank if no Windows Authentication is used, or when using Windows Authentication populated with the Domain name the Windows user belongs to.</p>
	<p>The following combinations of roles are possible:</p> <p>Project. SecurityUpdateAddUserGroup "User2", 999, "VER FLT", "BDomain" --> This creates a user with Verifier and Filter roles, but with no SET role</p> <p>Project. SecurityUpdateAddUserGroup "User2", 999, "VER SET FLT", "BDomain" --> This creates a user with Verifier, Settings and Filter roles</p> <p>Project. SecurityUpdateAddUserGroup "User2", 999, "VER", "BDomain" --> This creates a user with Verifier role only, with no SET and FLT role</p> <p>There is no need to combine SET/FLT roles with ADM, SLV, or SLM as these already contain FLT and SET roles by default.</p>	
<p>Example</p>	<p>The example below updates the database user security on a regular basis. The script can be updated to lookup users/roles and update the Perceptive Intelligent Capture user table.</p> <pre> Private Sub ScriptModule_UpdateSystemSecurity(ByVal InstanceName As String) Project.SecurityUpdateStart Project. SecurityUpdateAddUserGroup "User1", 777, "VER SET", "BDomain" Project. SecurityUpdateAddUserGroup "User1", 999, "VER SET", "BDomain" Project. SecurityUpdateAddUserGroup "User10", 777, "ADM", "BDomain" Project. SecurityUpdateAddUserGroupPwd ("User2", "pass", 777, "VER FLT", "") Project.SecurityUpdateCommit End Sub </pre>	

For additional information, see `SecurityUpdateStart`, `SecurityUpdateCommit`, `UpdateSystemSecurity`, and `PostImportBatch`

SecurityUpdateCommit

This method completes the security update process. This script call is required to complete updating the Perceptive Intelligent Capture users, roles, and groups.

When updating the security policy of Perceptive Intelligent Capture through a custom script, only the database tables are updated. The project security is not modified after a script update.

Definition	Description
Syntax	<code>Project.SecurityUpdateCommit</code>
Parameters	There are no parameters for this method.
Example	<p>The example below updates the database user security on a regular basis. The script can be updated to lookup users/roles and update the Perceptive Intelligent Capture user table.</p> <pre>Private Sub ScriptModule_UpdateSystemSecurity(ByVal InstanceName As String) Project.SecurityUpdateStart Project.SecurityUpdateAddUserGroup "User1", 777, "VER", "BDomain" Project.SecurityUpdateAddUserGroup "User2", 999, "SLV", "BDomain" Project.SecurityUpdateAddUserGroup "User3", 111, "VER", "BDomain" Project.SecurityUpdateAddUserGroup "User4", 888, "SLM", "BDomain" Project.SecurityUpdateAddUserGroup "User5", 222, "SET", "BDomain" Project.SecurityUpdateAddUserGroup "User6", 777, "VER FLT", "BDomain" Project.SecurityUpdateAddUserGroup "User10", 777, "ADM", "BDomain" Project.SecurityUpdateCommit End Sub</pre>

For additional information, see `SecurityUpdateStart`, `SecurityUpdateAddUserGroup`, `UpdateSystemSecurity`, and `PostImportBatch`.

SecurityUpdateStart

This method instantiates the security update process. This script call is required in order to begin updating the Perceptive Intelligent Capture users, roles, and groups.

When updating the security policy of Perceptive Intelligent Capture through a custom script, only the database tables are updated. The project security is not modified after a script update.

Definition	Description
Syntax	<code>Project.SecurityUpdateStart</code>
Parameters	There are no parameters for this method.

Definition	Description
Example	<p>The example below updates the database user security on a regular basis. The script can be updated to lookup users/roles and update the Perceptive Intelligent Capture user table.</p> <pre> Private Sub ScriptModule_UpdateSystemSecurity(ByVal InstanceName As String) Project.SecurityUpdateStart Project.SecurityUpdateAddUserGroup "User1", 777, "VER", "BDomain" Project.SecurityUpdateAddUserGroup "User2", 999, "SLV", "BDomain" Project.SecurityUpdateAddUserGroup "User3", 111, "VER", "BDomain" Project.SecurityUpdateAddUserGroup "User4", 888, "SLM", "BDomain" Project.SecurityUpdateAddUserGroup "User5", 222, "SET", "BDomain" Project.SecurityUpdateAddUserGroup "User6", 777, "VER FLT", "BDomain" Project.SecurityUpdateAddUserGroup "User10", 777, "ADM", "BDomain" Project.SecurityUpdateCommit End Sub </pre>

For more information, see `SecurityUpdateCommit`, `UpdateSystemSecurity`, and `PostImportBatch`.

SecurityUpdateUserParameter

This method establishes default group settings in Web Verifier for script imported users that do not have the SET role so that they are able to load projects and jobs.

With this method implemented, the corresponding group is found and assigned to the user as the `PrimaryUserGroup`.

If the group (or the user) cannot be found, a corresponding error message is shown.

This method works with auto-imported users as well as with normal users.

This method needs to be called between `SecurityUpdateStart` and `SecurityUpdateCommit`.

The group settings need to be configured in Web Verifier settings page by an administrator.

Definition	Description	
Syntax	<code>SecurityUpdateUserParameter (BSTR UserName, BSTR UserDomain, BSTR ParameterName, VARIANT Param1, VARIANT Param2)</code>	
Parameters	<i>ParameterName</i>	<p><code>PrimaryGroupID</code></p> <p>This parameter can have two variants:</p> <p>Param1: "GroupName" with Param2 as String that represents the group name displayed in the Web Verifier administrator group settings.</p> <p>"ExternalGroupID" with Param2 as Integer that represents the ExternalGroupID that was added in <code>SecurityUpdateAddUserGroup</code> or <code>SecurityUpdateAddUserGroupPwd</code>.</p>

Definition	Description
<p>Example</p>	<p>Add user A to two groups (100 and 101) and set his primary group for settings to be group 100. Add user Domain\B to one group and set his primary group for settings to be Autoimport_100. This is the displayed name of the group 100 in the Web Verifier administrator group settings.</p> <pre> Private Sub ScriptModule_UpdateSystemSecurity(ByVal InstanceName As String) Project.SecurityUpdateStart Project.SecurityUpdateAddUserGroupPwd("A", "pass", 100, "VER FLT", "") Project.SecurityUpdateAddUserGroupPwd("A", "pass", 101, "VER FLT", "") Project.SecurityUpdateUserParameter("A", "", "PrimaryGroupID", "ExternalGroupID", 100) Project.SecurityUpdateAddUserGroup("B", 100, "VER", "Domain") Project.SecurityUpdateUserParameter("B", "Domain", "PrimaryGroupID", "GroupName", "AutoImport_100") Project.SecurityUpdateCommit End Sub </pre>

Terminate

The Terminate event is called before a batch is closed after processing.

Description	Definition	
Syntax	ScriptModule_Terminate (ModuleName as String)	
Parameters	<i>ModuleName</i>	Name of the current module, values: "Designer", "Verifier" or "Server"
Example	<pre> Private Sub ScriptModule_Terminate(ByVal ModuleName As String) DB.Close Set DB = nothing End Sub This script can be added to one of the real projects, triggering the Terminate event in Runtime Server. This script will erase all state 0 batches that contain zero folders. Do not use this script piece together with the corresponding ProcessBatch Event Script within one project, because this Terminate Event script will make it impossible to load the ProcessBatch script: Private Sub ScriptModule_Terminate(ByVal ModuleName As String) On Error GoTo LABEL_ERROR Project.LogScriptMessageEx CDRTypeInfo, CDRSeveritySystemMonitoring, "Processing ScriptModule_Terminate event" Dim i As Long Dim pBatchRoot As New SCBCdrBATCHLib.SCBCdrBatchRoot pBatchRoot.ActivateSupport = True pBatchRoot.SetConnectionProperties("Version 5.4 SP1 Job", "Zero Folder Batch Terminator", False) pBatchRoot.Connect("Version 5.4 SP1 Job", "", "LOGIN_AS_CURRENT", "", "Zero Folder Batch Terminator") pBatchRoot.SetFilter(0) For i = 0 To pBatchRoot.BatchCount - 1 Step 1 If pBatchRoot.FolderCount(i) = 0 Then Project.LogScriptMessageEx CDRTypeWarning, CDRSeveritySystemMonitoring, "Zero Folder Batch Terminator detected batch with zero folders: [" & pBatchRoot.BatchID(i) & "]" pBatchRoot.DeleteBatch(pBatchRoot.BatchID(i), False, 0, 0) End If </pre>	

Description	Definition
	<pre> Next i Exit Sub LABEL_ERROR: Project.LogScriptMessageEx CDRTypeWarning, CDRSeveritySystemMonitoring, "Zero Folder Batch Terminator failed to search for zero folder batches. Error description: " & Err.Description End Sub </pre>

UpdateSystemSecurity

This event is triggered when the Runtime Server is configured to run with security updates.

Only one Runtime Server instance should be configured to update system security. The frequency of the security update is determined via the Runtime Server instance properties.

Description	Definition	
Syntax	ScriptModule_UpdateSystemSecurity(ByVal InstanceName As String)	
Parameters	<i>InstanceName</i>	The Runtime Server instance name that is calling the UpdateSystemSecurity event.
Example	<p>The following example updates the database user security on a regular basis. The script can be updated to lookup users/roles and update the Perceptive Intelligent Capture user table.</p> <pre> Private Sub ScriptModule_UpdateSystemSecurity(ByVal InstanceName As String) Project.SecurityUpdateStart Project.The following example "User1", 777, "VER", "BDomain" Project.The following example "User2", 999, "SLV", "BDomain" Project.The following example "User3", 111, "VER", "BDomain" Project.The following example "User4", 888, "SLM", "BDomain" Project.The following example "User5", 222, "SET", "BDomain" Project.The following example "User6", 777, "VER FLT", "BDomain" Project.The following example "User10", 777, "ADM", "BDomain " Project.SecurityUpdateCommit End Sub </pre>	

For additional information, see [ScriptModule Events](#), and [SecurityUpdateStart](#). You can also reference the examples for [SecurityUpdateCommit](#) and [PostImportBatch](#).

VerifierClassify

This event occurs only in Verifier when a document is manually classified.

Description	Definition	
Syntax	ScriptModule_VerifierClassify (pWorkdoc As ISCBCdrWorkdoc, Reason As CdrVerifierClassifyReason, ClassName As String)	
Parameters	<i>pWorkdoc</i>	Reference to the currently processed document.
	<i>Reason</i>	The reason why the script routine decided to reject or accept the document.
	<i>ClassName</i>	The name of the document class to which it is classified manually.

VerifierFormLoad

This event is triggered before the Verifier form is loaded. It enables the script to switch verification forms between different types of classes or to default the Verifier application to display a certain page instead of the first one (see the DisplayPage feature for additional information). It can also be used to modify the form before it gets displayed to the user. This event is triggered optionally in Designer application.

Description	Definition	
Syntax	ScriptModule_VerifierFormLoad (pWorkdoc As ISCBCdrWorkdoc, FormName As String, FormClassName As String)	
Parameters	<i>pWorkdoc</i>	Reference to the currently processed document.
	<i>FormName</i>	A string value that contains the current form name that Verifier application is going to load. The name can be modified in the custom script to initiate loading of a different form when required.
	<i>FormClassName:</i>	A string variable that contains the current class name of the verification form is to be loaded from. This name can be changed from within the Perceptive Intelligent Capture custom script to point to a different document class, in case the desired verification form is located in this different class.

Description	Definition
Example	<pre> Private Sub ScriptModule_VerifierFormLoad(pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc, FormClassName As String, FormName As String) Select Case UCase(FormClassName) Case "BASH" FormClassName = "Invoices" FormName = "Form_Invoices_2" Case "CONTACT" FormClassName = "Invoices" FormName = "Form_Invoices_1" End Select End Sub </pre>

Document Events

Cedar DocClass Event Interface

Document events are specific for each Cedar DocClass instance. Each DocClass has its own script module and implementation of script events.

FocusChanged

This event triggers each time before the focus inside the verification form is changes. You can modify the focus change by modifying the `pNewFieldIndex` parameter. You can write a different field index into that parameter, which causes the Verifier to change to a specific field instead of the originally selected field. The system triggers this event in Designer, if you set the `Reason` parameter to `CdrBeforeFormLoaded`, and if you enable the option in the Settings window, on the Compatibility tab.

Description	Definition	
Syntax	<code>Document_FocusChanged (pWorkdoc As ISCBCdrWorkdoc, Reason As CdrFocusChangeReason, OldFieldIndex As Long, pNewFieldIndex As Long)</code>	
Parameters	<i>pWorkdoc:</i>	Reference to the currently displayed workdoc.
	<i>Reason:</i>	Reason of the current focus change, which can be Tab key, Enter key, mouse click, or initial loading.
	<i>OldFieldIndex:</i>	Index of the current select field. In case of initial loading this -1.
	<i>pNewFieldIndex:</i>	Index of the field that should be selected now. This parameter can be modified during the script event to keep the focus in the previous field or set it to another field.

Description	Definition
Example	<pre> Private Sub Document_FocusChanged(pWorkdoc As SCBCdrWorkdoc, Reason As CdrFocusChangeReason, OldFieldIndex As Long, pNewFieldIndex As Long) 'Below you can find the sample of script code that helps to skip table 'data validation in Verifier (for a table with 2 columns): Dim theEmptyTable As SCBCdrPROJLib.SCBCdrTable Dim theEmptyTableField As SCBCdrPROJLib.SCBCdrField 'Initializes table and field references Set theEmptyTable = _ pWorkdoc.Fields("EmptyTable").Table(pWorkdoc.Fields("EmptyTable").ActiveTableIn dex) Set theEmptyTableField = pWorkdoc.Fields("EmptyTable") 'Makes table object valid theEmptyTable.CellValid(0,0) = True theEmptyTable.CellValid(1,0) = True theEmptyTable.RowValid(0) = True theEmptyTable.TableValid = True 'Makes table field valid '(table object is a part of more generic field object) theEmptyTableField.Valid = True theEmptyTableField.Changed = False 'Releases references Set theEmptyTable = Nothing Set theEmptyTableField = Nothing End Sub </pre>

OnAction

This event is triggered if any of the configured actions was caused by the user. Actions have to be configured in the Verifier design mode. Actions can either be caused if a user pressed a button or any of the configured keyboard short cuts.

Definition	Description	
Syntax	Document_OnAction (pWorkdoc As ISCBCdrWorkdoc, ActionName As String)	
Parameters	<i>pWorkdoc:</i>	Reference to the currently displayed workdoc.
	<i>ActionName:</i>	Name of the action that was assigned to the pressed button or short cut key.

Definition	Description
Example	<pre> Sub Document_OnAction(pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc, ByVal ActionName As String) If ActionName = "ShowBestSuppliers" Then Call fnShowBestSuppliers(pWorkdoc,pWorkdoc.Fields(FIELDNAME),"", "", "") End If End Sub </pre>

PostExtract

The PostExtract event is called after all defined analysis or evaluation methods are executed by the Cedar DocClass. During this event, it is possible to examine and change the results of one or more fields of the document.

You can also use this event in combination with generic Designer settings to establish multiple classifications. In Designer, establish a default classification result. Then set "pWorkdoc.DocClassName" to a different class in this event. This technique enables you to keep the generic extraction pointed toward the default class while moving the validation script to a different class.

Description	Definition	
Syntax	Document_PostExtract (pWorkdoc As ISCBCdrWorkdoc)	
Parameters	<i>pWorkdoc</i>	Current Workdoc object
Example	<pre> Private Sub Document_PostExtract(pWorkdoc As SCBCdrWorkdoc) Dim Number as string Dim Name as string 'get fields name and number Number = pWorkdoc.Fields("Number") Name = pWorkdoc.Fields("Name") End Sub </pre>	

PreExtract

The PreExtract event is called before any defined analysis or evaluation method is executed by the Cedar DocClass.

Description	Definition	
Syntax	Document_PreExtract (pWorkdoc As ISBCdrWorkdoc)	
Parameters	<i>pWorkdoc</i>	Current Workdoc object
Example	<pre> Private Sub Document_PreExtract(pWorkdoc As SCBCdrWorkdoc) Dim MyResult as string MyResult = DoSomeMagic(pWorkdoc) if (len(MyResult) > 0) then 'assign result to a single field pWorkdoc.Fields("Number") = MyResult; 'skip defined analysis and evaluation methods pWorkdoc.Fields("Number").FieldState = CDRFieldStateEvaluated end if end Sub </pre>	

PreVerifierTrain

ThePreVerifierTrain event has been added to control SLW training in Verifier, Learnset Manager, and Designer.

This event is called at the point when an application starts learning for a document in the supervised learning workflow (SLW).

Definition	Description	
Syntax	Document_PreVerifierTrain(pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc, pMode As Long)	
Parameters	<i>pMode</i>	It is reserved for further use and should not be used in the present software version.
Example	<p>The following script example demonstrates how the new script event can be used in order to apply a substitution of the primary Associative Search Engine field with another result referring to a different pool.</p> <pre> Private Sub Document_PreVerifierTrain(pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc, pMode As Long) If pWorkdoc.DocClassName = "NotGoodForPrimaryASEField" Then Project.AllClasses.ItemByName("Invoices").ClassificationField = "SecondaryAseField" End If End Sub </pre>	

Validate

Use the Validate event to perform validation on document level. At this point, the validation of all single fields is executed. If one of the fields is still invalid, pValid is FALSE. During the Document_Validate event, it is possible to implement validation rules combining several fields. This may cause some fields to be invalid again. Do not make the document invalid if all fields are valid because the Verifier needs an invalid field for focus control. If you want to keep the document invalid, always set at least one field to an invalid state.

It is also possible to make invalid fields valid during document validation. Therefore, you must set the Valid property of the appropriate fields to TRUE.

Description	Definition	
Syntax	Document_Validate (pWorkdoc As ISCBCdrWorkdoc, pValid As Boolean)	
Parameters	<i>pWorkdoc:</i>	Current Workdoc object
	<i>pValid:</i>	Parameter containing the current valid state of the Workdoc.
Example	<pre> Private Sub Document_Validate(pWorkdoc As SCBCdrWorkdoc, pValid As Boolean) Dim Number as string Dim Name as string 'get fields name and number and make a database lookup Number = pWorkdoc.Fields("Number") Name = pWorkdoc.Fields("Name") if LookupDBEntry(Name, Number) = FALSE then 'the Name/Number pair is NOT in the database 'set the document state to invalid pValid = FALSE 'make both fields invalid and provide an error description pWorkdoc.Fields("Number").Valid = FALSE pWorkdoc.Fields("Number").ErrorDescription = "Not in database" pWorkdoc.Fields("Name").Valid = FALSE pWorkdoc.Fields("Name").ErrorDescription = "Not in database" end if End Sub </pre>	

VerifierTrain

After a document processed in self-learning Verifier has been checked whether it is supposed to be automatically trained for the local project, the Verifier has to fire an event that adds a document to the local learnset.

Description	Definition	
Syntax	Document_VerifierTrain (pWorkdoc As ISCBCdrWorkdoc, ProposedClassName As String, WillTrain As Boolean, VerifierReason As CdrLocalTrainingReason, ScriptReason As String)	
Parameters	<i>pWorkdoc:</i>	Contains the reference to the currently processed document

Description	Definition	
	<i>ProposedClassName As String</i>	The proposed class name
	<i>WillTrain:</i>	Boolean value for the current learning state. True, when the document is going to be learned and False when it will not be learned.
	<i>VerifierReason:</i>	Contains the reason why the document was taken for training or why it was rejected. The reason parameter should be one of the predefined enumerated values for CdrLocalTrainingReason.
	<i>ScriptReason:</i>	Contains the reason why the script routine decided to reject or accept the document.

<Field_n> (Cedar FieldDef Event Interface)

Field events are specific for each Cedar field of each DocClass. Field events appear within the script sheet of their DocClass. That means all events for the field “Number” of the document class Invoice must be implemented within the script sheet of the DocClass Invoice.

Within the script the name of the fields will appear as specifier for the field. That means the Validate event for the field “Number” will appear as method "Number_Validate." During this documentation, <Field_n> is used as a placeholder for the name of the field. The Validate event is named here as <Field_n>_Validate.

CellChecked

This event occurs when a check box cell of the table is checked or unchecked by the user.

Description	Definition	
Syntax	<Field _n >_CellChecked (pTable As ISBCdrTable, pWorkdoc As ISBCdrWorkdoc, Row As Long, Column As Long, Checked As Boolean)	
Parameters	<i>pTable:</i>	Current Table object.
	<i>pWorkdoc:</i>	Current Workdoc object.
	<i>Row:</i>	This parameter contains the index of the current row on which the user clicked.
	<i>Column:</i>	This parameter contains the index of the current column on which the user clicked.
	<i>Checked:</i>	Boolean value that is TRUE when the cell is checked, otherwise its value is FALSE.

Description	Definition
Example	<pre>Private Sub Table_CellChecked(pTable As SCBCdrPROJLib.SCBCdrTable, pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc, ByVal Row As Long, ByVal Column As Long, ByVal Checked As Boolean) If Checked = True Then 'The cell (Row, Column) has been checked End If End Sub</pre>

CellFocusChanged

This event occurs each time the focus inside the verification table is going to be changed or can be changed potentially.

Description	Definition	
Syntax	<code><Fieldn>_CellFocusChanged (pTable As ISCBCdrTable, pWorkdoc As ISCBCdrWorkdoc, Reason As CdrTableFocusChangeReason, OldRow As Long, OldColumn As Long, pNewRow As Long, pNewColumn As Long)</code>	
Parameters	<i>pTable:</i>	Current Table object.
	<i>pWorkdoc:</i>	Current Workdoc object.
	<i>Reason:</i>	Parameter that contains the kind of focus change that has occurred
	<i>OldRow:</i>	This parameter contains the index of the derivation row.
	<i>OldColumn:</i>	This parameter contains the index of the derivation column.
	<i>pNewRow:</i>	This parameter contains the index of the destination row. This value can be changed (set back to OldRow value), to forbid an action, such as double-clicking on the special column.
	<i>pNewColumn:</i>	This parameter contains the index of the destination column. This value can be changed (set back to OldColumn value), to forbid an action, such as double-clicking on the special column.

Description	Definition
<p>Example</p>	<pre> Private Sub Table_CellFocusChanged(pTable As SCBCdrPROJLib.SCBCdrTable, pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc, ByVal Reason As SCBCdrPROJLib.CdrTableFocusChangeReason, ByVal OldRow As Long, ByVal OldColumn As Long, pNewRow As Long, pNewColumn As Long) Select Case Reason Case CdrTfcrCellBitmapClicked 'Occurs when a user clicks on cell's picture, e.g., on check-box image of a check-box cell. Case CdrTfcrCellDoubleClicked 'Occurs if a user double clicks on a table cell. Could be useful if it ' is designed to 'Implement a kind of database look-up, etc by double clicking on a cell. Case CdrTfcrCellLocationClicked 'Occurs when a user clicks on a word that is linked to one of the cells in image viewer. 'This will cause setting of keyboard focus to the corresponding table cell. Case CdrTfcrColumnMapped 'Occurs when a user maps a column. Case CdrTfcrColumnsSwapped 'Occurs when a user swaps two columns. Case CdrTfcrColumnUnmapped 'Occurs when a user unmaps a column. Case CdrTfcrEnterPressed 'Occurs when "Enter" key is pressed, i.e. cell (table) validation is activated. Case CdrTfcrFocusRefreshed 'Occurs when the application refreshes a table. Case CdrTfcrFormLoaded 'Occurs right after a new document to verify is loaded. Case CdrTfcrMouseClicked 'Occurs when a cell is selected by mouse click. Case CdrTfcrRowsMerged 'Occurs when rows were merged to one row. Case CdrTfcrRowsRemoved 'Occurs when a user removes a row. Case CdrTfcrTableCandidateChanged 'Occurs when a user changes current table candidate. Case CdrTfcrTabPressed 'Occurs when the focus is changed to another cell by arrow keys or Tab keys. Case CdrTfcrUnknownReason 'Focus is changed due to unknown reason. End Select 'Example of changing cell focus from the script: 'when document is opened, set focus to the first cell If Reason = CdrTfcrFormLoaded Then pNewRow = 0 pNewColumn = 0 End If 'Example of changing cell focus from the script: do not allow selection of first cell by mouse If OldRow = 0 And OldColumn = 0 And Reason = CdrTfcrMouseClicked Then pNewRow = 1 pNewColumn = 1 End If End Sub </pre>

Format

The Format event can be used to reformat the content of a Field, for example to unify a date or amount format or removing prefixes and suffixes. This event can be used to prepare the field data for validation. Be reminded that the content of pField.Text is normally used for learning within the Scripting Guide engines. If the user wants to change the output format for the field content use the script event FormatForExport.

Description	Definition	
Syntax	<Fieldn>_Format (pField As ISCBCdrField)	
Parameters	<i>pField:</i>	Field object
Example	<pre>Private Sub Amount_Format(pField As SCBCdrField) Dim NewAmount as string if MyReformatAmount(pField, NewAmount) = TRUE then 'reformatting of the text field is successful to prepare a field for validation pField.Text = NewAmount end if End Sub</pre>	

FormatForExport

The FormatForExport event can be used to reformat the content of a field, for example to unify a date or amount format or removing prefixes and suffixes and to keep this additional information within pField.FormattedText rather than to change pField.Text. This text is normally used for learning within the Scripting Guide engines. This formatted text can also be used for Export.

Description	Definition	
Syntax	<Fieldn>_FormatForExport (pField As ISCBCdrField)	
Parameters	<i>pField:</i>	Current field.
Example	<pre>Private Sub Amount_FormatForExport(pField As SCBCdrField) Dim NewAmount as string if MyReformatAmount(pField, NewAmount) = TRUE then 'reformatting is successful to generate a unified output format for the fields' content. 'Use the pField.FormattedText to save the reformatted information. 'You should then use pField.FormattedText also for the Export, instead of pField.Text pField.FormattedText = NewAmount end if End Sub</pre>	

PostAnalysis

The PostAnalysis event is called after the analysis step is performed. It is possible to examine the list of all candidates and to add further candidates to the Field.

Description	Definition	
Syntax	<Fieldn>_PostAnalysis (pField As ISCBCdrField, pWorkdoc As ISCBCdrWorkdoc)	
Parameters	<i>pField:</i>	Object containing the Field
	<i>pWorkdoc:</i>	Current Workdoc object
Example	<pre>Private Sub MyField_PostAnalysis(pField As SCBCdrField, pWorkdoc As SCBCdrWorkdoc) Dim cindex as long, count as long, id as long 'add a new candidate to the field if pWorkdoc.Wordcount > 42 then 'use the 42th word as new candidate count = 1'wordcount of new candidate id = 0 'rule-id for later backtracing pField.AddCandidate 42, count, id, cindex 'cindex is the new index of the candidate end if End Sub</pre>	

PostEvaluate

The PostEvaluate event is called after the evaluation step is performed. It is possible to examine the list of all candidates and to change their weights.

Description	Definition	
Syntax	<Fieldn>_PostEvaluate (pField As ISCBCdrField, pWorkdoc As ISCBCdrWorkdoc)	
Parameters	<i>pField:</i>	Object containing the Field
	<i>pWorkdoc:</i>	Current Workdoc object
Example	<pre>Private Sub MyField_PostEvaluate(pField As SCBCdrField, pWorkdoc As SCBCdrWorkdoc) 'set the weight of the first candidate to 1 if pField.CandidateCount > 0 then pField.Candidate(0).Weight = 1 end if End Sub</pre>	

PreExtract

The PreExtract event is called before any defined analysis or evaluation method for this Field is executed by the Cedar DocClass.

Description	Definition	
Syntax	<Fieldn>_PreExtract (pField As ISBCdrField, pWorkdoc As ISBCdrWorkdoc)	
Parameters	<i>pField:</i>	Object containing the Field
	<i>pWorkdoc:</i>	Current Workdoc object
Example	<pre>Private Sub Today_PreExtract(pField As SCBCdrField, pWorkdoc As SCBCdrWorkdoc) 'the field Today should contain the processing date of the document Dim today as date today = Date pField = Format(date, "yyyymmdd") End Sub</pre>	

SmartIndex

The SmartIndex event is called for the Field where the smart indexing was defined. This field usually provides the key for the select statement.

Description	Definition	
Syntax	<Fieldn>_SmartIndex (pField As ISBCdrField, pWorkdoc As ISBCdrWorkdoc)	
Parameters	<i>pField:</i>	Object containing the current Field
	<i>pWorkdoc:</i>	Current Workdoc object
Example	<pre>Private Sub CustomerNo_SmartIndex(pField As SCBCdrPROJLib.SCBCdrField, pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc) 'avoid validation for the Name field if filled by smart indexing pWorkdoc.Fields("Name").Valid = TRUE End Sub</pre>	

TableHeaderClicked

This event occurs when a user clicks on one of the table header buttons. There are three different table header buttons: Row Header button, the Column Header button, and Table Header button.

Description	Definition	
Syntax	<code><Fieldn>_TableHeaderClicked (pTable As ISCBCdrTable, pWorkdoc As ISCBCdrWorkdoc, ClickType As CdrTableHeaderClickType, Row As Long, Column As Long, pSkipDefaultHandler As Boolean)</code>	
Parameters	<i>pTable:</i>	Current Table object
	<i>pWorkdoc:</i>	Current Workdoc object
	<i>ClickType:</i>	The click type of the mouse depends on the place where the click occurred either for the Column Header, Row Header, or Table Header and the type of click that occurred, such as a single click, double-click, or right-click.
	<i>Row:</i>	This parameter contains the index of the current row on which the user clicked.
	<i>Column:</i>	This parameter contains the index of the current column on which the user clicked.
	<i>pSkipDefaultHandler:</i>	The default value is False. When the user wants to skip the default handling it has to be set to True.

Description	Definition
Example	<pre> Private Sub Table_TableHeaderClicked(pTable As SCBCdrPROJLib.SCBCdrTable, pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc, ByVal ClickType As SCBCdrPROJLib.CdrTableHeaderClickType, ByVal Row As Long, ByVal Column As Long, pSkipDefaultHandler As Boolean) Select Case ClickType Case CdrColumnHeaderClicked 'Table column header button has been clicked - 'define your message handler here Case CdrColumnHeaderDoubleClicked 'Table column header button has been double clicked - 'define your message handler here Case CdrColumnHeaderRightButtonClicked 'Right mouse button has been clicked on table column header - 'define your message handler here Case CdrRowHeaderClicked 'Table row header button has been clicked - 'define your message handler here Case CdrRowHeaderDoubleClicked 'Table row header button has been double clicked - 'define your message handler here Case CdrRowHeaderRightButtonClicked 'Right mouse button has been clicked on table row header - 'define your message handler here Case CdrTableHeaderClicked 'Table header button has been clicked - 'define your message handler here Case CdrTableHeaderDoubleClicked 'Table header button has been double clicked - 'define your message handler here Case CdrTableHeaderRightButtonClicked 'Right mouse button has been clicked on table header - 'define your message handler here End Select 'Skip default handler of the table header clicked event '(handler implemented in the Verifier component) pSkipDefaultHandler = True End Sub </pre>

Validate

The field Validate event can be used to perform project specific validation rules. Use the pValid parameter to return the validation decision. So if the parameter remains unchanged or if the event is not implemented, the document state gets valid if all fields are valid.

Description	Definition	
Syntax	<Fieldn>_Validate (pField As ISCBCdrField, pWorkdoc As ISCBCdrWorkdoc, pValid As Boolean)	
Parameters	<i>pField:</i>	Object containing the current Field
	<i>pWorkdoc:</i>	Current Workdoc object

Description	Definition	
	<i>pValid:</i>	Parameter containing the current valid state of the Field
Example	<pre> Private Sub Number_Validate(pField As SCBCdrField, pWorkdoc As SCBCdrWorkdoc, pValid As Boolean) 'check result of standard validation if pValid = FALSE then 'standard validation returns invalid, stop here exit sub end if 'perform additional check for number format if IsValidNumber(pField) = FALSE then pValid = FALSE pField.ErrorDescription = "Field is not a valid number" end if End Sub </pre>	

ValidateCell

This event method is called for each cell of the Table. Here you can implement validation checks specific for a single cell.

Definition	Description	
Syntax	<Fieldn>_ValidateCell (pTable As ISCBCdrTable, pWorkdoc As ISCBCdrWorkdoc, Row As Long, Column As Long, pValid As Boolean)	
Parameters	<i>pTable:</i>	Current Table object
	<i>pWorkdoc:</i>	Current Workdoc object
	<i>Row:</i>	Given Row of the Table
	<i>Column:</i>	Given column of the Table
	<i>pValid:</i>	Parameter containing the current valid state of the Table cell.

Definition	Description
Example	<pre> Private Sub MyTableField_ValidateCell(pTable As SCBCdrPROJLib.SCBCdrTable, pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc, ByVal Row As Long, ByVal Column As Long, pValid As Boolean) Select Case Column Case 0: 'check date in column 0 if CheckDate(pTable.CellText(Column, Row)) = FALSE then pValid = FALSE pTable. CellValidationErrorDescription(Column, Row) = "Invalid date" end if Case 2: 'check order number in column 2 if CheckOrderNumber(pTable.CellText(Column, Row)) = FALSE then pValid = FALSE pTable. CellValidationErrorDescription(Column, Row) = "Invalid order number" end if End Select End Sub </pre>

ValidateRow

Implement validation rules, which combine two or more cells of a row.

Description	Definition	
Syntax	<code><Fieldn>_ValidateRow (pTable As ISCBCdrTable, pWorkdoc As ISCBCdrWorkdoc, Row As Long, pValid As Boolean)</code>	
Parameters	<i>pTable:</i>	Table Object for which row is to be validated
	<i>pWorkdoc:</i>	Current Workdoc object
	<i>Row:</i>	Given row of the Table to be validated
	<i>pValid:</i>	Parameter that contains the current valid state of the row

Description	Definition
Example	<pre> Private Sub MyTableField_ValidateRow(pTable As SCBCdrPROJLib.SCBCdrTable, pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc, ByVal Row As Long, pValid As Boolean) 'check if quantity * single price = total price Dim quantity as long Dim s_price as double, t_price as double 'all cells must already have a valid format quantity = CLng(pTable.CellText("Quantity", Row)) s_price = CLng(pTable.CellText("Single Price", Row)) t_price = CLng(pTable.CellText("Total Price", Row)) if quantity*s_price = t_price then pValid = TRUE else pValid = FALSE pTable.RowValidationErrorDescription(Row) = "Invalid quantity or amounts" end if End Sub </pre>

ValidateTable

Implements a validation rule for the entire Table.

Description	Definition	
Syntax	<Fieldn>_ValidateTable (pTable As ISCBCdrTable, pWorkdoc As ISCBCdrWorkdoc, pValid As Boolean)	
Parameters	<i>pTable:</i>	Table object
	<i>pWorkdoc:</i>	Current Workdoc object
	<i>pValid:</i>	Parameter containing the current valid state of the Table

Description	Definition
Example	<pre> Private Sub MyTableField_ValidateTable (pTable As SCBCdrPROJLib.SCBCdrTable, pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc, pValid As Boolean) 'calculate the sum of all amounts and compare with the net amount fields Dim tablesum as double, netamount as double Dim cellamount as double Dim row as long For row = 0 to pTable.RowCount-1 cellamount = CLng(pTable.CellText("Total Price", Row)) tablesum = tablesum + cellamount Next row 'now compare sum with the content of the net amount field netamount = CDBl(pWorkdoc.Fields("NetAmount").Text if netamount = tablesum then pValid = TRUE else pValid = FALSE pTable.TableValidationErrorDescription ="Sum of table amounts and field net amount are different" end if End Sub </pre>

Workdoc Object Reference (SCBCdrWorkdocLib)

SCBCdrWorkdoc

Description

The Cedar Workdoc object stores all data of one document. The amount of data grows during the processing steps of OCR, classification and extraction.

Type Definitions

CdrEdgeSide

This the definition that determines the type of alignment or edges.

Available Types	Description
<i>CDREdgeLeft</i>	Chooses left alignment (left edges) in analysis.
<i>CDREdgeRight</i>	Chooses right alignment (right edges) in analysis.

CdrExportType

This data type determines which data from the current document will export. It is used in the method ExportToXML.

Available Types	Description
<i>CdrExportTypeOCRData</i>	Exports OCR data of words and characters of a Workdoc.

CDRHighlightMode

This definition is the highlighting mode for the workdoc that displays for the user, such as highlight candidates, highlight fields only, and so on.

Available Types	Description
<i>CDRHighlightAttractors</i>	Attractor highlighting
<i>CDRHighlightBlocks</i>	Block highlighting
<i>CDRHighlightCandidates</i>	Candidates highlighting
<i>CDRHighlightCandidatesAdvanced</i>	Highlights only candidates but according to their advanced highlighting type, also fires all mouse events for all words
<i>CDRHighlightCheckedWords</i>	Verified words highlighting
<i>CDRHighlightCheckedWordsAndCandidates</i>	Verified words and candidate highlighting
<i>CDRHighlightCheckedWordsAndField</i>	Verified words and field highlighting
<i>CDRHighlightCheckedWordsAndFields</i>	Verified words and fields highlighting
<i>CDRHighlightFields</i>	Fields highlighting
<i>CDRHighlightNothing</i>	No highlighting
<i>CDRHighlightParagraphs</i>	Paragraph highlighting
<i>CDRHighlightRectangles</i>	Variable rectangle highlighting
<i>CDRHighlightTables</i>	Table highlighting
<i>CDRHighlightTablesAdvanced</i>	Highlights checked words and selected table cell, also shows tool-tips for all words and fires all mouse events for all words
<i>CDRHighlightTextLines</i>	Text lines highlighting
<i>CDRHighlightTextLinesAdvanced</i>	Highlights text lines according their block number, show tool-tips with line confidences, also fires all mouse events

Available Types	Description
<i>CDRHighlightTrainedFields</i>	Trained fields highlighting
<i>CDRHighlightVerticalEdgesLeft</i>	Left aligned edges highlighting
<i>CDRHighlightVerticalEdgesRight</i>	Right aligned edges highlighting
<i>CDRHighlightWords</i>	Word highlighting

CDRClassifyResult

This data type is responsible for specifying the result of classification for a specific document class and specific classification engine. This is the same as the cell inside the classification matrix within Designer.

Available Types	Description
<i>CDRClassifyMaybe</i>	Document may belong to DocClass but weights are not available
<i>CDRClassifyNo</i>	Document does not belong to this DocClass
<i>CDRClassifyNotApplied</i>	Classification engine is not applied to this DocClass
<i>CDRClassifyWeighted</i>	Classification weight property has valid content
<i>CDRClassifyYes</i>	For sure document belongs to this DocClass

CDRDocState

This definition determines the current state of the document within the workflow.

Available Types	Description
<i>CDRDocStateAnalyzed</i>	Document is analyzed
<i>CDRDocStateBlocks</i>	Blocks are analyzed in document
<i>CDRDocStateClassified</i>	Document is classified
<i>CDRDocStateDeleted</i>	Document is deleted
<i>CDRDocStateEvaluated</i>	Document is evaluated
<i>CDRDocStateExported</i>	Document is exported
<i>CDRDocStateHaveDocs</i>	Images or CIDocs are assigned to documents
<i>CDRDocStateLanguage</i>	Language detection executed
<i>CDRDocStateReset</i>	Initial state of document

Available Types	Description
<i>CDRDocStateValid</i>	Validity state of document
<i>CDRDocStateWorktext</i>	Worktext is assigned to document

CDRPageAssignment

This data type is responsible for specifying how the Document Pages are assigned to the Workdoc.

Available Types	Description
<i>CDRPageAssignAllPages</i>	Assign all DocPages of Image or CIDoc to Workdoc
<i>CDRPageAssignNewPage</i>	First Page of Image or CIDoc appended as last DocPage to Workdoc
<i>CDRPageAssignNoPage</i>	No DocPages assigned to Workdoc

CDRPDFExportStyle

This data type is responsible for specifying the export type of PDF image out of Perceptive Intelligent Capture.

Available Types	Description
<i>CDRPDF_ImgOnly</i>	Export only Image to PDF
<i>CDRPDF_ImgOnTxt</i>	Export Image on top of text to PDF
<i>CDRPDF_NoExport</i>	No Export for single DocPage
<i>CDRPDF_NoThumbnails</i>	No thumbnail generated for DocPage
<i>CDRPDF_TxtOnly</i>	Export only text to PDF

CDRDocFileType

This data type is the enumeration that contains the type of input file.

Available Types	Description
<i>CDRDocFileTypeCroCIDoc</i>	Cairo CIDocument
<i>CDRDocFileTypeCrolImage</i>	Cairo image object
<i>CDRDocFileTypeRawText</i>	Created from plain text without document
<i>CDRDocFileTypeUnknown</i>	Unknown file type, maybe attachment

Methods and Properties

AddDocFile

This method adds a file into the Workdoc. File types include CIDoc, image, and raw text.

Description	Definition	
Syntax	<code>AddDocFile (Path As String, FileType As CDRDocFileType, Assignment As CDRPageAssignment)</code>	
Parameters	<i>FilePath:</i>	Path to the file to be added
	<i>FileType:</i>	File type of the specified file, such as a CIDoc or Image.
	<i>Assignment:</i>	It specifies how DocPages are assigned to the Workdoc
Example	This code shows how to add a CI-PDF file to the workdoc. <code>pWorkdoc.AddDocFile("C:\coversheet.pdf",CDRDocFileTypeCroCIDoc,CDRPageAssignNewPage)</code>	

AddField

This method adds a field to the Workdoc.

Description	Definition	
Syntax	<code>AddField (Name As String)</code>	
Parameter	<i>Name :</i>	Contains the name for the new field
Example	This example adds the field "AdditionalField" to the workdoc <code>pWorkdoc.AddField("AdditionalField")</code>	

AddHighlightRectangle

This method adds a highlight rectangle on the page described by the following parameters. Set HighlightMode to CDRHighlightRectanglesto highlight all rectangles.

Description	Definition	
Syntax	<code>AddHighlightRectangle (Left As Long, Top As Long, Width As Long, Height As Long, PageNr As Long, Color As OLE_COLOR)</code>	
Parameters	<i>Left:</i>	Left of highlight rectangle
	<i>Top:</i>	Top of highlight rectangle
	<i>Width:</i>	Width of highlight rectangle

Description	Definition	
	<i>Height:</i>	Height of highlight rectangle
	<i>PageNr:</i>	Document page number of highlight rectangle
	<i>Color:</i>	Color of highlight rectangle
Example	<code>pWorkdoc.AddHighlightRectangle(10,10,100,100,1,vbCyan)</code>	

AnalyzeAlignedBlocks

This method splits the document into blocks that contain only left or right aligned lines. Using this method on a document with centered lines only usually results in one block per line.

Description	Definition	
Syntax	<code>AnalyzeAlignedBlocks (edgeSide As CDREdgeSide, leftAlignTolerance As Long, XDist As Double, YDist As Double, Join As Boolean, minDistance As Double)</code>	
Parameters	<i>edgeSide:</i>	Determines whether left or right aligned blocks are to be found
	<i>leftAlignTolerance:</i>	The distance (in mm) that aligned lines might differ. Useful if document was scanned slightly tilted.
	<i>XDist:</i>	A value, depending on the font size of a word, that specifies how far off an existing block of words may be to belonging to that block. If its horizontal distance from the block is greater that XDist, then a new block is created.
	<i>YDist:</i>	This value specifies (in mm) the maximum vertical distance for a word from a block. If its distance is greater that YDist, a new block is generated
	<i>Join:</i>	Specifies whether overlapping blocks are to be joined. Set to TRUE if you want to join them.
	<i>minDistance:</i>	This parameter is a factor to be multiplied with leftAlignTolerance. It specifies the minimal horizontal distance of two edges. Set this value to 0 to ignore its effect.

AnalyzeBlocks

This method determines all the TextBlocks of text present in a Workdoc that are a minimum XDist apart from each other on X-axis and a minimum of YDist apart from each other on Y-axis.

Description	Definition	
Syntax	<code>AnalyzeBlocks (XDist As Double, YDist As Double)</code>	
Parameters	<i>XDist:</i>	Minimum X distance between two TextBlocks

Description	Definition	
	<i>YDist:</i>	Minimum Y distance between two TextBlocks
Example	<p>pWorkdoc.AnalyzeBlocks(4,4)</p> <p>Increasing the distance will result in bigger text blocks. Minimizing the distance will increase the number of smaller text blocks.</p> <p>Note Search strings comprising multiple words will match candidates only if the multiple words candidates reside within the same text block. Use the AnalyzeBlocks method to adjust the text blocks to your requirements.</p>	

AnalyzeEdges

This method Analyzes a document set of words that are, within a certain tolerance, aligned either right or left. Use Highlight mode (CDRHighlightVerticalEdgesLeft or CDRHighlightVerticalEdgesRight) to make the results visible.

Description	Definition	
Syntax	<code>AnalyzeEdges (edgeSide As CDREdgeSide, AlignTolerance As Double, YDist As Double, MinNoOfWords As Long, minDistance As Double, [pageNr As Long = TRUE])</code>	
Parameters	<i>edgeSide:</i>	Set this parameter to either CDREdgeLeft or CDREdgeRight to specify if you want edges that contain left or right aligned words.
	<i>AlignTolerance:</i>	This value (in mm) specifies how far the left (right) values of words bounding rectangle may differ in order for it to still be considered aligned.
	<i>YDist:</i>	Specifies (in mm) how far two words may be apart vertically and still belong to the same edge.
	<i>MinNoOfWords:</i>	Specifies how many words have to belong to a valid edge. Edges that contain less than MinNoOfWords after analyzing the document are deleted.
	<i>minDistance:</i>	This parameter is a factor to be multiplied with AlignTolerance. It specifies the minimal horizontal distance of two edges. Set this value 0 to ignore its effect.
	<i>pageNr:</i>	[optional,defaultvalue(-1)] Specifies the page to be analyzed for edges. Set to -1 (default) if analysis is needed for all pages.

AnalyzeEdges2

This method is similar to AnalyzeEdges method, but it applies the processing for visible text lines only (in case 'vbCheckedOnly' parameter is set to TRUE), otherwise it works exactly like AnalyzeEdges.

Description	Definition	
Syntax	AnalyzeEdges2 (edgeSide As CDREdgeSide, AlignTolerance As Double, YDist As Double, MinNoOfWords As Long, minDistance As Double, pageNr As Long, vbCheckedOnly As Boolean)	
Parameters	<i>edgeSide:</i>	Set this parameter to either CDREdgeLeft or CDREdgeRight to specify if you want edges that contain left or right aligned words.
	<i>AlignTolerance:</i>	This value (in mm) specifies how far the left (right) values of words bounding rectangle may differ in order for it to still be considered aligned.
	<i>YDist:</i>	Specifies (in mm) how far two words may be apart vertically and still belong to the same edge.
	<i>MinNoOfWords</i>	Specifies how many words have to belong to a valid edge. Edges that contain less than MinNoOfWords after analyzing the document are deleted.
	<i>minDistance:</i>	This parameter is a factor to be multiplied with AlignTolerance. It specifies the minimal horizontal distance of two edges. Set this value 0 to ignore its effect.
	<i>pageNr:</i>	Specifies the page to be analyzed for edges. Set to -1 (default) if analysis is needed for all pages.
	<i>vbCheckedOnly:</i>	If set to TRUE, the method applies processing for visible text lines only, otherwise this function works exactly like AnalyzeEdges.

AnalyzeParagraphs

This method is used to determine all the paragraphs present in Workdoc.

Description	Definition
Syntax	AnalyzeParagraphs ()

AppendWorkdoc

This method is used to append a given Workdoc to the existing Workdoc.

Description	Definition	
Syntax	AppendWorkdoc (pWorkdoc As ISCBCdrWorkdoc)	
Parameter	<i>pWorkdoc:</i>	Workdoc that is to be appended

AssignDocToPage

Use this method to assign a Page of an Image or CIDoc to a specific DocPage of the Workdoc. This method requires that there are already documents inserted to the Workdoc using the AddDocFile function and that the SetPageCount function is called prior to using this method.

Description	Definition	
Syntax	AssignDocToPage (DocIndex As Long, DocPage As Long, WorkdocPage As Long)	
Parameters	<i>DocIndex:</i>	Zero-based CIDoc or Image Index
	<i>DocPage:</i>	Zero-based DocPage inside the Image or CIDoc
	<i>WorkdocPage:</i>	Zero-based DocPage inside the Workdoc

AttractorColor

This property sets or returns the color that is used for attractor highlighting.

Description	Definition
Syntax	AttractorColor As OLE_COLOR (read/write)
Example	This example sets the AttractorColor to green. pWorkdoc.AttractorColor = vbGreen

BatchID

A property of the Workdoc that allows you to retrieve the Batch ID in which the current Workdoc resides.

Description	Definition
Attribute	strBatchID As String (Read Only)
Example	The following script example shows how to retrieve the Batch ID. Dim strBatchID As String strBatchID = pWorkdoc.NamedProperty("BatchID")

BlockColor

This property sets or returns the color that is used for block highlighting.

Description	Definition
Syntax	BlockColor As OLE_COLOR (read/write)
Example	This example sets the color for block highlighting to cyan pWorkdoc.BlockColor = vbCyan

BlockCount

This property returns the number of TextBlocks of the Workdoc. Use this property before accessing the TextBlock property where an index is required. The range of valid indices for TextBlocks is from 0 to BlockCount -1.

Description	Definition
Syntax	BlockCount As Long (read only)
Example	<pre> This example writes the text of each block to the string array 'strBlockText'. Dim strBlockText() As String Dim intBlockCount As Integer Dim i as Long intBlockCount = pWorkdoc.BlockCount -1 ReDim strBlockText(intBlockCount) For i=0 To intBlockCount strBlockText(i) = pWorkdoc.TextBlock(i).Text Next i </pre>

CandidateColor

This property sets or returns the color that is used for candidate highlighting.

Note The candidate color is not customizable in Verifier Thick Client.

Description	Definition
Syntax	CandidateColor As OLE_COLOR (read/write)
Example	<pre> This example sets the candidate color to magenta pWorkdoc.CandidateColor = vbMagenta </pre>

Clear

Use this method to clear all the memories and to remove all the documents from Workdoc. This leaves the Workdoc in an initial state.

Description	Definition
Syntax	Clear ()

ClearHighlightRectangles

This parameter removes all highlighted rectangles.

Description	Definition
Syntax	ClearHighlightRectangles ()

ClsEngineConfidence

This property sets or returns a confidence level for a classification engine specified by its index in the collection of classification engines.

Description	Definition
Syntax	ClsEngineConfidence (lMethodIndex As Long) As Long (read/write)
Parameter	<i>lMethodIndex:</i> Zero-based engine index in collection of classification engines.
Example	<pre> This example shows a message box with the confidence value for each classification engine. Dim dblIndividualResult As Double Dim lEngineIndex As Long For lEngineIndex = 0 To Project.ClassifySettings.Count dblIndividualResult = (pWorkdoc.ClsEngineConfidence(lEngineIndex)) MsgBox "The classification confidence is " & dblIndividualResult Next lEngineIndex </pre>

ClsEngineDistance

This property sets or returns the distance value for a classification engine specified by its index in a collection of classification engines.

Description	Definition
Syntax	ClsEngineDistance (lMethodIndex As Long) As Long (read/write)
Parameter	<i>lMethodIndex:</i> Zero-based engine index in collection of classification engines.
Example	<pre> This example shows a message box for each class, showing the classification engine distance. Dim dblIndividualResult As Double Dim lEngineIndex As Long For lEngineIndex = 0 To Project.ClassifySettings.Count dblIndividualResult = (pWorkdoc.ClsEngineDistance(lEngineIndex)) MsgBox "The engine distance is " & dblIndividualResult Next lEngineIndex </pre>

ClsEngineResult

Use this property to access a classification result matrix. This matrix is used during the classification step to store the results of each used classification method for each document class (DocClass) of the project. The matrix has one column for each classification method and one column for the combined result of all methods. A row contains the results for a single DocClass, therefore there is one row for each DocClass in the classification matrix. The matrix is created during the classification step, but not saved to disk. After reloading the Workdoc, the matrix is no longer available.

The method returns the classification matrix as CDRClassifyResult. See the type definition for further details.

Description	Definition	
Syntax	ClsEngineResult (MethodIndex As Long, DocClassIndex As Long) As CDRClassifyResult (read/write)	
Parameters	<i>MethodIndex:</i>	MethodIndex = 0 can be used to access the voted result of all classification methods. A MethodIndex of 1 - n can be used to access the results of the single classification methods. The sorting of the classification methods within the array is determined by the Collection of classification settings of the Perceptive Intelligent Capture Project. You can access this Collection from the script as Project.ClassifySettings, which has a type of SCBCroCollection. Use the Count property to get the number of used classification engines or use the ItemIndex / ItemName property to find the index of classification method or the name for an index.
	<i>DocClassIndex:</i>	The DocClassIndex is determined by the Collection of all DocClasses. You can access this Collection from the script as Project.AllClasses, which has a type of SCBCroCollection. Use the Count property to get the number of DocClasses or use the ItemIndex / ItemName property to find the index of DocClass or the name for an index.
Example	<p>The following example sets the classification result of the Brainware Classify Engine to YES for a document in docclass "VOID". If Brainware Classify is the only engine or all other classes would be CDRClassifyNo, the document would get classified as VOID.</p> <pre>pWorkdoc.ClsEngineResult(Project.ClassifySettings.ItemIndex("Brainware Classify Engine"), Project.AllClasses.ItemIndex("VOID"))= CDRClassifyYes</pre>	

ClsEngineWeight

This property provides access to the classification weights within the Classification Result Matrix.

Description	Definition
Syntax	ClsEngineWeight (MethodIndex As Long, DocClassIndex As Long) As Double (read/write)

Description	Definition	
Parameters	<i>MethodIndex:</i>	MethodIndex = 0 can be used to access the voted result of all classification methods. A MethodIndex of 1 - n can be used to access the results of the single classification methods. The sorting of the classification methods within the array is determined by the Collection of classification settings of the Perceptive Intelligent Capture Project. You can access this Collection from the script as Project.ClassifySettings, which has a type of SCBCroCollection. Use the Count property to get the number of used classification engines or use the ItemIndex / ItemName property to find the index of classification method or the name for an index.
	<i>DocClassIndex:</i>	The DocClassIndex is determined by the collection of all document classes. You can access this Collection from script as Project.AllClasses that is a type of SCBCroCollection. Use the Count property to get the number of DocClasses or use the ItemIndex / ItemName property to find the index of DocClass or the name for an index.

CreationDate

A property of the Workdoc that allows the developer to retrieve the Creation Date of the current Workdoc. When a document is placed in a new exception batch, the attribute updates to a new date/time stamp.

Description	Definition
Attribute	Read Only
Example	The script sample below shows how to retrieve the Creation Date. <pre>Dim dtCreationDate As Date dtCreationDate = pWorkdoc.NamedProperty("CreationDate")</pre>

CreationDateAsFileTimeUTC

A new property of the Workdoc has been introduced to allow the developer to retrieve the Creation Date in UTC of the current Workdoc.

When a document is placed in a new exception batch, the attribute updates to a new date/time stamp.

Description	Definition
Attribute	Read Only
Example	The script sample below shows how to retrieve the Creation Date. <pre>Dim dtCreationDateUTC As Long dtCreationDateUTC = pWorkdoc.NamedProperty("CreationDateAsFileTimeUtc")</pre>

CreateFromWorktext

This method creates Workdoc from the OCR'd text of an Image.

Description	Definition	
Syntax	<code>CreateFromWorktext (pWorktext As ISCBCroWorktext)</code>	
Parameter	<i>pWorktext:</i>	Object pointer of Worktext.

CurrentBatchState

This is a property that returns the temporary document batch state, a numeric value between 0 and 999. This value is set by the methods LoadWorkdoc and UpdateDocument of the Cedar Batch component.

Description	Definition
Syntax	<code>pWorkdoc.CurrentBatchState (Read only)</code>

DeleteFile

This method deletes all wdcs and corresponding TIFs of the Workdoc.

Description	Definition	
Syntax	<code>DeleteFile (DeleteDocFiles As Boolean)</code>	
Parameter	<i>DeleteDocFiles:</i>	Flag to inform whether to delete files or not

DisplayPage

This property sets or returns the displayed DocPage specified by the zero-based index of the Workdoc in the Viewer (Index 0 represents page 1).

Description	Definition
Syntax	<code>DisplayPage As Long (read/write)</code>
Example	<p>If a customer requires Verifier to display a specific page of each document instead of the first one when opening the document, use the DisplayPage property in the script.</p> <p>In the example below, the script displays Page 3 if the document has 4 pages or more.</p> <pre>Private Sub ScriptModule_VerifierFormLoad(pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc, FormClassName As String, FormName As String) If pWorkdoc.PageCount >=3 Then pWorkdoc.DisplayPage = 2 End Sub</pre>

DocClassName

This property sets or returns the name of the DocClass to which the document was classified.

Description	Definition
Syntax	DocClassName As String (read/write)
Example	<pre> Private Sub ScriptModule_PreClassify(pWorkdoc As SCBCdrWorkdoc) if (DoSomeMagic(pWorkdoc) = TRUE) then 'assign "Invoice" as result of the classification pWorkdoc.DocClassName = 'Invoice' else 'do nothing and continue with normal classification end if End Sub </pre>

DocFileCount

This property returns the number of documents from which the Workdoc is built.

Description	Definition
Syntax	DocFileCount As Long (read only)

DocFileDatabaseID – Unique ID

The read only property pWorkdoc.DocFileDatabaseID returns the database ID of document files attached to a Perceptive Intelligent Capture Workdoc. It corresponds to the [File].[Id] value in the database. The document file index has to be passed as a parameter when using DocFileDatabaseID property.

Use this property in custom script as a unique identifier of document files that were processed by Perceptive Intelligent Capture.

Description	Definition		
Attribute	Read only		
Syntax	DocFileDatabaseID (ByVal Index As long) As Long		
Parameter	<table border="1"> <tr> <td><i>Index</i></td> <td>The index parameter has a valid range from 0 to DocFileCount -1</td> </tr> </table>	<i>Index</i>	The index parameter has a valid range from 0 to DocFileCount -1
<i>Index</i>	The index parameter has a valid range from 0 to DocFileCount -1		
Example	<pre> Dim lUniqueID As Long lUniqueID = pWorkdoc.DocFileDatabaseID(pWorkdoc.DocFileCount - 1) </pre> <p>The script example above demonstrates how to retrieve the unique ID of the last document file attached to a Workdoc.</p>		

DocFileName

This property returns the full pathname of a document (image or text file) from which the Workdoc is built.

Description	Definition	
Syntax	DocFileName (index As Long) As String (read only)	
Parameter	<i>Index:</i>	The index parameter has a valid range from 0 to DocFileCount-1.
Example	<p>If a Workdoc was created from a single document, such as a multi-TIFF file, the name of the document file can be retrieved accessing the 0 index.</p> <pre> Path = pWorkdoc.DocFileName(0) The script function below returns the TIF file creation date. Public Function fnGetFileDate(pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc) As String Dim FSO As New Scripting.FileSystemObject Dim oFile As Scripting.File Dim strFileName As String Dim dtCreated As Date strFileName = Replace(pWorkdoc.DocFileName(0), ".wdc", ".tif") If FSO.FileExists(strFileName) Then Set oFile = FSO.GetFile(strFileName) dtCreated = oFile.DateCreated fnGetFileDate = Month(dtCreated) & "/" & Day(dtCreated) & "/" & Year(dtCreated) End If Set FSO = Nothing Set oFile = Nothing End Function </pre>	

DocFileType

This property returns the file type of the document by the specified index.

Description	Definition	
Syntax	DocFileType (index As Long) As CDRDocFileType (read only)	
Parameter	<i>Index:</i>	The index parameter has a valid range from 0 to DocFileCount-1.

DocState

This property sets or returns the current state of the document.

Description	Definition	
Syntax	DocState As CDRDocState (read/write)	

EdgeCount

This property returns the number of vertical edges found in a document.

Description	Definition	
Syntax	EdgeCount (edgeSide As CDREdgeSide) As Long (read only)	
Parameter	<i>edgeSide:</i>	Flag to distinguish between left and right edges.

ErrorDescription

This property sets or returns an error description.

Description	Definition
Syntax	ErrorDescription As String (read/write)
Example	<pre> Private Sub Document_Validate(pWorkdoc As SCBCdrWorkdoc, pValid As Boolean) Dim Number as string Dim Name as string 'get fields name and number and make a database lookup Number = pWorkdoc.Fields("Number") Name = pWorkdoc.Fields("Name") if LookupDBEntry(Name, Number) = FALSE then 'the Name/Number pair is NOT in the database 'set the document state to invalid pValid = FALSE 'make both fields invalid and provide an error description pWorkdoc.Fields("Number").Valid = FALSE pWorkdoc.Fields("Number").ErrorDescription = "Not in database" pWorkdoc.Fields("Name").Valid = FALSE pWorkdoc.Fields("Name").ErrorDescription = "Not in database" end if End Sub </pre>

ExportToXML

This method exports OCR data results of the current workdoc into an XML file with a predefined format. The export captures word data and the associated characters data.

Description	Definition	
Syntax	<code>ExportToXml (ByVal DocumentLanguage As String, ByVal DocumentType As String, ByVal Customer As String, ByVal eExportType As CDREExportType, ByVal XMLFilePath As String)</code>	
Parameter	<i>DocumentLanguage, DocumentType, Customer</i>	You can customize these parameters to use it later for filtering purposes. The values have no correlation with the Workdoc. Note If the strings are empty, the value defaults to Default, Null is not allowed.
	<i>eExportType</i>	This parameter defines the type of information from the current document for the export to the XML files.
	<i>XMLFilePath</i>	Defines the path for the XML file. Leave it as an empty string to save the XML files in the current application start folder. It is recommended to define the file path in script. You can specify an existing folder terminated by a back slash, or define the target name for the XML file explicitly. Example Folder: C:\TMP\ File Name: C:\TMP\myfilename.xml Note You can only specify existing folders; the method will not create them. If the method fails to create the file, an error message will notify you about the failed export.
XML file format	Document section	Presents the general document information: <ul style="list-style-type: none"> - page count - line count - word count
	Words section	Presents information about the single words in the document: <ul style="list-style-type: none"> - word text - word length - page number - position of the word in pixels
	Characters section	Presents information about the single characters of the word: <ul style="list-style-type: none"> - character text - character position in pixels Note For CI documents, this method exports only the word positions, no individual character positions.

Description	Definition
	<p>Example of the XML format with parameter eExportType set to CDRExportTypeOCRData</p> <pre> <?xml version="1.0" encoding="UTF-8" standalone="yes"?> <Document> <Name>00000473</Name> <DocumentType>Default</DocumentType> <DocumentLanguage>Default</DocumentLanguage> <Customer>Default</Customer> <PageCount>1</PageCount> <Pages> <Page id="0" DocFileType="Image"/> </Pages> <LineCount>39</LineCount> <WordCount>334</WordCount> <Words> <Word id="0"> <Text>UNICOM</Text> <Length>6</Length> <StartPos>0</StartPos> <Page>0</Page> <Line>1</Line> <Top>131</Top> <Left>303</Left> <Height>102</Height> <Width>564</Width> <Characters> <Char id="0"> <Code>U</Code> <Top>133</Top> <Left>303</Left> <Height>100</Height> <Width>80</Width> </Char> <Char id="1">...</Char> ... </Characters> </Word> <Word id="1">...</Word> ... </Words> </Document> </pre>
<p>Example</p>	<p>This line exports the OCR data for the current pWorkdoc into an XML file located in the C:\Temp folder.</p> <pre>pWorkdoc.ExportToXml("", "", "", CDRExportTypeOCRData, "C:\Temp\")</pre>

FieldColor

This property sets or returns the color that is used to highlight valid and invalid Fields.

Description	Definition	
Syntax	<code>FieldColor (FieldValid As Boolean) As OLE_COLOR (read/write)</code>	
Parameter	<i>FieldValid:</i>	If set to TRUE it specifies the color for valid Fields or it specifies the color for invalid Fields if FALSE.

Fields

This property provides access to all Fields of a document.

Description	Definition
Syntax	<code>Fields As ISCBCdrFields (read only)</code>
Example	To read the text content of a simple Field use the following command: <pre>Dim FieldContent as string FieldContent = pWorkdoc.Fields.Item("MyField").Text</pre>

FileName

This property contains the database ID of the Workdoc and returns the database Workdoc ID and Name.

Note To retrieve the file name of the image from which the Workdoc was created, use the DocFileName property.

Description	Definition
Syntax	<code>Filename As String (read only)</code>

Folder

This property accesses the Folder to which the Workdoc belongs.

Description	Definition
Syntax	<code>Folder As ISCBCdrFolder (read only)</code>

FolderIndex

This property provides the index of the Folder to which a Workdoc belongs.

Description	Definition
Syntax	<code>FolderIndex As Long (read only)</code>

ForceClassificationReview

In the application, the PostClassify event has been extended so that it can force a manual classification review even if the classification succeeded.

Description	Definition
Attribute	Read/Write
Example	<p>The script sample below shows how the manual classification process can be forced from custom script event "PostClassify".</p> <pre>Private Sub ScriptModule_PostClassify(pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc) If pWorkdoc.DocClassName = "VeryImportantClass" Then pWorkdoc.ForceClassificationReview = True End If End Sub</pre>

GetEdge

This method returns the coordinates for the left, top, and bottom of the corners for an edge, which is interpreted as a rectangle.

Description	Definition
Syntax	GetEdge (edgeSide As CDREdgeSide, edgeIndex As Long, pLeft As Long, pTop As Long, pBottom As Long, pPageNr As Long)
Parameters	<p><i>edgeSide</i>: Set this parameter to either CDREdgeLeft or CDREdgeRight to specify if you want edges that contain left or right aligned words.</p>
	<p><i>edgeIndex</i>: Index of the edge to be returned, valid indices are from 0 to the result of EdgeCount - 1.</p>
	<p><i>pLeft</i>: Contains left coordinate of the edge.</p>
	<p><i>pTop</i>: Contains top coordinate of the edge.</p>
	<p><i>pBottom</i>: Contains bottom coordinate of the edge.</p>
	<p><i>pPageNr</i>: Contains page number of the edge.</p>

GetFileSizeKB

This method retrieves the file size of an image or document through a custom script.

Description	Definition
Syntax	GetFileSizeKB(pWorkdoc As SCBCdrWorkdoc) As Integer

Description	Definition
Example	<pre> Private Function GetFileSizeKB(pWorkdoc As SCBCdrWorkdoc) As Integer Dim FSO As FileSystemObject Dim ImageFile As File On Error GoTo ErrHandler Set FSO = New FileSystemObject Set ImageFile = FSO.GetFile(pWorkdoc.DocFileName(0)) GetFileSizeKB = Round(ImageFile.Size/1024) Exit Function ErrHandler: GetFileSizeKB = -1 End Function </pre>

GetWorktextForPageArea

A function that returns a worktext object from a specific location on a document. The worktext object contains text and positional information relating to the area specified in GetWorktextForPageArea. You can view this as a temporary zone to read a piece of information through a script and review the returned result for that area.

The area to search starts from Left and Top coordinates and finishes at Width and Height coordinates, provided in pixels. These are the same coordinates that you would enter for a reading zone. For more information, see “Setting up Zone Analysis” in the *Perceptive Intelligent Capture Designer Help*.

The scripiter may test their page area coordinates using a zone.

Description	Definition	
Syntax	GetWorktextForPageArea(Page, Left, Top, Width, Height, IncludePartial)	
Parameters	<i>Page:</i>	page number of the image. 0 represents the first page of a multi page document.
	<i>Left:</i>	left coordinate of the page area.
	<i>Top:</i>	top most coordinate of the page area.
	<i>Width:</i>	width (length) of the area.
	<i>Height:</i>	height of the area
	<i>includePartial:</i>	Boolean flag. > False – restricts reading of worktext to specified area. > True - completes words that appear partially in the specified area with outside information.

Description	Definition
Example of includePartial	The word "Invoice" exists on the page, but our page area only captures "Inv". Setting <i>includePartial</i> to <i>False</i> will return only "Inv", setting <i>includePartial</i> to <i>True</i> will return the entire word "Invoice".
Example of the code to use	<p>The example below takes the OCR results of the top left page area and places the result into the first row table cell.</p> <pre> Dim ptrWorkText As SCBCroWorktext Set ptrWorkText = New SCBCroWorktext Set ptrWorkText = pWorkdoc.GetWorktextForPageArea(0, 100, 100, 300, 300,True) pWorkdoc.Fields.ItemByName("TableField").Table(0).CellWorktext(0,0) = ptrWorkText </pre>

HighlightCandidate

This property sets or returns the position of the highlighted Candidate.

Description	Definition
Syntax	HighlightCandidate As Long (read/write)

HighlightField

This property sets or returns the position of the highlighted Field.

Description	Definition
Syntax	HighlightField As Long (read/write)

HighlightMode

This property sets or returns the current mode of highlighting.

Description	Definition
Syntax	HighlightMode As CDRHighlightMode (read/write)

IgnoreAnalysisFailures

This is an optional capability to ignore any errors during Perceptive Intelligent Capture's extraction analysis phase. Otherwise, the extraction analysis stops in the middle of field extraction and does not apply processing for other fields and does not fire further events.

This capability is optional and by default switched off to ensure the backwards compatibility is not affected in any way.

If set to TRUE, any errors occurring during extraction analysis phase are ignored. Errors will not cause a sudden termination of the extraction process. Instead, traces are left in the component logs for the CdrProj library (at tracing level 1):

```
0|0|13:10:14.840|LErr:0|hRes:0x80005141|cdrproj\scbcdrdocclass.cpp|Wed Sep 12 13:07:13 2012|2416|F|Error preprocessing zone !
Zone rectangle out of image.|||
0|0|13:10:14.840|LErr:0|hRes:0|cdrproj\scbcdrdocclass.cpp|Wed Sep 12 13:07:13 2012|2416||Level2||SAVINGS|
```

By default, this option is switched off. It can be activated at any time, for example in the PreExtract event.

Description	Definition
Syntax	<code>pWorkdoc.NamedProperty("IgnoreAnalysisFailures")</code>
Example	<pre>' Cedar Document Class Script for Class "Level2" Private Sub SAVINGS_PreExtract(pField As SCBCdrPROJLib.ISCBCdrField, pWorkdoc As SCBCdrPROJLib.ISCBCdrWorkdoc) pWorkdoc.NamedProperty("IgnoreAnalysisFailures") = True End Sub</pre>

Image

This property returns an Image object for the specified DocPage of the Workdoc.

Description	Definition		
Syntax	<code>Image (index As Long) As ISCBCroImage (read only)</code>		
Parameter	<table border="1"> <tr> <td><i>Index:</i></td> <td>Index of the DocPage that is valid from 0 to PageCount - 1.</td> </tr> </table>	<i>Index:</i>	Index of the DocPage that is valid from 0 to PageCount - 1.
<i>Index:</i>	Index of the DocPage that is valid from 0 to PageCount - 1.		

IsPlainText

This property sets or returns if worktext is plain text or not.

Description	Definition
Syntax	<code>IsPlainText As Boolean (read/write)</code>

Language

This property sets or returns the language of the document, as it was specified by the language detection or the default language of the Project.

Description	Definition
Syntax	<code>Language As String (read/write)</code>

LineColor

This property sets or returns the Color that is used for line highlighting.

Description	Definition
Syntax	LineColor As OLE_COLOR (read/write)

Load

This method loads a file from given root path and this root path is not the absolute path of the file.

Description	Definition	
Syntax	Load (Filename As String, ImageRootPath As String)	
Parameters	Filename:	Name of the file.
	ImageRootPath:	Relative path of the file.

PageCount

This property returns the number of displayable DocPages of the Workdoc.

Description	Definition
Syntax	PageCount As Long (read only)
Example	intImageCount=pWorkdoc.PageCount 'Get the number of pages in Workdoc

Pages

This property returns a single DocPage of the Workdoc.

Description	Definition	
Syntax	Pages (PageIndex As Long) As ISBCdrDocPage (read only)	
Parameter	PageIndex:	Index of the DocPage to access, which is valid from 0 to PageCount-1.

Paragraph

This property provides access to the paragraph array of the Workdoc.

Description	Definition	
Syntax	Paragraph (index As Long) As ISBCdrTextBlock (read only)	
Parameter	Index:	Specifies the index of the paragraph. Valid indexes are from 0 to the result of ParagraphCount - 1.

ParagraphCount

This property returns the number of paragraphs in the

Description	Definition
Syntax	ParagraphCount As Long (read only)

PCAppType

Use this named property to optimize the check amount extraction rates.

Description	Definition	
Syntax	<code>pWorkdoc.NamedProperty ("PCAppType")</code>	
eParameters	“POD” – default setting Proof of deposit	The Check Analysis Engine is tuned to minimize the error rate. You can limit candidate lists to values with the highest confidence levels. The engine often fails to recognize values above 1 million USDollars. See the HVOL option below.
	“RMT” Remittance	The Check Analysis Engine is tuned to a maximum read rate without rejection. It expects to use all results and alternative answers. It is not necessary to accept only answers with high confidence values, because users can perform cross-validation using remittance coupons and databases.
	“HVOL”	This parameter enables the engine to recognize amounts larger than 1 million US dollars.
Example	<pre>Private Sub Document_PreExtract(pWorkdoc As SCBCdrPROJLib.ISCBCdrWorkdoc) : : pWorkdoc.NamedProperty("PCAppType") = "HVOL" 'High Value Amount Range : : End Sub</pre>	

PCCheckType

Use this named property to configure the check type recognition.

Description	Definition
Syntax	<code>pWorkdoc.NamedProperty ("PCCheckType")</code>

Description	Definition	
Parameters	<p>“ALL”</p> <ul style="list-style-type: none"> - Personal checks - Business checks - Cash tickets - Deposit slips - Money orders 	<p>This parameter sets the Check Analysis Engine to expect all supported types of documents: personal checks, business checks, cash tickets, deposit slips, and money orders.</p>
	<p>“P”</p> <ul style="list-style-type: none"> - Personal checks 	<p>This parameter sets the Check Analysis Engine to expect the input stream to consist of only personal checks. If your documents consist of 99.5% personal checks, this setting may improve processing speed while not significantly affecting accuracy.</p>
	<p>“PB” – default setting</p> <ul style="list-style-type: none"> - Personal checks - Business checks 	<p>This parameter sets the Check Analysis Engine to expect checks and cash tickets. Use this setting if your documents consist mainly of checks.</p>
	<p>“PBD”</p> <ul style="list-style-type: none"> - Personal checks - Business checks - Cash tickets - Deposit slips 	<p>This parameter sets the Check Analysis Engine to expect checks, cash tickets, and deposit slips.</p>
Example	<pre>Private Sub Document_PreExtract(pWorkdoc As SCBCdrPROJLib.ISCBCdrWorkdoc) : : pWorkdoc.NamedProperty("PCCheckType") = "P" 'Personal Checks only : : End Sub</pre>	

PCDateHint

Use this named property to configure the reference date for the check date recognition by the Check Analysis Engine. By default, the application uses the system date.

Description	Definition
Syntax	pWorkdoc.NamedProperty ("PCDateHint")

Description	Definition	
	<i>Format</i>	Use the following format for the reference date. YYYY.MM.DD Example 2015.06.18 To set the reference date to the system date, add an empty string.
Example	<pre> Private Sub Document_PreExtract(pWorkdoc As SCBCdrPROJLib.ISCBCdrWorkdoc) : : pWorkdoc.NamedProperty("PCDateHint") = "2013.04.12" 'April 12th 2012 : : End Sub </pre> <p>The following example sets the reference date to the system date:</p> <pre> Private Sub Document_PreExtract(pWorkdoc As SCBCdrPROJLib.ISCBCdrWorkdoc) : : pWorkdoc.NamedProperty("PCDateHint") = " " 'System date : : End Sub </pre>	

PCReReadAlways

Use this named property to enable re-analyzing the Check Analysis Engine fields. This is helpful if you need to perform Designer or scripting testing, such as for document rotation. It is also helpful if you change one of the Check Analyses Engine settings. By default, re-analysis is switched off.

Description	Definition	
Syntax	<code>pWorkdoc.NamedProperty ("PCReReadAlways")</code>	
Parameters	<i>True</i>	The Check Analysis Engine repeats the analysis, if <ul style="list-style-type: none"> • Execute in the Designer's Definition Mode for extraction • AnalyzeField method is used in script If AnalyzeField on a Check Analysis Engine field, all fields with Check Analysis Engine assigned will be re-analyzed. The "<Field>_PostAnalysis" event only triggers for the field for which the Analyzed field was triggered. <ul style="list-style-type: none"> • AnalyzeDocument is used in script.

Description	Definition	
	<i>False – default setting</i>	The Check Analysis Engine will not execute a secondary analysis on a document when PIC did not unload the document from the memory.
Example	<pre>Private Sub Document_PreExtract(pWorkdoc As SCBCdrPROJLib.ISCBCdrWorkdoc) : : pWorkdoc.NamedProperty("PCReReadAlways") = True'Switch on re-Analysis : : End Sub</pre>	

PDFExport

This method generates a PDF file from Workdoc based on CDRPDFExportStyle.

Description	Definition	
Syntax	PDFExport (FileName As String)	
Parameter	<i>FileName:</i>	Name of the exported PDF file.

PDFGetInfoType

This method returns the export type of a given Page in a PDF file.

Description	Definition	
Syntax	PDFGetInfoType (PageIdx As Long, pExportStyle As CDRPDFExportStyle)	
Parameters	<i>PageIdx:</i>	Page number of PDF.
	<i>pExportStyle:</i>	Type of Export.

PDFSetInfoType

This method sets the type of export of a PDF.

Description	Definition	
Syntax	PDFSetInfoType (PageIdx As Long, ExportStyle As CDRPDFExportStyle)	
Parameters	<i>PageIdx:</i>	Zero-based DocPage Number.
	<i>ExportStyle:</i>	Type of export

ReadZone

This is part of the OCR-on-demand concept.

Description	Definition	
Syntax	<code>ReadZone (PageIndex As Long, [left As Double = FALSE], [top As Double = FALSE], [right As Double = 1], [bottom As Double = 1])</code>	
Parameters	<i>PageIndex:</i>	Specifies the DocPage where the OCR or text conversion should be executed. Valid indices are 0 to PageCount - 1 for working on single pages or -1 for executing OCR on all DocPages.
	<i>Right:</i>	[in,optional,defaultvalue(1)] Specifies the right border of the OCR region in percent. Use 100 here to read until the right border.
	<i>Left:</i>	[in,optional,defaultvalue(0)] Specifies a left offset for the OCR region in percent. Use 0 here to read from the left border.
	<i>Top:</i>	[in,optional,defaultvalue(0)] Specifies the top offset for the OCR region in percent. Use 0 here to read from the top border.
	<i>Bottom:</i>	[in,optional,defaultvalue(1)] Specifies the bottom line of the OCR region in percent. Use 100 here to read until the bottom border.

Refresh

This method refreshes the Workdoc's DocPage that is currently shown in the Viewer.

Description	Definition
Syntax	<code>Refresh ()</code>

RenameDocFile

Use this method to change the name of the CIDoc or Image at a given DocIndex by the given new name.

Description	Definition	
Syntax	<code>RenameDocFile (DocIndex As Long, NewName As String)</code>	
Parameters	<i>DocIndex:</i>	Specifies the zero-based CIDoc or Image Index.
	<i>NewName:</i>	New name given to the document at DocIndex.

ReplaceFirstImage

This method replaces the first image in Workdoc.

Description	Definition	
Syntax	ReplaceFirstImage (Path As String)	
Parameter	<i>Path:</i>	Image path to replace the existing workdoc image.

Save

This method saves a Workdoc with given file name and its DocFiles relatively at given ImageRootPath.

Description	Definition	
Syntax	Save (Filename As String, ImageRootPath As String)	
Parameters	<i>Filename:</i>	Filename of Workdoc
	<i>ImageRootPath:</i>	Relative path where all corresponding DocFiles are saved, empty if files are saved in the same directory as the Workdoc.

SkipTableCellMassValidation

This method allows you to optionally activate special "skip table cell mass validation" mode for validation of table cells. By default, Perceptive Intelligent Capture uses "mass validation of invalid cells". This means that upon an *Enter* click within an invalid cell, all other invalid cells are automatically re-validated by the system. This behavior may lead to performance problems in Perceptive Intelligent Capture projects with a large number (1000+) of invalid cells that must each be corrected manually. It may be also unacceptable if validation routines are unavailable for some of the processed transactions and manual review by the Verifier user is required for all cells.

You can invoke this script at any time and is in effect for the next fired cell validation event. You can also turn mass validation back on at any time. One of the possible events in which this script sample can be integrated is "ScriptModule_VerifierFormLoad".

Description	Definition
Example	The validation mode can be switched individually for each processed document via the following script: <pre>pWorkdoc.NamedProperty("SkipTableCellMassValidation") = True</pre>

SetDocPageIndex

This method has been added to allow the script implementation of the page merging workflow step.

Description	Definition
Example	<p>The following short script example shows how this method can be used to append one document to another.</p> <pre> For j = 0 To thePreviousWorkdoc.PageCount -1 Step 1 theNextWorkdoc.InsertPage (thePreviousWorkdoc, j, True, theNextWorkdoc.PageCount) theNextWorkdoc.Pages (theNextWorkdoc.PageCount - 1).SetDocPageIndex(0, j + 1) End If </pre>

ShowTooltips

This property sets or returns if tooltips display when moving the mouse pointer over a displayed Workdoc.

Description	Definition
Syntax	ShowTooltips As Boolean (read/write)

SkipTrainingWithEngine

This property identifies whether the specified trainable engine has to skip this document in the training process.

Description	Definition		
Syntax	SkipTrainingWithEngine (bstrEngineName As String) As Boolean (read/write)		
Parameter	<table border="1"> <tr> <td><i>bstrEngineName:</i></td> <td>Name of classification engine.</td> </tr> </table>	<i>bstrEngineName:</i>	Name of classification engine.
<i>bstrEngineName:</i>	Name of classification engine.		

Table

Returns a Table for a given index of the Workdoc.

Description	Definition		
Syntax	Table (index As Long) As ISCBCdrTable (read only)		
Parameter	<table border="1"> <tr> <td><i>Index:</i></td> <td>Specifies the index of the Table. Valid indices are from 0 to TableCount-1.</td> </tr> </table>	<i>Index:</i>	Specifies the index of the Table. Valid indices are from 0 to TableCount-1.
<i>Index:</i>	Specifies the index of the Table. Valid indices are from 0 to TableCount-1.		

TableCount

This property returns the number of Table objects stored within the Workdoc.

Description	Definition
Syntax	TableCount As Long (read only)

TextBlock

This property returns the TextBlock by an index of the Workdoc.

Description	Definition		
Syntax	TextBlock (index As Long) As ISCBCdrTextBlock (read only)		
Parameter	<table border="1"> <tr> <td><i>Index:</i></td> <td>[in] Specifies the index of the TextBlock. Valid indices are from 0 to BlockCount-1.</td> </tr> </table>	<i>Index:</i>	[in] Specifies the index of the TextBlock. Valid indices are from 0 to BlockCount-1.
<i>Index:</i>	[in] Specifies the index of the TextBlock. Valid indices are from 0 to BlockCount-1.		

Textline

This property returns text line by an index of the Workdoc.

Description	Definition		
Syntax	Textline (index As Long) As ISCBCdrTextBlock (read only)		
Parameter	<table border="1"> <tr> <td><i>Index:</i></td> <td>Zero-based index.</td> </tr> </table>	<i>Index:</i>	Zero-based index.
<i>Index:</i>	Zero-based index.		

TextlineCount

This property retrieves the number of text lines present in a Workdoc.

Description	Definition
Syntax	TextlineCount As Long (read only)

TrainedWithEngine

This property indicates whether this document is trained with the specified engine.

Description	Definition		
Syntax	TrainedWithEngine (bstrEngineName As String) As Boolean (read only)		
Parameter	<table border="1"> <tr> <td><i>bstrEngineName:</i></td> <td>Name of engine.</td> </tr> </table>	<i>bstrEngineName:</i>	Name of engine.
<i>bstrEngineName:</i>	Name of engine.		

UnloadDocs

This method releases all the Images and CIDocs that belong to this Workdoc.

Description	Definition
Syntax	UnloadDocs ()

Word

This property provides access to the Word array of the Workdoc.

Description	Definition
Syntax	Word (index As Long) As ISBCdrWord (read only)
Parameter	<i>Index:</i> [in] Index of the requested Word. Valid indices are from 0 to WordCount-1.

WordColor

This property sets or returns the color that is used for Word highlighting.

Description	Definition
Syntax	WordColor As OLE_COLOR (read/write)

WordCount

This property returns the number of Words of the Workdoc.

Description	Definition
Syntax	WordCount As Long (read only)
Example	<pre>Private Sub MyField_PostAnalysis(pField As SCBCdrField, pWorkdoc As SCBCdrWorkdoc) Dim cindex as long, count as long, id as long 'add a new candidate to the field if pWorkdoc.Wordcount > 42 then 'use the 42th word as new candidate count = 1 'wordcount of new candidate id = 0 'rule-id for later backtracing pField.AddCandidate 42, count, id, cindex 'cindex is the new index of the candidate end if End Sub</pre>

WordSegmentationChars

This property sets or returns a string that contains the characters used for the segmentation of Words.

Description	Definition
Syntax	WordSegmentationChars As String (read/write)

Worktext

Provides access to the raw OCR results represented by the SCBCroWorktext object.

Description	Definition
Syntax	<code>Worktext As ISBCCroWorktext (read only)</code>

SCBCdrFields

Description

This is a collection of all Field objects contained in the current WorkDoc object.

Methods and Properties

Add

This method adds a new Field with the specified name to the Field Collection.

Description	Definition	
Syntax	<code>Add (NewItem As ISBCdrField, ItemName As String) As Long</code>	
Parameters	<i>NewItem:</i>	[in] Pointer to a SCBCdrField object that should be added to the Collection.
	<i>ItemName:</i>	[in] Name of the Field item inside the Collection. This name must be used to access the item inside the Collection.

Clear

This method removes all items from the Collection and releases their reference count.

Description	Definition
Syntax	<code>Clear ()</code>

Collection

This property returns the Collection that is internally used to store the Fields.

Description	Definition
Syntax	<code>Collection As ISBCCroCollection (read only)</code>

Count

This property returns the number of items within the Field Collection.

Description	Definition
Syntax	Count As Long (read only)

Item

These read-only properties return a specified item from the Collection. The Item property is the default property of the ISCBCdrFields Collection.

Description	Definition	
Syntax	Item (Index As Variant) As ISCBCdrField (read only)	
Parameter	<i>Index:</i>	The index can either be a long value specifying the index within the collection, valid range from 1 to Count, or a string specifying the item by name.

ItemByIndex

This property returns an item from the Collection specified by the index.

Description	Definition	
Syntax	ItemByIndex (Index As Long) As ISCBCdrField (read only)	
Parameter	<i>Index:</i>	Index of the item to retrieve from the Collection, valid range from 1 to Count
Example	<code>strClassName = theProject.AllClasses.ItemByIndex(intClass).Name</code>	

ItemByName

This property returns the Field from the Collection by the specified Field name.

Description	Definition	
Syntax	ItemByName (Name As String) As ISCBCdrField (read only)	
Parameter	<i>Name:</i>	[in] Name of the item to retrieve from the Collection.

Description	Definition
Example	<pre> Private Sub Document_FocusChanged(pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc, ByVal Reason As SCBCdrPROJLib.CdrFocusChangeReason, ByVal OldFieldIndex As Long, pNewFieldIndex As Long) If pWorkdoc.Fields.ItemByName("InteractiveTableExtractionAllowed").Text = "No" Then Project.AllClasses.ItemByName(pWorkdoc.DocClassName).Fields.ItemByName(" LineItems").AllowInteractiveExtraction = False Else Project.AllClasses.ItemByName(pWorkdoc.DocClassName).Fields.ItemByName(" LineItems").AllowInteractiveExtraction = True End If End Sub </pre>

ItemExists

This method returns TRUE if an item with the specified name exists inside the Collection, or FALSE is returned.

Description	Definition	
Syntax	ItemExists (Name As String) As Boolean	
Parameter	<i>Name:</i>	Name of item to search for.

ItemIndex

The index of an item specified by name is returned

Description	Definition	
Syntax	ItemIndex (Name As String) As Long (read only)	
Parameter	<i>Name:</i>	Name specifying an item in the Collection.

ItemName

The name of an item is returned specified by index.

Description	Definition	
Syntax	ItemName (Index As Long) As String (read only)	
Parameter	<i>Index:</i>	Index specifying an item in the collection, valid range from 1 to Count.

MoveItem

This method moves an item specified by OldIndex from OldIndex to NewIndex.

Description	Definition	
Syntax	MoveItem (OldIndex As Long, NewIndex As Long)	
Parameters	<i>OldIndex:</i>	[in] Index of item to remove valid range from 1 to Count.
	<i>NewIndex:</i>	[in] New index of the item after the move has occurred, valid range from 1 to Count.

Remove

This method removes the specified item from the Collection and releases the reference count to this item.

Description	Definition	
Syntax	Remove (ItemName As String)	
Parameter	<i>ItemName:</i>	[in] Name of item to remove.

RemoveByIndex

This method removes the specified item from the Collection and releases the reference count to this item.

Description	Definition	
Syntax	RemoveByIndex (Index As Long)	
Parameter	<i>Index:</i>	[in] Index of item to remove, valid range from 1 to Count.

Rename

This method Renames the item specified by Oldname from OldName to NewName.

Description	Definition	
Syntax	Rename (OldName As String, NewName As String)	
Parameters	<i>OldName:</i>	[in] Name of item to rename
	<i>NewName:</i>	[in] New name of item in Collection.

Tag

This property stores a variant for each item of the Collection.

Description	Definition	
Syntax	Tag (Index As Long) As Variant (read/write)	
Parameter	<i>Index:</i>	Specifies the item index, valid range from 1 to Count.

SCBCdrField

Description

This object contains the data that are evaluated and that should be extracted from the Document.

Type Definitions

CDRFieldState

This enumeration contains the state of the Field.

Available Types	Description
<i>CDRFieldStateAnalyzed</i>	Field is analyzed
<i>DRFieldStateEvaluated</i>	Field is evaluated
<i>CDRFieldStateFormatted</i>	Field is formatted
<i>CDRFieldStateReset</i>	Initial state of a Field
<i>CDRFieldStateValid</i>	Validity state of Field

Methods and Properties

ActiveTableIndex

This property reads the position where the Table is activated or activate the Table at given zero-based index.

Description	Definition
Syntax	ActiveTableIndex As Long (read/write)

Description	Definition
Example	<pre>'Initializes table and field references Set theEmptyTable = _ pWorkdoc.Fields("EmptyTable").Table(pWorkdoc.Fields("EmptyTable").ActiveTableIndex) Set theEmptyTableField = pWorkdoc.Fields("EmptyTable")</pre>

AddCandidate

This method adds a new Candidate to the Field based on the specified Word ID.

Description	Definition								
Syntax	AddCandidate (WordNr As Long, WordCount As Long, FilterID As Long, pIndex As Long)								
Parameters	<table border="1"> <tr> <td><i>WordNr:</i></td> <td>Specifies the Word index within the Word array of the Workdoc. Must be within 0 to pWorkdoc.WordCount - 1.</td> </tr> <tr> <td><i>WordCount:</i></td> <td>[in] Specifies the number of Words to use for the Candidate. If WordCount is greater than 1 the second word for the Candidate is defined with WordNr + 1, the third with WordNr + 2.</td> </tr> <tr> <td><i>FilterID:</i></td> <td>[in] This parameter can be used to store a filter identifier inside the Candidate. So later it is possible to see which filter expression has created the Candidate.</td> </tr> <tr> <td><i>pIndex:</i></td> <td>[out] Returns the index of the new Candidate within the Candidate array.</td> </tr> </table>	<i>WordNr:</i>	Specifies the Word index within the Word array of the Workdoc. Must be within 0 to pWorkdoc.WordCount - 1.	<i>WordCount:</i>	[in] Specifies the number of Words to use for the Candidate. If WordCount is greater than 1 the second word for the Candidate is defined with WordNr + 1, the third with WordNr + 2.	<i>FilterID:</i>	[in] This parameter can be used to store a filter identifier inside the Candidate. So later it is possible to see which filter expression has created the Candidate.	<i>pIndex:</i>	[out] Returns the index of the new Candidate within the Candidate array.
	<i>WordNr:</i>	Specifies the Word index within the Word array of the Workdoc. Must be within 0 to pWorkdoc.WordCount - 1.							
	<i>WordCount:</i>	[in] Specifies the number of Words to use for the Candidate. If WordCount is greater than 1 the second word for the Candidate is defined with WordNr + 1, the third with WordNr + 2.							
	<i>FilterID:</i>	[in] This parameter can be used to store a filter identifier inside the Candidate. So later it is possible to see which filter expression has created the Candidate.							
<i>pIndex:</i>	[out] Returns the index of the new Candidate within the Candidate array.								
Example	<pre>Private Sub MyField_PostAnalysis(pField As SCBCdrField, pWorkdoc As SCBCdrWorkdoc) Dim cindex as long, count as long, id as long 'add a new candidate to the field if pWorkdoc.Wordcount > 42 then 'use the 42th word as new candidate count = 1 'wordcount of new candidate id = 0 'rule-id for later backtracing pField.AddCandidate 42, count, id, cindex 'cindex is the new index of the candidate end if End Sub</pre>								

AddCandidate2

This method adds a new Candidate to the Field based on the specified Worktext.

Description	Definition	
Syntax	AddCandidate2 (pWorktext As ISBCCroWorktext, pIndex As Long)	
Parameters	<i>pWorktext:</i>	[in] Must be an initialized Worktext as it was created calling a SCBCroZone.Recognize method.
	<i>pIndex:</i>	[out] Returns the index of the new Candidate within the Candidate array.

AddTable

This method adds a Table into the Table array of this Field.

Description	Definition
Syntax	AddTable ()

BoostDigitsOnly

This method sets or returns whether only digits should be boosted.

Description	Definition
Syntax	BoostDigitsOnly as Boolean

BoostField

This method sets or returns whether a field should be boosted.

Description	Definition
Syntax	BoostField as Boolean

Candidate

This property returns a Candidate of the Field.

Description	Definition	
Syntax	Candidate (index As Long) As ISBCCroCandidate (read only)	
Parameters	<i>Index:</i>	Index of the Candidate. Valid indices are 0 to CandidateCount-1

CandidateByFilterID

This method finds the first candidate by specified filter ID or creates a new one if no such candidate is found.

Description	Definition	
Syntax	CandidateByFilterID (ByVal FilterID As Long, ByVal CreateNew As Boolean, pCandidateIndex As Long) as ISCBCdrCandidate	
Parameters	<i>FilterID:</i>	The filter ID that is used to find the candidate.
	<i>CreateNew:</i>	Create a new candidate if set to True.
	<i>pCandidateIndex:</i>	The index of the found candidate

CandidateCount

This method returns the number of candidates for a field.

Description	Definition
Syntax	CandidateCount As Long

Changed

This property returns the changed state of the Field. If the changed state becomes TRUE, the field must be validated even if it was previously validated.

Description	Definition
Syntax	Changed As Boolean (read/write)

CustomDetailsString

This property sets or returns the CustomDetailsString.

Description	Definition
Syntax	CustomDetailsString as String

CustomStatusLong

This property sets or returns the CustomStatusLong.

Description	Definition
Syntax	CustomStatusLong as Long

DeleteLine

This method deletes a line from a specific index position.

Description	Definition	
Syntax	DeleteLine (LineIndex As Long)	
Parameter	<i>LineIndex:</i>	Index of Line, zero-based indexing
Example	<pre> `This loop deletes the existing line objects in the field: Dim lngLineCounter As Long For lngLineCounter = (pField.LineCount - 1) To 0 Step -1 pField.DeleteLine(lngLineCounter) Next `Then add as many lines as required and populate with the required string: pField.InsertLine(0) pField.Line(0)="Line1" pField.InsertLine(1) pField.Line(1)="Line2" </pre>	

DeleteTable

This method deletes a Table from the Table array of this Field.

Description	Definition	
Syntax	DeleteTable (TableIndex As Long)	
Parameter	<i>TableIndex:</i>	Zero-based Index of the Table

ErrorDescription

This property stores the reason if a script validation could not be performed successfully.

Description	Definition
Syntax	ErrorDescription As String (read/write)

Description	Definition
Example	<pre> Private Sub Number_Validate(pField As SCBCdrField, pWorkdoc As SCBCdrWorkdoc, pValid As Boolean) if pValid = FALSE then 'Standard validation returns invalid, stop here exit sub end if 'Perform additional check for number format if IsValidNumber(pField) = FALSE then pValid = FALSE pField.ErrorDescription = "Field is not a valid number" end if End Sub </pre>

ExternalText

This method sets or returns the extended text.

Description	Definition
Syntax	ExternalText As String

FieldID

This read-only property returns the internally used FieldID.

Description	Definition
Syntax	FieldID As Long (read only)

FieldState

This property sets or returns the current execution state of the Field.

Description	Definition
Syntax	FieldState As CDRFieldState (read/write)

Description	Definition
Example	<pre> Private Sub Document_PreExtract(pWorkdoc As SCBCdrWorkdoc) Dim MyResult as string MyResult = DoSomeMagic(pWorkdoc) if (len(MyResult) > 0) then 'assign result to a single field pWorkdoc.Fields("Number") = MyResult; 'skip defined analysis and evaluation methods pWorkdoc.Fields("Number").FieldState = CDRFieldStateEvaluated end if end Sub </pre>

FieldVersion

This property returns the field data of the specified version.

Description	Definition		
Syntax	FieldVersion As String (ByVal index As Long)		
Parameter	<table border="1"> <tr> <td><i>Index:</i></td> <td>ByVal index As Long</td> </tr> </table>	<i>Index:</i>	ByVal index As Long
<i>Index:</i>	ByVal index As Long		

FindCandidate

This method searches inside the list of Candidates if there is a Candidate based on the specified WordID.

Description	Definition				
Syntax	FindCandidate (WordID As Long, pCandIndex As Long)				
Parameters	<table border="1"> <tr> <td><i>WordID:</i></td> <td>[in] Specifies a WordID inside the Word array of the Workdoc searched for.</td> </tr> <tr> <td><i>pCandIndex:</i></td> <td>[out] Contains the index of the Candidate if someone was found or -1 if no Candidate was found.</td> </tr> </table>	<i>WordID:</i>	[in] Specifies a WordID inside the Word array of the Workdoc searched for.	<i>pCandIndex:</i>	[out] Contains the index of the Candidate if someone was found or -1 if no Candidate was found.
	<i>WordID:</i>	[in] Specifies a WordID inside the Word array of the Workdoc searched for.			
<i>pCandIndex:</i>	[out] Contains the index of the Candidate if someone was found or -1 if no Candidate was found.				

FindCandidateByPos

This is a method to find a candidate by its position.

Description	Definition		
Syntax	FindCandidateByPos (ByVal Page as Long, ByVal Param1 as Long, ByVal Left as Long, ByVal Top as Long, ByVal Width as Long, By Val Height as Long, CandidateIndex as Long) as ISCBCdrCandidate		
Parameters	<table border="1"> <tr> <td><i>ByVal Page:</i></td> <td>Long</td> </tr> </table>	<i>ByVal Page:</i>	Long
<i>ByVal Page:</i>	Long		

Description	Definition	
	<i>ByVal Param1:</i>	Long
	<i>ByVal Left:</i>	Long
	<i>ByVal Top:</i>	Long
	<i>ByVal Width:</i>	Long
	<i>ByVal Height:</i>	Long
	<i>CandidateIndex:</i>	Long

FormattedText

This property can be set only in FormatForExport field event. It is emptied before validate field event. For details, see [FormatForExport](#).

GetFirstCandidatePropsByPage

This is a method to get the first candidate's properties by page.

Description	Definition	
Syntax	CandidatePropsByPage (ByVal Page As Long, ByVal Param1 As Long, ByVal Left As Long, ByVal Top As Long, ByVal Width As Long, ByVal Height As Long, ByVal Text As String, ByVal Weight As Double) as Long	
Parameter	<i>CandidateProps:</i>	ByVal Page As Long ByVal Param1 As Long ByVal Left As Long ByVal Top As Long ByVal Width As Long ByVal Height As Long ByVal Text As String ByVal Weight As Double

GetNextCandidatePropsByPage

This is a method to get the next candidate's properties by page.

Description	Definition	
Syntax	CandidatePropsByPage (ByVal Left As Long, ByVal Top As Long, ByVal Width As Long, ByVal Height As Long, ByVal Text As String, ByVal Weight As Double) as Long	

Description	Definition	
Parameter	<i>CandidateProps:</i>	ByVal Left As Long ByVal Top As Long ByVal Width As Long ByVal Height As Long ByVal Text As String ByVal Weight As Double

GetUniqueEntryID

This method retrieves other column values for the specified pool entry.

Note The Remote Matching Service and alphanumeric indexes do not support this method. For these, use the `UniqueID` property as demonstrated in the example below.

Description	Definition	
Syntax	GetUniqueEntryId (IdHigh As Long, IdLow As Long)	
Parameters	<i>IdHigh:</i>	[out] Upper part of the 64-bit unique ID.
	<i>IdLow:</i>	[out] Lower part of the 64-bit unique ID.
Example	<pre>Public Function GetASSAInfo (pworkdoc as SCBCdrPROJLib.SCBCdrWorkdoc, cand as SCBCdrWkDocLib.SCBCdrCandidate) As String 'Function input: Workdoc, ASSA Candidate Dim lNumericIdHigh As Long Dim lNumericIdLow As Long GetASSAInfo="" If cand.IsIDAlphNum = True Then GetASSAInfo = cand.UniqueID Else GetASSAInfo = Cand.GetUniqueEntryID(lNumericIDhigh, lnumericIdLow) End If End Function</pre>	

Height

This property sets or returns the height of the Field in pixels

Description	Definition
Syntax	Height As Long (read/write)

Description	Definition
Example	<pre>'copy the positional information to the new object pCopyField.Height = pField.Height</pre>

InsertLine

This method inserts a line at the given LineIndex in a Field.

Description	Definition	
Description	Definition	
Syntax	InsertLine (LineIndex As Long)	
Parameter	<i>LineIndex:</i>	Zero-based LineIndex at which position line has to be inserted.
Example	<p>The following script code should be used when attempting to insert new lines to a field in custom script:</p> <pre>'This loop deletes the existing line objects in the field: Dim lngLineCounter As Long For lngLineCounter = (pField.LineCount - 1) To 0 Step -1 pField.DeleteLine(lngLineCounter) Next 'Then add as many lines as required and populate with the required string: pField.InsertLine(0) pField.Line(0)="Line1" pField.InsertLine(1) pField.Line(1)="Line2" Attempting to use pfield.text="Line1" + VbCrLf & "Line2" will not work.</pre>	

IsIDAlphNum

Sets / returns whether an Associative Search Engine's (ASE) unique ID is alphanumeric. If TRUE, than the field is AlphaNumeric, if FALSE than the field is numeric.

When accessing this attribute from the CdrWorkDoc Object, the property is taken from the ASE configured for the Classification Field.

When accessing this attribute from the CdrField Object, the property is taken directly from the ASE Field for the class.

In some complex project configuration, the following considerations may apply where direct access to Fields is needed to look directly at the ASE Field attribute rather than the workdoc.

- 1) When the project hierarchy has a Parent class where the Classification Field UniqueID is of Type A (eg AlphaNumeric) but the same field on a Child Class if of Type B (eg Numeric). In this instance accessing the workdoc.IsIDAlphNum will always return the Parent setting, thus requiring the scripter to access the Cedar Field property directly.
- 2) When the project has many ASE fields, and the IsIDAlphNum is being retrieved/set.

Description	Definition
Syntax	IsIDAlphNum As Boolean
Example	<pre>Dim pFieldDef As SCBCdrFieldDef Dim pSettings As SCBCdrSupExSettings Dim bIsAlphNum As Boolean Set pFieldDef = Project.AllClasses(pWorkdoc.DocClassName).Fields("MyASSA") Set pSettings = pFieldDef.AnalysisSetting(Project.DefaultLanguage) bIsAlphNum = pSettings.IsIDAlphNum</pre>

LastModificationEndDate

This property sets or returns the LastModificationEndDate.

Description	Definition
Syntax	LastModificationEndDate As Date

LastModificationEndDateAsFileTimeUtc

This property sets or returns the LastModificationEndDateAsFileTimeUtc.

Description	Definition
Syntax	LastModificationEndDateAsFileTimeUtcAs Date (read/write)

Left

This property sets or returns the left border of the Field in pixels.

Description	Definition
Syntax	<code>Left As Long (read/write)</code>

Line

This property sets or returns the text of a single line.

Description	Definition		
Syntax	<code>Line (index As Long) As String (read/write)</code>		
Parameter	<table border="1"> <tr> <td><i>Index:</i></td> <td>Index of the line must be from 0 to LineCount-1.</td> </tr> </table>	<i>Index:</i>	Index of the line must be from 0 to LineCount-1.
<i>Index:</i>	Index of the line must be from 0 to LineCount-1.		

LineCaption

If a Field has more than one line, it is possible to assign a caption to each line to provide information about the content of the line.

Description	Definition		
Syntax	<code>LineCaption (index As Long) As String (read/write)</code>		
Parameter	<table border="1"> <tr> <td><i>Index:</i></td> <td>Index of the line, must be from 0 to LineCount-1</td> </tr> </table>	<i>Index:</i>	Index of the line, must be from 0 to LineCount-1
<i>Index:</i>	Index of the line, must be from 0 to LineCount-1		

LineCount

This property returns the number of lines of a multi-line header field. This equals the number of Worktext objects (In Perceptive Intelligent Capture, each line of a multi-line header field is represented by a separate individual Worktext object).

Description	Definition
Syntax	<code>LineCount As Long (read/write)</code>

LineWorktext

This property provides access to the Worktext of each single line of the Field. The line index corresponds to the Worktext object.

Description	Definition		
Syntax	<code>LineWorktext (index As Long) As ISCBCroWorktext (read/write)</code>		
Parameter	<table border="1"> <tr> <td><i>Index:</i></td> <td>Index of the line, must be from 0 to LineCount-1.</td> </tr> </table>	<i>Index:</i>	Index of the line, must be from 0 to LineCount-1.
<i>Index:</i>	Index of the line, must be from 0 to LineCount-1.		

MultilineText

This property sets or returns multiline text for all lines at once that are separated with line break chars (same as "vbCrLf" in WinWrap script).

Description	Definition
Syntax	MultilineText As String (read/write)

Name

This property returns the name of the Field as it was defined within the design environment.

Description	Definition
Syntax	Name As String (read only)

PageNr

This property sets or returns the DocPage number where the Field is located.

Description	Definition
Syntax	PageNr As Long (read/write)

PutUniqueEntryId

This method sets the unique ID (64 bit) for the field content from associative search pool.

Note The Remote Matching Service and alphanumeric indexes do not support this method.

Description	Definition
Syntax	PutUniqueEntryId (IdHigh As Long, IdLow As Long)
Parameters	<i>IdHigh:</i> [in] Upper part of the 64-bit unique ID.
	<i>IdLow:</i> [in] Lower part of the 64-bit unique ID.
Example	<pre>Dim intNewCandidate As long Dim lngUniqueID As Long lngUniqueID = pWorkdoc.Fields("VendorASSA").Candidate(intNewCandidate).FilterID pWorkdoc.Fields("VendorASSA").PutUniqueEntryId(0, lngUniqueID)</pre>

RemoveCandidate

This method removes a Candidate from the Candidate array.

Description	Definition	
Syntax	RemoveCandidate (CandIndex As Long)	
Parameter	<i>CandIndex:</i>	Zero-based Candidate Index.

SkipTrainingWithEngine

This property identifies whether the specified trainable engine has to skip this field in the training process.

Description	Definition	
Syntax	SkipTrainingWithEngine (bstrEngineName As String) As Boolean (read/write)	
Parameter	<i>bstrEngineName:</i>	Name of the extraction engine.

Table

This property returns the Table object from an array of Tables of this Field at a specified index.

Description	Definition	
Syntax	Table (index As Long) As ISCBCdrTable (read only)	
Parameter	<i>Index:</i>	Position of a Table in an array of Tables, zero-based indexing.

TableCount

This property returns the number of Tables according to the Field.

Description	Definition	
Syntax	TableCount As Long (read only)	

Tag

Use this property to store an arbitrary variant in the Field.

Description	Definition	
Syntax	Tag As Variant (read/write)	

Text

Use this property to read and write the text of the Field. In case of multi-line Fields, the Text property refers to all lines at once as one single string, combining lines with spaces in between.

Description	Definition
Syntax	Text As String (read/write)

Top

This property sets or returns the top border of the Field in pixels.

Description	Definition
Syntax	Top As Long (read/write)

TrainedWithEngine

This property returns whether this field is trained with the specified engine.

Description	Definition
Syntax	TrainedWithEngine (bstrEngineName As String) As Boolean (read only)
Parameter	<i>bstrEngineName</i> : Name of the Engine

Valid

This property sets or returns the valid state of the Field.

Description	Definition
Syntax	Valid As Boolean (read/write)

Width

This property sets or returns the width of the Field in pixels.

Description	Definition
Syntax	Width As Long (read/write)

Worktext

This property provides access to the Worktext of the Field. In case of multi-line Fields, the Worktext property refers to the first Worktext the header field consists of, which represents the first line of the multi-line header field.

Description	Definition
Syntax	Worktext As ISBCCroWorktext (read/write)

SCBCdrCandidate

Description

Cedar Candidates are generated during the analysis step and are representing possible results of a Field.

Methods and Properties

Attractor

This property returns the attractor of the Candidate by a zero-based index.

Description	Definition
Syntax	Attractor (index As Long) As ISBCdrAttractor (read only)
Parameter	<i>Index:</i> Specifies the index in the attractor array, must be between 0 and AttractorCount - 1.

AttractorCount

This property returns the number of attractors for this Candidate.

Description	Definition
Syntax	AttractorCount As Long (read only)

CopyToField

Use this method to copy all required properties from the Candidate to the Field result.

Description	Definition
Syntax	CopyToField (pField As ISBCdrField)
Parameter	<i>pField:</i> Reference to the Field containing the Candidate. States which field should get the values from the Candidate.

FilterID

This is the FilterID value as it was specified by the AddCandidate method of the Field

Description	Definition
Syntax	<code>FilterID As Long (read only)</code>
Example	<pre>Dim intNewCandidate As long Dim lngUniqueID As Long lngUniqueID = pWorkdoc.Fields("VendorASSA").Candidate(intNewCandidate).FilterID pWorkdoc.Fields("VendorASSA").PutUniqueEntryId(0, lngUniqueID)</pre>

FormatConfidence

This property sets or returns the confidence of the string match algorithm performed by the format search engine that has created the Candidate.

Description	Definition
Syntax	<code>FormatConfidence As Double (read/write)</code>

Height

This property returns the height of the Candidate in pixels.

Description	Definition
Syntax	<code>Height As Long (read only)</code>

KeepSpaces

This property specifies if the text created from several Words should keep the spaces between these Words or not.

Description	Definition
Syntax	<code>KeepSpaces As Boolean (read/write)</code>

Left

This property returns the left border of the Candidate in pixels.

Description	Definition
Syntax	<code>Left As Long (read only)</code>

Line

This property returns the text of a single line. A Candidate can consist of one or more lines.

Description	Definition	
Syntax	Line (index As Long) As String (read only)	
Parameter	<i>Index:</i>	Index of the Line, must be from 0 to LineCount-1.

LineCaption

If a Candidate has more than one line, it is possible to assign a caption to each line to provide information about the content of the line.

Description	Definition	
Syntax	LineCaption (index As Long) As String (read/write)	
Parameter	<i>Index:</i>	Index of the line, must be from 0 to LineCount – 1

LineCount

This property returns the number of lines of the Candidate or can be used to set the number of lines of a Field.

Description	Definition	
Syntax	LineCount As Long (read/write)	

LineWordCount

This property returns the number of words of the specified line.

Description	Definition	
Syntax	LineWordCount (index As Long) As Long (read only)	
Parameter	<i>Index:</i>	Index of the line.

LineWordID

This property returns the Word ID of the specified Line and Word index.

Description	Definition	
Syntax	LineWordID (LineIndex As Long, WordIndex As Long) As Long (read only)	
Parameters	<i>LineIndex:</i>	Index of the Line, must be from 0 to LineCount-1.
	<i>WordIndex:</i>	Index of the Word within the Line.

LineWorktext

This property returns the Worktext object of the single line specified by the zero-based index within a multi-line Field.

Description	Definition	
Syntax	LineWorktext (index As Long) As ISBCCroWorktext (read/write)	
Parameter	<i>Index:</i>	Zero-based index of single line

PageNr

This property returns the DocPage number where the Candidate is located.

Description	Definition
Syntax	PageNr As Long (read only)
Example	<pre>Private Sub RestoreFieldPosition(pField As SCBCdrField, pCopyField As SCBCdrField) 'write the saved fields positional data back to the original field pField.PageNr = pCopyField.PageNr End Sub</pre>

RemoveAttractor

This method removes the attractor specified by index.

Description	Definition	
Syntax	RemoveAttractor (AttractorIndex As Long)	
Parameter	<i>AttractorIndex:</i>	Index of attractor to be removed, valid range from 0 to AttractorCount-1.

Text

This property returns the text of the Candidate.

Description	Definition
Syntax	Text As String (read only)

Top

This property returns the top border of the Candidate in pixels.

Description	Definition
Syntax	Top As Long (read only)

Weight

This property sets or returns the result of the evaluation, which is between 0 and 1.

Note The value can be higher than 1 (1 equals 100%) in case the sum of different single candidate weights resulting from position and environment of the candidate exceeds 100%. Candidates with more than 100% will also be accounted for selection.

Description	Definition
Syntax	Weight As Double (read/write)

Width

This property returns the width of the Candidate in pixels.

Description	Definition
Syntax	Width As Long (read only)

WordCount

This property returns the Word count of the Candidate.

Description	Definition
Syntax	WordCount As Long (read only)

WordID

This property returns the Word ID of the specified Word index within the first line.

Description	Definition	
Syntax	WordID (index As Long) As Long (read only)	
Parameter	<i>Index:</i>	Zero-based index of the Word within the line.

Worktext

This property returns the Worktext object of the first line.

Description	Definition
Syntax	Worktext As ISCBCroWorktext (read only)

SCBCdrTable

Descriptions

The Cedar Table object represents a logical Table in a Document that is assigned to a Cedar Field of a Workdoc.

Type Definitions

CDRTableHighlightMode

This lists the enumerations that contains the highlighting mode of a Table.

Available Types	Description
<i>CDRTableHighlightAllCells</i>	Highlight all cells of Table
<i>CDRTableHighlightAllColumns</i>	Highlight all columns of Table
<i>CDRTableHighlightAllColumnsAdvanced</i>	Advanced highlighting mode for both mapped and unmapped columns
<i>CDRTableHighlightAllRows</i>	Highlight all rows of Table
<i>CDRTableHighlightCell</i>	Highlight particular cell (as set by HighlightColumnIndex and HighlightRowIndex)
<i>CDRTableHighlightColumn</i>	Highlight column (as set by HighlightColumnIndex)
<i>CDRTableHighlightNothing</i>	Highlight nothing
<i>CDRTableHighlightRow</i>	Highlight row (as set by HighlightRowIndex)

Available Types	Description
<i>CDRTableHighlightTable</i>	Highlight whole Table

CDRLocation

This table lists the enumerations that contain the location of a row, column, or cell in a Table.

Available Types	Description
<i>CDRLocationBottom</i>	Bottom corner coordinate
<i>CDRLocationLeft</i>	Left corner coordinate
<i>CDRLocationRight</i>	Right corner coordinate
<i>CDRLocationTop</i>	Top corner coordinate

Methods and Properties

AddColumn

This method adds a new column to a Table. It returns the index of the new column (zero-based).

Description	Definition	
Syntax	<code>AddColumn (ColumnName As String) As Long</code>	
Parameter	<i>ColumnName:</i>	[in] Name of column

AddRow

This method adds a new row to a Table. Returns the index of the new row (zero-based).

Description	Definition
Syntax	<code>AddRow () As Long</code>

AddUMColumn

This method adds a new unmapped column to a Table. Returns the index of the new unmapped column.

Description	Definition	
Syntax	<code>AddUMColumn (pUMColumnIndex As Long)</code>	
Parameter	<i>pUMColumnIndex:</i>	The method returns the zero-based index of the new column to this parameter.

AppendRows

This method appends new rows over the specified range within the document.

Description	Definition	
Syntax	AppendRows (top As Long, height As Long, PageNumber As Long)	
Parameters	<i>Top:</i>	Top of region used for creation or new rows
	<i>Height:</i>	Height of region used for creation or new rows
	<i>PageNumber:</i>	DocpPage number of region

CellColor

This property sets or returns the color of the Table cell.

Description	Definition	
Syntax	CellColor (IsValid As Boolean) As OLE_COLOR (read/write)	
Parameter	<i>IsValid:</i>	Flag indicating if color refers to valid or invalid Table cells

CellLocation

This property sets or returns the location of the Table cell.

Description	Definition	
Syntax	CellLocation (Column As Variant,RowIndex As Long, Location As CDRLocation) As Long (read/write)	
Parameters	<i>Column:</i>	Zero-based index or name of column
	<i>RowIndex:</i>	Zero-based index of row
	<i>Location:</i>	Location parameter

CellText

This property sets or returns the text of the Table cell.

Description	Definition	
Syntax	CellText (Column As Variant,RowIndex As Long) As String (read/write)	
Parameters	<i>Column:</i>	Zero-based index or name of column
	<i>RowIndex:</i>	Zero-based index of row

Description	Definition
Example	<pre> Private Sub MyTableField_ValidateCell(pTable As SCBCdrPROJLib.SCBCdrTable, pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc, ByVal Row As Long, ByVal Column As Long, pValid As Boolean) Select Case Column Case 0: 'check date in column 0 if CheckDate(pTable.CellText(Column, Row)) = FALSE then pValid = FALSE pTable. CellValidationErrorDescription(Column, Row) = "Invalid date" end if Case 2: 'check order number in column 2 if CheckOrderNumber(pTable.CellText(Column, Row)) = FALSE then pValid = FALSE pTable. CellValidationErrorDescription(Column, Row) = "Invalid order number" end if End Select End Sub </pre>

CellValid

This property sets or returns the validity flag of the Table cell.

Description	Definition	
Syntax	CellValid (Column As Variant,RowIndex As Long) As Boolean (read/write)	
Parameters	<i>Column:</i>	Zero-based index of name of column
	<i>RowIndex:</i>	Zero-based index of row
Example	<pre> ' Makes table object valid theEmptyTable.CellValid(0,0) = True theEmptyTable.CellValid(1,0) = True </pre>	

CellValidationErrorDescription

This property sets or returns the ErrorDescription for the cell validation.

Description	Definition	
Syntax	CellValidationErrorDescription (Column As Variant,RowIndex As Long) As String (read/write)	
Parameters	<i>Column:</i>	Zero-based index or name of column
	<i>RowIndex:</i>	Zero-based index of row
Example	<pre> Private Sub MyTableField_ValidateCell(pTable As SCBCdrPROJLib.SCBCdrTable, pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc, ByVal Row As Long, ByVal Column As Long, pValid As Boolean) Select Case Column Case 0: 'check date in column 0 if CheckDate(pTable.CellText(Column, Row)) = FALSE then pValid = FALSE pTable. CellValidationErrorDescription(Column, Row) = "Invalid date" end if Case 2: 'check order number in column 2 if CheckOrderNumber(pTable.CellText(Column, Row)) = FALSE then pValid = FALSE pTable. CellValidationErrorDescription(Column, Row) = "Invalid order number" end if End Select End Sub </pre>	

CellVisible

This property sets or returns Visible flag of the Table cell (currently not used).

Description	Definition	
Syntax	CellVisible (Column As Variant,RowIndex As Long) As Boolean (read/write)	
Parameters	Column:	Zero-based index or name of column
	RowIndex:	Zero-based index of row

CellWorktext

This property sets or returns the Worktext object of the cell.

Description	Definition	
Syntax	CellWorktext (Column As Variant,RowIndex As Long) As ISCBCroWorktext (read/write)	
Parameters	Column:	Zero-based index or name of column
	RowIndex:	Zero-based index of row

CellWorktextChanged

This property sets or returns a flag that indicates whether the cell Worktext has changed.

Description	Definition	
Syntax	CellWorktextChanged (Column As Variant,RowIndex As Long) As Boolean (read/write)	
Parameters	Column:	Zero-based index or name of column
	RowIndex:	Zero-based index of row

Clear

This method clears the content of the Table., It removes all columns and all rows and resets all Table attributes.

Description	Definition
Syntax	Clear ()

ClearColumn

This method clears the content of an existing column.

Description	Definition	
Syntax	ClearColumn (Column As Variant)	
Parameter	<i>Column:</i>	Zero-based index or name of column

ClearRow

This method clears the content of an existing row.

Description	Definition	
Syntax	ClearRow (RowIndex As Long)	
Parameter	<i>RowIndex:</i>	Zero-based index of row.

ClearUMColumn

This method clears the content of an unmapped column.

Description	Definition	
Syntax	ClearUMColumn (UMColumnIndex As Long)	
Parameter	<i>UMColumnIndex:</i>	Zero-based index of unmapped column to be cleared.

ColumnColor

This property sets or returns the color of the column.

Description	Definition	
Syntax	ColumnColor (IsValid As Boolean) As OLE_COLOR (read/write)	
Parameter	<i>IsValid:</i>	Flag indicating if color refers to valid or invalid columns

ColumnCount

This property returns the number of columns.

Description	Definition	
Syntax	ColumnCount As Long (read only)	

ColumnExportEnable

This property sets or returns the ExportEnable flag of a column.

Description	Definition	
Syntax	ColumnExportEnable (Column As Variant) As Boolean (read/write)	
Parameter	<i>Column:</i>	Zero-based index or name of column

ColumnIndex

This property returns the column index for the name of a column.

Description	Definition	
Syntax	ColumnIndex (ColumnName As String) As Long (read only)	
Parameter	<i>ColumnName:</i>	Name of the column

ColumnLabelLocation

This property sets or returns the location of a column label (referring to first label line in case of multi-page Tables).

Description	Definition	
Syntax	ColumnLabelLocation (Column As Variant, Location As CDRLocation) As Long (read/write)	
Parameters	<i>Column:</i>	Zero-based index or name of column
	<i>Location:</i>	Location parameter

ColumnLabelText

This property sets or returns the column label.

Description	Definition	
Syntax	ColumnLabelText (Column As Variant) As String (read/write)	
Parameter	<i>Column:</i>	Zero-based index or name of column

ColumnLocation

This property sets or returns the location of the column.

Description	Definition	
Syntax	ColumnLocation (Column As Variant, PageNr As Long, Location As CDRLocation)As Long (read/write)	
Parameters	<i>Column:</i>	Zero-based index or name of column
	<i>PageNr:</i>	DocPage number
	<i>Location:</i>	Location parameter

ColumnMapped

This property sets or returns a flag that indicates whether a column has been mapped.

Description	Definition	
Syntax	ColumnMapped (Column As Variant) As Boolean (read/write)	
Parameter	<i>Column:</i>	Zero-based index or name of column

ColumnName

This property returns the name of a column.

Description	Definition	
Syntax	ColumnName (ColumnIndex As Long) As String (read only)	
Parameter	<i>ColumnIndex:</i>	Zero-based Index of column

ColumnValid

This property sets or returns a validity flag for a column. If the flag is set to false, the invalid state of the table field is not changed automatically.

Description	Definition	
Syntax	ColumnValid (Column As Variant) As Boolean (read/write)	
Parameter	<i>Column:</i>	Zero-based index or name of column

ColumnVisible

This property sets or returns the visible flag of a column. This method affects the visibility of the column in Verifier.

Description	Definition	
Syntax	ColumnVisible (Column As Variant) As Boolean (read/write)	
Parameter	<i>Column:</i>	Zero-based index or name of column
Example	theTableSettings.ColumnVisible(2) = True 'Set the Column visible to True to show, False to hide.	

DeleteColumn

This method deletes a column specified by its name or by index.

Description	Definition	
Syntax	DeleteColumn (Column As Variant)	
Parameter	<i>Column:</i>	Zero-based index or name of column

DeleteRow

This method deletes a row specified by an index.

Description	Definition	
Syntax	DeleteRow (RowIndex As Long)	
Parameter	<i>RowIndex:</i>	Zero-based index of row

DeleteUMColumn

This method deletes an unmapped column specified by index.

Description	Definition	
Syntax	DeleteUMColumn (UMColumnIndex As Long)	
Parameter	<i>UMColumnIndex:</i>	Zero-based index of unmapped column to be deleted

FieldName

This property sets or returns the name of the CdrField to which the CdrTable object belongs.

Description	Definition
Syntax	FieldName As String (read/write)

FillColumn

This method fills the column with Words of specified area. If the Table is empty, each text line is assigned to a Table row. Otherwise the existing row segmentation is used.

Description	Definition	
Syntax	FillColumn (left As Long, top As Long, width As Long, height As Long, PageNumber As Long, Column As Variant)	
Parameters	<i>Left:</i>	Left position of area in pixel
	<i>Top:</i>	Top of area in pixel
	<i>Width:</i>	Width of area in pixel
	<i>Height:</i>	Height of area in pixel
	<i>PageNumber:</i>	DocPage number of area
	<i>Column:</i>	Zero-based index or name of destination column

FooterLocation

This property sets or returns the location of the Table footer.

Description	Definition	
Syntax	FooterLocation (Location As CDRLocation) As Long (read/write)	
Parameter	<i>Location:</i>	Location parameter

FootPageNr

This property sets or returns the DocPage number of the Table footer.

Description	Definition
Syntax	FootPageNr As Long (read/write)

FooterText

This property sets or returns the text of the Table footer.

Description	Definition
Syntax	FooterText As String (read/write)

HeaderLocation

This property sets or returns the location of the Table header.

Description	Definition
Syntax	HeaderLocation (Location As CDRLocation) As Long (read/write)
Parameter	<i>Location:</i> Location parameter

HeaderPageNr

This property sets or returns the DocPage number of the Table header.

Description	Definition
Syntax	HeaderPageNr As Long (read/write)

HeaderText

This property sets or returns the text of the Table header.

Description	Definition
Syntax	HeaderText As String (read/write)

HighlightColumnIndex

This property sets or returns the index of the column to be highlighted.

Description	Definition
Syntax	HighlightColumnIndex As Long (read/write)

HighlightMode

This property sets or returns the HighlightMode of Table.

Description	Definition	
Syntax	HighlightMode As CDRTTableHighlightMode (read/write)	
	<i>CDRTTableHighlightTable</i>	Highlights whole table.
	<i>CDRTTableHighlightAllColumns:</i>	Highlights all columns
	<i>CDRTTableHighlightAllRows:</i>	Highlight all rows
	<i>CDRTTableHighlightAllCells:</i>	Highlights all cells
	<i>CDRTTableHighlightColumn:</i>	Highlights single column, as set by HighlightColumnIndex
	<i>CDRTTableHighlightRow:</i>	Highlights single row, as set by HighlightRowIndex.
	<i>CDRTTableHighlightCell:</i>	Highlights single cell, as set by HighlightColumnIndex and HighlightRowIndex.

HighlightRowIndex

This property sets or returns the index of the row to be highlighted.

Description	Definition
Syntax	HighlightRowIndex As Long (read/write)

HighlightUMColumnIndex

This property sets or returns the zero-based index of an unmapped column to be highlighted.

Description	Definition
Syntax	HighlightUMColumnIndex As Long (read/write)

InsertColumn

This method Inserts a new column after specified ColumnIndex.

Description	Definition	
Syntax	InsertColumn (ColumnIndex As Long, ColumnName As String)	
Parameters	<i>ColumnIndex:</i>	Zero-based index of existing column, after which the new column is inserted.

Description	Definition	
	<i>ColumnName:</i>	Name of new column

InsertRow

This method inserts a new row after specified RowIndex.

Description	Definition	
Syntax	InsertRow (RowIndex As Long)	
Parameter	<i>RowIndex:</i>	Zero-based index of existing row, after which the new row is inserted.

InsertUMColumn

This parameter inserts a new, unmapped column.

Description	Definition	
Syntax	InsertUMColumn (UMColumnIndex As Long)	
Parameters	<i>UMColumnIndex:</i>	Zero-based index of new column.

LabellinePageNr

This property sets or returns the DocPage number of the label line, which is the first occurrence for multi-page Tables.

Description	Definition	
Syntax	LabellinePageNr As Long (read/write)	

LocationExplicit

This property sets and returns the LocationExplicit flag.

Description	Definition	
Syntax	LocationExplicit As Boolean (read/write)	

MapColumn

This maps unmapped column. It transfers the content of an unmapped source column to a specified target column.

Description	Definition	
Syntax	MapColumn (UMColumnIndex As Long, Column As Variant)	

Description	Definition	
Parameters	<i>UMColumnIndex:</i>	Zero-based index of unmapped source column
	<i>Column:</i>	Zero-based index or name of destination column

MergeRows

This method merges two rows specified by two indices.

Description	Definition	
Syntax	<code>MergeRows (RowIndex1 As Long, RowIndex2 As Long)</code>	
Parameters	<i>RowIndex1:</i>	Zero-based index of row 1
	<i>RowIndex2:</i>	Zero-based index of row 2

RemoveAllColumns

This method removes all mapped table columns.

Description	Definition
Syntax	<code>RemoveAllColumns ()</code>

RemoveAllRows

This method removes all table rows.

Description	Definition
Syntax	<code>RemoveAllRows ()</code>

RemoveAllUMColumns

This method removes all unmapped table columns.

Description	Definition
Syntax	<code>RemoveAllUMColumns ()</code>

RowColor

This property sets or returns the color of the row.

Description	Definition
Syntax	<code>RowColor (IsValid As Boolean) As OLE_COLOR (read/write)</code>

Description	Definition	
Parameter	<i>IsValid:</i>	Flag indicating if color refers to valid or invalid rows

RowCount

This property returns the number of the rows.

Description	Definition
Syntax	RowCount As Long (read only)

RowLocation

This property sets or returns the location of the row.

Description	Definition	
Syntax	RowLocation (RowIndex As Long, Location As CDRLocation) As Long (read/write)	
Parameters	<i>RowIndex:</i>	Zero-based index of row
	<i>Location:</i>	Location parameter

RowNumber

This property sets or returns the actual number of row.

Description	Definition	
Syntax	RowNumber (RowIndex As Long) As Long (read/write)	
Parameter	<i>RowIndex:</i>	Zero-based index of row
Example	<pre>Private Sub Tabelle_ValidateCell(pTable As SCBCdrPROJLib.SCBCdrTable, pWorkdoc As_ SCBCdrPROJLib.SCBCdrWorkdoc, ByVal Row As Long, ByVal Column As Long, pValid As Boolean) Dim nCurrentRow, nRow, nLine As Integer While (nLine < pTable.RowCount) And (nRow = nCurrentRow) nRow = pTable.RowNumber(nLine) nLine = nLine + 1 Wend End Sub</pre>	

RowPageNr

This property sets or returns the DocPage number of a row

Description	Definition	
Syntax	RowPageNr (RowIndex As Long) As Long (read/write)	
Parameter	<i>RowIndex:</i>	Zero-based index of row

RowValid

This property sets or returns a validity flag of a row.

Description	Definition	
Syntax	RowValid (RowIndex As Long) As Boolean (read/write)	
Parameter	<i>RowIndex:</i>	Zero-based index of row

RowValidationErrorDescription

This property sets or returns an ErrorDescription for a row validation.

Description	Definition	
Syntax	RowValidationErrorDescription (RowIndex As Long) As String (read/write)	
Parameter	<i>RowIndex:</i>	Zero-based index of row

Description	Definition
Example	<pre> Private Sub MyTableField_ValidateRow(pTable As SCBCdrPROJLib.SCBCdrTable, pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc, ByVal Row As Long, pValid As Boolean) 'check if quantity * single price = total price Dim quantity as long Dim s_price as double, t_price as double 'all cells must already have a valid format quantity = CLng(pTable.CellText("Quantity", Row)) s_price = CLng(pTable.CellText("Single Price", Row)) t_price = CLng(pTable.CellText("Total Price", Row)) if quantity*s_price = t_price then pValid = TRUE else pValid = FALSE pTable.RowValidationErrorDescription(Row) = "Invalid quantity or amounts" end if End Sub </pre>

Significance

This property sets or returns the significance for the corresponding evaluation property of the Table.

Description	Definition		
Syntax	Significance (EvalPropIndex As Long) As Double (read/write)		
Parameters	<i>EvalPropIndex:</i>	Index of evaluation property:	
		1.	percentage of required columns identified
		2.	percentage of table columns mapped
		3.	average percentage of elements found in cell, for which element is required
		4.	Average no-overlap to neighboring cells (column view)
5.	Average no-overlap to neighboring cells (row view)		

SwapColumns

This method swaps two specified columns.

Description	Definition	
Syntax	SwapColumns (ColumnIndex1 As Long, ColumnIndex2 As Long)	
Parameters	ColumnIndex1:	Zero-based index of column 1
	ColumnIndex2:	Zero-based index of column 2

TableColor

This property sets or returns the color of the Table.

Description	Definition	
Syntax	TableColor (IsValid As Boolean) As OLE_COLOR (read/write)	
Parameter	IsValid:	Flag indicating if color refers to a valid or an invalid Table.

TableFirstPage

This property sets or returns the DocPage number of the beginning of a Table (must be set after creation of a Table, but cannot change afterwards).

Description	Definition
Syntax	TableFirstPage As Long (read/write)

TableLastPage

This property sets or returns the DocPage number of the end of a Table (must be set after creation of a Table, and after assigning the first DocPage, but must not change afterwards).

Description	Definition
Syntax	TableLastPage As Long (read/write)

TableLocation

This property sets or returns the location of a Table.

Description	Definition	
Syntax	TableLocation (PageNr As Long, Location As CDRLocation) As Long (read/write)	
Parameters	<i>PageNr:</i>	DocPage number
	<i>Location:</i>	Location parameter

TableValid

This property sets or returns a validity flag of the Table.

Description	Definition
Syntax	TableValid As Boolean (read/write)

TableValidationErrorDescription

This property sets or returns an ErrorDescription for the Table validation.

Description	Definition
Syntax	TableValidationErrorDescription As String (read/write)

Description	Definition
Example	<pre> Private Sub MyTableField_ValidateTable (pTable As SCBCdrPROJLib.SCBCdrTable, pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc, pValid As Boolean) 'calculate the sum of all amounts and compare with the net amount fields Dim tablesun as double, netamount as double Dim cellamount as double Dim row as long For row = 0 to pTable.RowCount-1 cellamount = CLng(pTable.CellText("Total Price", Row)) tablesun = tablesun + cellamount Next row 'now compare sum with the content of the net amount field netamount = CDBl(pWorkdoc.Fields("NetAmount").Text if netamount = tablesun then pValid = TRUE else pValid = FALSE pTable.TableValidationErrorDescription = "Sum of table amounts and field net amount are different" end if End Sub </pre>

Tag

This property sets and returns a tag associated with the Table.

Description	Definition
Syntax	Tag As String (read/write)

TotalSignificance

This property sets and returns the total significance of the Table.

Description	Definition
Syntax	TotalSignificance As Double (read/write)

UMCellColor

This property sets or returns a color of an unmapped Table cell.

Description	Definition
Syntax	UMCellColor As OLE_COLOR (read/write)

UMCellLocation

This property sets or returns the location of an unmapped Table cell.

Description	Definition	
Syntax	UMCellLocation (UMColumnIndex As Long,RowIndex As Long, Location As CDRLocation) As Long (read/write)	
Parameters	<i>UMColumnIndex:</i>	Zero-based index of unmapped column
	<i>RowIndex:</i>	Zero-based index of unmapped row
	<i>Location:</i>	Location parameter

UMCellText

This property sets or returns the text of an unmapped Table cell.

Description	Definition	
Syntax	UMCellText (UMColumnIndex As Long,RowIndex As Long) As String (read/write)	
Parameters	<i>UMColumnIndex:</i>	Zero-based index of unmapped column
	<i>RowIndex:</i>	Zero-based index of row

UMCellVisible

This property sets or returns a Visible flag of an unmapped Table cell.

Description	Definition	
Syntax	UMCellVisible (UMColumnIndex As Long,RowIndex As Long) As Boolean (read/write)	
Parameters	<i>UMColumnIndex:</i>	Zero-based index of unmapped column
	<i>RowIndex:</i>	Zero-based index of row

UMCellWorktext

This property sets or returns the Worktext Object of an unmapped cell.

Description	Definition	
Syntax	UMCellWorktext (UMColumnIndex As Long,RowIndex As Long) As ISBCCroWorktext (read/write)	
Parameters	<i>UMColumnIndex:</i>	Zero-based index of unmapped column
	<i>RowIndex:</i>	Zero-based index of row

UMColumnColor

This property sets or returns the color of an unmapped column.

Description	Definition
Syntax	UMColumnColor As OLE_COLOR (read/write)

UMColumnCount

This property returns the number of unmapped columns.

Description	Definition
Syntax	UMColumnCount As Long (read only)

UMColumnLabelLocation

This property sets or returns the location of an unmapped column label.

Description	Definition	
Syntax	UMColumnLabelLocation (UMColumnIndex As Long, Location As CDRLocation) As Long (read/write)	
Parameters	<i>UMColumnIndex:</i>	Zero-based index of unmapped column
	<i>Location:</i>	Location parameter

UMColumnLabelText

This property sets or returns the text of a label of an unmapped column.

Description	Definition	
Syntax	UMColumnLabelText (UMColumnIndex As Long) As String (read/write)	
Parameter	<i>UMColumnIndex:</i>	Zero-based index of unmapped column

UMColumnLocation

This property sets or returns the location of an unmapped column.

Description	Definition	
Syntax	UMColumnLocation (UMColumnIndex As Long, PageNr As Long, Location As CDRLocation) As Long (read/write)	
Parameters	<i>UMColumnIndex:</i>	Zero-based index of unmapped column
	<i>PageNr:</i>	DocPage number
	<i>Location:</i>	Location parameter

UMColumnVisible

This property sets or returns a Visible flag of an unmapped column(currently not used).

Description	Definition	
Syntax	UMColumnVisible (UMColumnIndex As Long) As Boolean (read/write)	
Parameter	<i>UMColumnIndex:</i>	Zero-based index of unmapped column

UnMapColumn

This method unmaps a column. It transfers content from a specified source column to a new, unmapped column.

Description	Definition	
Syntax	UnMapColumn (Column As Variant) As Long	
Parameter	<i>Column:</i>	Zero-based index or name of source column

WeightingFactor

This property sets or returns a Weighting Factor for a corresponding evaluation property.

Description	Definition	
Syntax	WeightingFactor (EvalPropIndex As Long) As Double (read/write)	
Parameters	<i>EvalPropIndex:</i>	Index of evaluation property:
		1. percentage of required columns identified
		2. percentage of table columns mapped
		3. average percentage of elements found in cell, for which element is required
		4. Average no-overlap to neighboring cells (column view)
5. Average no-overlap to neighboring cells (row view)		

SCBCdrTextblock

Description

This object represents a TextBlock on a Document. A TextBlock may contain one or more lines.

Methods and properties

Color

This property sets or returns the color that is used for TextBlock highlighting.

Description	Definition
Syntax	Color As OLE_COLOR (read/write)

Height

This property returns the height of the TextBlock in pixels.

Description	Definition
Syntax	Height As Long (read only)

Left

This property returns the left border of the TextBlock in pixels

Description	Definition
Syntax	Left As Long (read only)

PageNr

This property returns the number of the DocPage where the TextBlock is located.

Description	Definition
Syntax	PageNr As Long (read only)

Text

This property returns the whole text of the TextBlock.

Description	Definition
Syntax	Text As String (read only)

Top

This property returns the top border of the TextBlock in pixels.

Description	Definition
Syntax	Top As Long (read only)

Visible

This property controls whether the highlighted rectangle of the TextBlock should be visible if the TextBlock highlighting is enabled.

Description	Definition
Syntax	Visible As Boolean (read/write)

Weight

This property returns the block weight.

Description	Definition
Syntax	Weight As Double (read only)

Width

This property returns the width of the TextBlock in pixels.

Description	Definition
Syntax	Width As Long (read only)

WordCount

This property returns the number of Words that belong to the TextBlock.

Description	Definition
Syntax	WordCount As Long (read only)

WordID

Use this property as an index for the Word array of the Workdoc.

Description	Definition
Syntax	WordID (index As Long) As Long (read only)
Parameter	Index: Index of Word inside the TextBlock. Must be between 0 and WordCount -1

SCBCdrWord

Description

This object represents a textual Word of a Document.

Methods and Properties

Color

This property sets or returns the color that is used for highlighting checked Words.

Description	Definition
Syntax	Color As OLE_COLOR (read/write)

Height

This property returns the height of the Word in pixels.

Description	Definition
Syntax	Height As Long (read only)

Left

This property returns the left border of the Word in pixels.

Description	Definition
Syntax	Left As Long (read only)

PageNr

This property returns the number of the DocPage where the Word is located.

Description	Definition
Syntax	PageNr As Long (read only)

StartPos

This property returns the index of the first character of the Word inside the Worktext that is attached to the Workdoc.

Description	Definition
Syntax	StartPos As Long (read only)

Text

This property returns the text of the Word.

Description	Definition
Syntax	Text As String (read only)

TextLen

This property returns the number of characters of the Word.

Description	Definition
Syntax	TextLen As Long (read only)

Tooltip

This property sets or returns a tooltip string that displays in the checked Words highlight mode.

Description	Definition
Syntax	Tooltip As String (read/write)

Top

This property returns the top border of the Word in pixels.

Description	Definition
Syntax	Top As Long (read only)

Visible

If the Word highlighting for checked Words is enabled, this property sets or returns if the highlighted rectangle of the Word should be visible.

Description	Definition
Syntax	Visible As Boolean (read/write)

Width

This property returns the width of the Word in pixels.

Description	Definition
Syntax	Width As Long (read only)

Worktext

This property returns the Worktext object of the Word.

Description	Definition
Syntax	Worktext As ISCBCroWorktext (read only)

SCBCdrDocPage

Description

An object that represents a single DocPage within a Workdoc.

Type Definitions

CDRPageSource

The following table shows the enumeration that contains the Page source.

Available Types	Descriptions
CDRPageSourceFrontPage	Front Page assigned to Workdoc

Available Types	Descriptions
CDRPageSourceRearPage	Rear Page assigned to Workdoc
CDRPageSourceUnknown	Assigned Page to Workdoc is not known

CroLinesDir

This table shows the enumeration that specifies the direction of a line.

Available Types	Descriptions
CroLinesDir_Horizontal	Horizontal line
CroLinesDir_Vertical	Vertical line

CroLinesKooType

This table shows the enumerations that specifies coordinate types for a line.

Available Types	Descriptions
CroLinesKoorType_Angle	Angle of line
CroLinesKoorType_FirstPX	Starting abscissa of line
CroLinesKoorType_FirstPY	Starting ordinate of line
CroLinesKoorType_Length	Length of line
CroLinesKoorType_SecondPX	Ending abscissa of line
CroLinesKoorType_SecondPY	Ending ordinate of line
CroLinesKoorType_Thick	Thickness of line

Methods and Properties

DisplayImage

This property specifies the index of the image, which is displayed if the DocPage is visible inside the Viewer.

Description	Definition
Syntax	<code>DisplayImage As Long (read/write)</code>

DocIndex

This property specifies the index of the document inside the Workdoc to which this DocPage belongs.

Description	Definition	
Syntax	DocIndex (ImageIndex As Long) As Long (read only)	
Parameters	ImageIndex:	ImageIndex of the DocPage. Valid indices are 0 to ImageCount-1.

For additional information, see `DocFileName` and `DocFileType` properties of the `SCBCdrWordoc` object.

DocPageIndex

This property specifies the DocPage offset inside the document where this DocPage belongs.

Description	Definition	
Syntax	DocPageIndex (ImageIndex As Long) As Long (read only)	
Parameters	ImageIndex:	Index of the Image of the DocPage. Valid indexes are 0 to ImageCount-1.

GetResolution

This method returns the resolution of the specified Image in pixels.

Description	Definition	
Syntax	GetResolution (ImageIndex As Long, pXRes As Long, pYRes As Long)	
Parameters	ImageIndex:	[in] Index of the Image of the DocPage. Valid indices are 0 to ImageCount-1.
	pXRes:	[out] Will contain the x resolution after execution of the method.
	pYRes:	[out] Will contain the y resolution after execution of the method.

Height

This property returns the height of the DocPage in millimeter.

Description	Definition	
Syntax	Height As Double (read only)	

Image

This property returns an Image object for the specified index of the DocPage.

Description	Definition	
Syntax	Image (index As Long) As ISCBCroImage (read only)	
Parameters	Index:	Index of the Image of the DocPage. Valid indices are 0 to ImageCount-1.

ImageCount

This property returns the number of images available for the DocPage.

Description	Definition
Syntax	ImageCount As Long (read only)

Line

This property returns some specific property of a line, of some specific index,direction and coordinate type.

Description	Definition	
Syntax	Line (LineIndex As Long, LineDir As CroLinesDir, KooType As CroLinesKooType) As Long (read only)	
Parameters	LineIndex:	Zero-based index of the Line.
	LineDir:	Direction of Line (Horizontal or Vertical).
	KooType:	Information of a Line (starting X, starting Y, End X, End Y, and so on)

LinesCount

This property returns the number of horizontal or vertical Lines present in a document.

Description	Definition	
Syntax	LinesCount (LinesDir As CroLinesDir) As Long (read only)	
Parameters	LinesDir:	Direction of Line (Horizontal or Vertical).

OriginalDocumentFileName

This property allows the Scriptor to access the page property to examine the original file name for the image. This is useful for the Scriptor when attempting to track original file names for pages when a document is split or merged through the Verifier, Web Verifier or through the Page Separation engine.

Description	Definition
Syntax	<pre>pWorkdoc.Pages(0).OriginalDocumentFileName</pre>
Example	<pre>Private Sub CreateCollectionofPageOrgFileName(pWorkdoc As SCBCdrPROJLib.ISCBCdrWorkdoc) Dim WdcPageCount As Long ' Total Number of pages associated to the WorkDoc Dim CurPage As Long ' Current Page Number Dim OrgFilename As String ' Original File Name of the selected page Dim OrgFileNames() As String ' Array of Original File Name of all Pages of the WorkDocument WdcPageCount = pWorkdoc.PageCount ReDim OrgFileNames(WdcPageCount) For CurPage=0 To WdcPageCount-1 OrgFileNames(CurPage) = pWorkdoc.Pages(CurPage).OriginalDocumentFileName Next CurPage ' Write the original file name of all pages to log. For CurPage=0 To WdcPageCount-1 OrgFilename = OrgFileNames(CurPage) Project.LogScriptMessageEx CDRTTypeInfo, CDRSeverityLogFileOnly, " Original File Name of Page: " & CStr(CurPage+1) & " is [" & OrgFilename & "]" Next CurPage End Sub</pre>

PageSource

This property sets or returns a source of a DocPage. At the time of scanning, a DocPage can be directly assigned to Workdoc.

Description	Definition
Syntax	PageSource As CDRPageSource (read/write)

Rotate

This method rotates the underlying Images by the specified angle.

Description	Definition		
Syntax	Rotate (angle As Double)		
Parameter	<table border="1"> <tr> <td>Angle:</td> <td>Specifies the rotation angle in a range of -180.0 to +180.0.</td> </tr> </table>	Angle:	Specifies the rotation angle in a range of -180.0 to +180.0.
Angle:	Specifies the rotation angle in a range of -180.0 to +180.0.		

Rotation

This property returns the rotation angle as it was applied by the Rotate method.

Description	Definition
Syntax	Rotation As Double (read only)

Text

This property returns the text of the DocPage if OCR was already executed.

Description	Definition
Syntax	Text As String (read only)

Width

This property returns the width of the DocPage in millimeters.

Description	Definition
Syntax	Width As Double (read only)

SCBCdrFolder

Description

A Folder may represent an array of Workdocs within a Batch. A Folder may contain one or more Workdocs. During classification and extraction it is possible to access all Workdocs of the same Folder from script.

Methods and Properties

AddDocument

This method adds a Workdoc into a Folder at the last position and also returns the position where the Workdoc is appended.

Description	Definition	
Syntax	AddDocument (pWorkdoc As ISCBCdrWorkdoc, pNewIndex As Long)	
Parameters	pWorkdoc:	[in] Added Workdoc Object
	pNewIndex:	[out] Index position in a Folder where Workdoc is inserted

Clear

This method frees all the allocated memory by Folder.

Description	Definition
Syntax	Clear ()

Document

This property returns a Workdoc from the specified index of the document array of the Folder.

Description	Definition	
Syntax	Document (Index As Long) As ISCBCdrWorkdoc (read only)	
Parameter	Index:	The index of the Workdoc within the Folder. Must be from 0 to DocumentCount-1.

DocumentCount

This property returns the number of Workdocs within the Folder.

Description	Definition
Syntax	DocumentCount As Long (read only)

FolderData

This property provides the possibility to store and load a variable number of strings using any string as an index key.

Description	Definition	
Syntax	FolderData (Index As String) As String (read/write)	
Parameter	Index:	Any non-empty string that is used as an index key
Example	<pre>'writing FolderData pWorkdoc.Folder.FolderData("NumberFound") = "1" pWorkdoc.Folder.FolderData("Number") = pWorkdoc.Field("Number") 'reading FolderData if pWorkdoc.Folder.FolderData("NumberFound") = "1" then if len(pWorkdoc.Field("Number")) > 0 then 'takeover the result from the other workdoc pWorkdoc.Field("Number") = pWorkdoc.Folder.FolderData("Number") else 'compare results if pWorkdoc.Field("Number") = pWorkdoc.Folder.FolderData("Number") then 'found the same number again else 'found a different number on this document end if end if end if</pre>	

InsertDocument

This method inserts a Workdoc into a Folder at some given position.

Description	Definition	
Syntax	InsertDocument (Index As Long, pWorkdoc As ISCBCdrWorkdoc)	
Parameters	Index:	Index at which Workdoc is to be inserted, zero-based indexing
	pWorkdoc:	Workdoc object

MoveDocument

Use this method to move a Workdoc from one position to another position in a Folder.

Description	Definition	
Syntax	MoveDocument (FromIndex As Long, ToIndex As Long)	
Parameters	FromIndex:	Zero-based Index from where Workdoc is moved
	ToIndex:	Zero-based index where Workdoc is to be placed

RemoveDocument

This method removes a Workdoc from a given index from a Folder.

Description	Definition	
Syntax	RemoveDocument (index As Long)	
Parameter	Index:	Zero-based index in a Folder from where Workdoc is to be removed

Cedar Project Object Reference (SCBCdrPROJLib)

Description

The Cedar Project object represents a complete Project definition including all Document Classes, Field Definitions, and used classification and extraction methods.

Type Definitions

CDRClassifyMode

This type defines the algorithms for how the results of several classification engines can be combined.

Available Types	Description
CDRClassifyAverage	Average is computed
CDRClassifyMax	Maximum is computed
CDRClassifyWeightedDistance	For each cell of classification matrix difference between maximum of column and classification weight is calculated

CDRDatabaseWorkflowTypes

The Workflow Type of the batch. These are standard Perceptive Intelligent Capture workflow settings for batches.

This type of interface is a member of the Cedar project library.

Available Types	Description
CDRAutoTrainingFailed	Automatic document training failed
CDRAutoTrainingSucceeded	Automatic document training succeeded
CDRClassificationFailed	Automatic document classification failed
CDRClassificationSucceeded	Automatic document classification succeeded
CDRCleanupFailed	Automatic document cleanup failed
CDRCleanupSucceeded	Automatic document cleanup succeeded
CDRDocumentSeparationFailed	Automatic document separation failed
CDRDocumentSeparationSucceeded	Automatic document separation succeeded
CDREmailImportFailed	Automatic document import from exchange server failed
CDREmailImportSucceeded	Automatic document import from exchange server succeeded

Available Types	Description
CDRExportFailed	Automatic document export failed
CDRExportSucceeded	Automatic document export succeeded
CDRExtractionFailed	Automatic document extraction failed
CDRExtractionSucceeded	Automatic document extraction succeeded
CDRFileSystemExportFailed	Runtime Server based DB import from file system batches failed
CDRFileSystemExportSucceeded	Runtime Server based DB import from file system batches succeeded
CDRImportFailed	Automatic document import failed
CDRImportSucceeded	Automatic document import succeeded
CDRManualClassificationIncomplete	Manual document classification was not completed
CDRManualClassificationSucceeded	Manual document classification succeeded
CDRManualDocumentSeparationIncomplete	Manual document separation failed
CDRManualDocumentSeparationSucceeded	Manual document separation succeeded
CDRManualFinalValidationFullyIncomplete	Manual final document validation was not completed
CDRManualFinalValidationSucceeded	Manual final document validation succeeded
CDRManualTrainingFailed	Manual document training failed
CDRManualTrainingSucceeded	Manual document training succeeded
CDRModifiedByDesignerApplication	The document was saved via Designer application without changing its workflow status
CDRModifiedByVerifierApplication	The document was saved via Verifier application without changing its workflow status
CDROCRFailed	Automatic document OCR failed
CDROCRSucceeded	Automatic document OCR succeeded
CDRPartialManualValidationIncomplete	Partial manual document validation was not completed
CDRPartialManualValidationSucceeded	Partial manual document validation succeeded
CDRReserved	Reserved for system use
CDRReset	Initial state of document

Available Types	Description
CDRScanningFailed	Images scanning failed
CDRScanningSucceeded	Images scanning succeeded

CdrSLWDifferentResultsAction

When the Template and Associative Search engines determine different results during classification, there are different options how the program should continue the processing.

Available Types	Description
CdrDoNothing	Let Verifier user decide to skip special processing altogether.
CdrDoSmartDecision	Make a smart decision, for example, the machine makes the decision for the classification. Note The system determines which one is the right DocClass based on an algorithm that compares the results of the associative search and the template classification. You can select this feature from the Supervised Learning tab in the Designer application.
CdrUseDocumentClassName	Automatically assign current document class name to the supplier field content.
CdrUseSupplierField	Automatically assign supplier field content to the document class name.

CdrForceValidationMode

This table defines the options for Force Validation.

Available Types	Description
CdrForceValDefault	CdrForceValidationModeDefault: ForceValidationMode inherited
CdrForceValForbidden	CdrForceValidationModeForbidden: ForceValidation (3*return) not allowed
CdrForceValPermitted	CdrForceValidationModePermitted: ForceValidation (3*return) allowed

CdrLicenseCounter

This table lists the data type definitions for all available license counters to be interrogated in script.

Available Types	Description
TLCFineReaderRemainingUnits	Remaining page units available to be processed by the FineReader licensing scheme.
TLCPeiodDocumentsClassified	Documents classified within the licensing period.
TLCPeiodDocumentsExported	Documents exported within the licensing period.
TLCPeiodDocumentsExtracted	Documents extracted within the licensing period.
TLCPeiodDocumentsOCRed	Documents OCRed within the licensing period.
TLCPeiodDocumentsProcessed	Docuemnts processed within the licensing period.
TLCPeiodDocumentsValidatedVerifier	Documents validated in Verifier within the licensing period.
TLCPeiodPagesImported	Pages imported within the licensing period.
TLCPeiodPagesOCRed	Pages OCRed within the licensing period.
TLCPeiodPagesProcessed	Pages Processed within the licensing period.
TLCTotalDocumentsClassified	Total Overall Classified documents.
TLCTotalDocumentsExported	Total Overall Exported documents.
TLCTotalDocumentsExtracted	Total Overall Extracted documents.
TLCTotalDocumentsOCRed	Total Overall OCRed documents.
TLCTotalDocumentsProcessed	TotalOverall Processed documents.
TLCTotalDocumentsValidatedVerifier	Total Overall documents validated in verifier.
TLCTotalPagesImported	Total Overall Pages Imported documents.
TLCTotalPagsOCRed	Total Overall Pages Processed documents.
TLCTotalPagesProcessed	Total Overall Pages Processed documents.

CdrLicenseFeatureName

These are the data type definitions for all available license features to be interrogated in script.

Each data type item below is represented in the license file and may appear. If the item appears in the license file, that means the feature is licensed and available for usage.

Available Types	Description
CDRLfnA2iACheckReader	The A2iA Check Reader License Feature.
CDRLfnA2iAFieldReaderCustom	The A2iA Field Reader custom License Feature.
CDRLfnA2iAFieldReaderSingleField	The A2iA Field Reader Single Field License Feature.
CDRLfnAddressAnalysisEngine	The Address Analysis Engine License Feature.
CDRLfnAddressAnalysisEngine2	The Address Analysis2 Engine License Feature.
CDRLfnASSAClassifyEngine	The ASSA Classification Engine License Feature.
CDRLfnAssociativeSearchEngine	The Associative Search Engine Field License Feature.
CDRLfnAutomaticLearningProcessing	The Automatic Learning Processing License Feature.
CDRLfnAutomaticLearningSupervising	The Learnset Manager License Feature.
CDRLfnBrainwareClassifyEngine	The Brainware Classifier License Feature.
CDRLfnBrainwareExtraction	The Brainware Extraction evaluation engine License Feature.
CDRLfnBrainwareFieldExtraction	The Brainware Field Extraction License Feature.
CDRLfnBrainwareLayoutClassification	The Brainware Layout Classifier engine License Feature.
CDRLfnBrainwareTableExtraction	The Brainware Table Extraction engine License Feature.
CDRLfnCairoImage	The Cairo Image License Feature.
CDRLfnCairoOMR	The Cairo OMR License Feature.
CDRLfnCaptureService	The Capture Service License Feature.
CDRLfnCleqsBarcode	The Cleqs Barcode OCR License Feature.
CDRLfnCloseLicensingPeriodBySlaveServer	Close Licensing Period by Slave Server
CDRLfnConcurrentVerifierSessionCount	The Web Verifier session count License Feature.
CDRLfnCustomer	The customer name License Feature.
CDRLfnCustomerID	The customer ID License Feature.

Available Types	Description
CDRLfnDesignerDesignLicense	The Designer application module License Feature.
CDRLfnDisableUpdateForVerifier	The ability to disable an update for verifier License Feature.
CDRLfnEMailsImporting	The EMail Importing License Feature.
CDRLfnFineReader	The FineReader4 License Feature.
CDRLfnFineReader5	The FineReader5 License Feature.
CDRLfnFineReader7	The FineReader7 License Feature.
CDRLfnFineReader8	The FineReader8 License Feature.
CDRLfnFirmwareHDSerialNumber	The Hard Disk Serial Number License Feature.
CDRLfnFormatAnalysisEngine	The Format Analysis engine License Feature.
CDRLfnFormsClassifyEngine	The Forms Classifier engine License Feature.
CDRLfnHardwareBindingEnabled	The HW binding enabled License Feature.
CDRLfnImageSizeClassification	The Image Size classifier engine License Feature.
CDRLfnIMailBasicComponents	The Imail components License Feature.
CDRLfnISIS	The ISIS driver License Feature.
CDRLfnKadmos	The Kadmos OCR License Feature.
CDRLfnKadmos4	The Kadmos4 OCR License Feature.
CDRLfnKofax	The Kofax driver License Feature.
CDRLfnLanguageClassifyEngine	The Language Classifier Engine License Feature.
CDRLfnLicenseCountingByReprocessing	The License Counting when reprocessing documents License Feature.
CDRLfnLicenseExpirationDate	The License expiration date License Feature.
CDRLfnLicenseVersion	The License version License Feature.
CDRLfnLicensingPeriodInDays	The License period in days License Feature.
CDRLfnMasterLicenseHexID	The License HexID License Feature.
CDRLfnNonImageDocumentsProcessing	The electronic document processing License Feature.
CDRLfnNonImageDocumentsProcessing	The electronic document processing License Feature.

Available Types	Description
CDRLfnOverallVerifierSessionCount	The overall verifier session count License Feature.
CDRLfnPeriodDocumentsClassified	The documents classified count License Feature.
CDRLfnPeriodDocumentsExported	The documents exported count License Feature.
CDRLfnPeriodDocumentsExtracted	The documents extracted count License Feature.
CDRLfnPeriodDocumentsOCRed	The documents OCRed count License Feature.
CDRLfnPeriodDocumentsProcessed	The documents Processed count License Feature.
CDRLfnPeriodDocumentsValidatedVerifier	The documents validated in verifier License Feature.
CDRLfnPeriodPagesImported	The Pages imported License Feature.
CDRLfnPeriodPagesOCRed	The Pages OCRed License Feature.
CDRLfnPeriodPagesProcessed	The Pages Processed License Feature.
CDRLfnPhraseClassifyEngine	The Phrase Classifier engine License Feature.
CDRLfnPrimaryDongleID	The Primary Dongle ID License Feature.
CDRLfnProcessedDocumentsPerDay	The Processed Documents Per Day License Feature.
CDRLfnQualitySoftBarcode	The QualitySoft Barcode OCR engine License Feature.
CDRLfnQualitySoftBarcodeDM	The QualitySoft Barcode DM OCR engine License Feature.
CDRLfnQualitySoftBarcodePDF417	The QualitySoft Barcode PDF OCR engine License Feature.
CDRLfnRecoStar	The RecoStar OCR engine License Feature.
CDRLfnSecondaryDongleID	The Secondary Dongle ID License Feature.
CDRLfnSecondaryHDSerialNumber	The Secondary Hard Disk Serial Number License Feature.
CDRLfnSecondaryMACAddress	The Secondary MAC Address License Feature.
CDRLfnSelfLearningManager	The Learnset Manager Module License Feature.
CDRLfnSERSCSI	The SCSI Driver License Feature.
CDRLfnServer	The RTS Server Module License Feature.
CDRLfnServerCount	The RTS Server count License Feature.
CDRLfnSupplierExtraction	The supplier extraction License Feature.

Available Types	Description
CDRLfnSVRS	The SVRS driver License Feature.
CDRLfnTableAnalysisEngine	The Table Analysis engine License Feature.
CDRLfnTemplateClassifyEngine	The Template Classifier engine License Feature.
CDRLfnTWAIN	The Twain Driver License Feature.
CDRLfnVerifier	The Verifier application module License Feature.
CDRLfnVerifierCount	The Verifier application count License Feature.
CDRLfnZoneAnalysisEngine	The Zone Analysis engine License Feature.

CdrMessageType

This type defines the different message types.

Available Types	Description
CDRTypeInfo	An informational message.
CDRTypeWarning	A warning message.
CDRTypeError	An error message.

CdrMessageSeverity

This type defines the different message severities.

Available Types	Description
CDRSeverityLogFileOnly	Store the message to the application log file only.
CDRSeveritySystemMonitoring	Store the message in the log file and forward it to the host instance's MMC console and to the System Monitoring service of the Runtime Server. This option is applicable when the call is executed from within the Runtime Server application only.
CDRSeverityEmailNotification	Store the message in the log file and forward it to the MMC console / System Monitoring view and send as an email to the system administrators via System Monitoring service of Runtime Server. This option is applicable when the call is executed from within the Runtime Server application only.

Methods and Properties

ActivateLicensing

This method is used as a call to enable license activation in the custom script. The call is used as a prerequisite prior to retrieving information for the licensing utilization. By calling activate licensing, the script creates a connection to the active license being utilized.

Description	Definition	
Syntax	ActivateLicensing (ModuleName as text, LicensePath as text)	
Parameters	ModuleName:	A text that represents the application activating licensing. Any value may be entered here.
	LicensePath:	A text that contains the location of the license share file that will be queried. The licensePath must be accessible from the location of the script execution. The license path must point to the Runtime.lic file explicitly.
Example	<pre>Code to retrieve licensing utilization information for active licensing counters. 'the Project represents the project library object. Dim theProject As New SCBCdrPROJLib.SCBCdrProject 'The location of the shared license file that is being updated. Dim LicenseShareLocation As String LicenseShareLocation="\\MasterRTS\License\Runtime.lic" 'Activate licensing within the code for project. This enables you to reference the license in the next command. theProject.ActivateLicensing("CustomEXE", LicenseShareLocation) 'Call the License Reporting function, this has several options available theProject.ReportLicensingStatus(True, SCBCdrPROJLib.CDRMessageSeverity.CDRSeverityLogFileOnly)</pre>	

For additional information see the [ReportLicensingStatus](#), [GetLicenseValueByName](#), and [GetLicenseValueByID](#) methods.

AllClasses

This property returns a Collection of all defined DocClasses of this Project.

Description	Definition
Syntax	AllClasses As ISCBCdrDocClasses (read only)

For more information, see [ISCBCdrDocClasses](#) and [ISCBCdrDocClass](#).

BaseClasses

This property returns a Collection that contains all defined BaseDocClasses.

Description	Definition
Syntax	<code>BaseClasses As ISCBCdrDocClasses (read only)</code>

For more information, see [ISCBCdrDocClasses](#) and [ISCBCdrDocClass](#).

ClassificationMode

This property returns the used classification mode.

Description	Definition
Syntax	<code>ClassificationMode As CDRClassifyMode (read/write)</code>

DefaultClassifyResult

This property returns the default DocClass name to which a document is redirected if no other DocClass fits.

Description	Definition
Syntax	<code>DefaultClassifyResult As String (read/write)</code>

DefaultLanguage

This property returns the language used as default.

Description	Definition
Syntax	<code>DefaultLanguage As String (read only)</code>

Description	Definition
Example	<pre> Private Sub Document_FocusChanged(pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc, ByVal Reason As SCBCdrPROJLib.CdrFocusChangeReason, ByVal OldFieldIndex As Long, pNewFieldIndex As Long) 'Set the table column to be invisible, check that the verifier form hasn't been loaded yet. If Reason=CdrBeforeFormLoaded Then 'The Table Setting to use to set table properties. Dim theTableSettings As SCBCdrBrainwareTableEngineLib.SCBCdrTableSettings Dim theAnalysisSettings As Object Project.AllClasses.ItemByName("Invoices").GetFieldAnalysisSettings("Tab le", Project.DefaultLanguage, theAnalysisSettings) 'Get the table settings for the TABLE field. Set theTableSettings = theAnalysisSettings theTableSettings.ColumnVisible(2) = True 'Set the Column visible to True to show, False to hide. End If End Sub </pre>

ExtractClassificationField

In case the ASSA field of the current document class is also the classification field of the project, this method extracts such field.

Note If you use this method on a workdoc having the fields already extracted, the fields indexes might change. However, a new extraction of the whole document restores the original indexes.

Description	Definition	
Syntax	ExtractClassificationField (pWorkdoc As ISCBCdrWorkdoc)	
Parameter	pWorkdoc	Current workdoc

Filename

This property returns the file name of the Project including the directory path.

Description	Definition
Syntax	Filename As String (read only)

ForceValidation

If ForceValidation is set to 'permitted' then the user can overrule the validation by pressing three times on the Return key. If it is set to 'forbidden' then the user cannot change the content of the field disregarding the validation rules.

Description	Definition
Syntax	ForceValidation As CdrForceValidationMode (read/write)

GetVerifierProject

This method returns the *Verifier* Project.

Description	Definition
Syntax	GetVerifierProject (ppVal As Object)
Parameter	ppVal: [out] Verifier Project Object

LastAddressPoolUpdate

This property returns the time when the address pool was updated for the last time.

Description	Definition
Syntax	LastAddressPoolUpdate As Date (read only)

Lock

This method locks the Project for updating.

Description	Definition
Syntax	Lock ()

LogScriptMessage

This method is obsolete and LogScriptMessageEx should be used instead.

LogScriptMessageEx

This method enables the developer to utilize the in-built functionality output messages directly to the core product logs, MMC, or system monitoring notification.

Description	Definition
Syntax	LogScriptMessageEx(ByVal Type As CDRMessageType, ByVal Severity As CDRMessageSeverity, ByVal Message As String)

Description	Definition	
Parameters	Type:	The CdrMessageType option to determine whether the message is classified to either an Information, a Warning, or an Error.
	Severity:	Represents the severity code of the message. Reference CdrMessageSeverity for additional information on options. This option depicts where the message appears (Log, System Monitoring, or as an Email).
	Message:	The message text to display or send.
Example	<pre>Project.LogScriptMessageEx(CDRTypeInfo, CDRSeverityLogFileOnly, "My message")</pre> <p>The script above can be placed in any event; when the event is triggered, a custom script message will be written to the core product log file (H_, D_, V_ or U_ log).</p> <pre>[Info] 30 01:59:33.312 3108 668184k/1428344k 514004k/3520792k 57176k/67252k 238 38/43 My message</pre>	

For more information, see [CdrMessageType](#) and [CdrMessageSeverity](#).

MinClassificationDistance

This property sets or returns the minimal distance of classification results.

Description	Definition
Syntax	MinClassificationDistance As Double (read/write)

MinClassificationWeight

This property sets or returns the minimal classification weight

Description	Definition
Syntax	MinClassificationWeight As Double (read/write)

MinParentClsDistance

This property sets or returns the minimal distance between the classification weight of the parent and the derived DocClasses.

Description	Definition
Syntax	MinParentClsDistance As Double (read/write)

MinParentClsWeight

This property sets or returns the minimal parent classification weight. This value is used as a threshold during parent classification.

Description	Definition
Syntax	<code>MinParentClsWeight As Double (read/write)</code>

MoveDocClass

This method moves a DocClass specified by its Name to a new ParentDocClass specified by NewParentName.

Description	Definition	
Syntax	<code>MoveDocClass (Name As String, NewParentName As String)</code>	
Parameters	Name:	Name of moved DocClass
	NewParentName:	Name of new ParentDocClass

NoUI

This property sets or returns NoUI. If NoUI is set to TRUE, then no login dialog box displays.

Description	Definition
Syntax	<code>NoUI As Boolean (read/write)</code>

Page

This property returns the Cairo Page object of the current Project.

Description	Definition
Syntax	<code>Page As ISCBCroPage (read only)</code>

ParentWindow

This property sets the parent window of the login dialog box.

Description	Definition	
Syntax	<code>ParentWindow As Long (write only)</code>	
Parameters	lhWnd:	[in] Window handle of windows operating system.

PerformScriptCommandRTS

This method allows the developer to restart or stop the Runtime Server through a custom script.

You can use this method to perform a stop on Runtime Server should a third-party, such as SAP, be unavailable.

This method stops the currently running Runtime Server instance executing the script to either stop or restart.

Description	Definition	
Syntax	PerformScriptCommandRTS (CommandID As Long, MessageType As Long, UserCode As Long, MessageDescription As String)	
Parameters	CommandID:	Identifier of the command to execute on the RTS instance. Two commands that are currently supported: Forcing the RTS instance to stop document processing (with the "CommandID" parameter set to "0"). Restarting the RTS instance (with the "CommandID" parameter set to "1").
	MessageType:	The type of message to log when the command executes: "0" for informational message, "1" for warnings and "2" for error messages. Note that error messages are additionally forwarded to MMC administration console of the Runtime Server.
	UserCode:	User error code of the message. This error code can be defined by the developer as any custom error number.
	MessageDescription:	The description of the message to log in the common Runtime Server log file and in the case of error messages on the MMC administration console.
Example	Two examples depicting a stop and a restart of the RTS instance executing project code. <pre> script code stops document processing for the current Runtime Server ' instance and logs specified message as error with error code "777" Project.PerformScriptCommandRTS 0, 2, 777, "RTS is going to be stopped from Custom Script" ' This script code restarts the current Runtime Server instance and logs ' specified message as warning with error code "999" Project.PerformScriptCommandRTS 1, 1, 999, "RTS is going to be restarted from Custom Script" </pre>	

ReleaseAllAdsPools

This method releases the memory used by all ADS Pools loaded in memory by RTS or Verifier.

Use this feature when the project has multiple large ADS pools from different classes that require a lot of memory. If the documents are sorted by class in different batches, only the required pools for a class are loaded in memory when processing the batch. The drawback is a potential decrease in performance because the pools have to be reloaded each time a batch is processed.

Description	Definition
Syntax	<code>Project.ReleaseAllAdsPools()</code>
Example 1	<p>The example below shows the implementation for RTS processing. It is placed in the Initialize event.</p> <pre>Private Sub ScriptModule_Initialize(ByVal ModuleName As String) Project.ReleaseAllAdsPools() End Sub</pre>
Example 2	<p>The example below shows the implementation for Verifier and Web Verifier process. It is placed in the BatchOpen event.</p> <pre>Private Sub ScriptModule_BatchOpen(ByVal Username As String, ByVal BatchDatabaseID As Long, ByVal ExternalGroupID As Long, ByVal ExternalBatchID As String, ByVal TransactionID As Long, ByVal _ WorkflowType As SCBCdrPROJLib.CDRDatabaseWorkflowTypes, ByVal BatchState As Long) Project.ReleaseAllAdsPools() End Sub</pre> <p>The scripts above would provide an entry in the log similar to this:</p> <pre>[Info] 20 14:05:26.812 7488 4820400k/3448008k 5829788k/6631068k 195812k/200160k 543 73/57 Disconnecting ADS Pool for class: Invoices, field: VendorName</pre>

ReportLicensingStatus

This method retrieves either all license counter information, or just the active license counter information.

An active counter license is the document or page limit licensing that is present in the license file.

References the Product Licensing Guide for further details on licensing counters that are present or available in the license file.

The information is saved in the H, D, V or U_log file.

Description	Definition
Syntax	<code>ReportLicensingStatus (ReportActiveLicensingOnly As Boolean, Severity As SCBCdrPROJLib.CDRMessageSeverity)</code>

Description	Definition	
Parameters	ReportActiveLicensingOnly:	A Boolean flag to indicate if all licensing counters should be outputted (False), or if only the license counters active in the license file should be outputted (True).
	Severity:	<p>The location of the utilization output to be sent to. This relates to the defined types shown in CdrMessageSeverity type definition (Log File, Email, or RTS System Monitoring).</p> <p>CDRSeverityLogFileOnly if the license status report is saved in the program's log file only.</p> <p>CDRSeveritySystemMonitoring if the licensing status report is saved in the program's log file and should be forwarded to the System Monitoring service for cumulative system monitoring. This option is applicable when the call is executed from within the Runtime Server only.</p> <p>CDRSeverityEmailNotification if the licensing status report is saved in the program's log file, forwarded to the System Monitoring service for cumulative system monitoring, and also sent by email to the system administrators. This option is applicable when the call is executed from within the Runtime Server program only.</p> <p>An example of a log file output is:</p> <p>Requested current licensing status for license "Internal" with ID 00999-D7CDV811. License updated last time at 2007-11-16 21:02:55. Current licensing period is [2] of 30 days. Project was started at 2007-10-17 15:20:31.</p> <p>License status for [Processed Pages per Day = 500] (active). Current utilization: 0.65%. Units processed: 97 in period of 1 day(s). Units credit: 14903.</p>

Description	Definition
Example	<pre>Code to retrieve licensing utilization information for all licensing counters. Project represents the project library object. Dim theProject As New SCBCdrPROJLib.SCBCdrProject 'The location of the shared license file that is being updated. Dim LicenseShareLocation As String LicenseShareLocation="\\MasterRTS\License\Runtime.lic" 'Activate licensing within the code for project. This enables you to reference the license in the next command. theProject.ActivateLicensing("CustomEXE", LicenseShareLocation) 'Call the License Reporting function, this has several options available theProject.ReportLicensingStatus(False, SCBCdrPROJLib.CDRMessageSeverity.CDRSeverityLogFileOnly) 'Return all license counters theProject.ReportLicensingStatus(False, SCBCdrPROJLib.CDRMessageSeverity.CDRSeverityEmailNotification) 'Return only the active license counters theProject.ReportLicensingStatus(True, SCBCdrPROJLib.CDRMessageSeverity.CDRSeverityEmailNotification)</pre>

For more information, see [ActivateLicensing](#), [GetLicenseValueByName](#), and [GetLicenseValueByID](#).

ShowValidationTemplates

This method displays the validation templates and their settings in a given container.

Description	Definition	
Syntax	ShowValidationTemplates (pContainer As ISCBCdrPPGContainer)	
Parameter	pContainer:	Container used to save the validation templates and their settings.

SLWDifferentResultsAction

This property sets or returns the action to complete if a template classification and supplier extraction has different results.

Description	Definition
Syntax	SLWDifferentResultsAction As CdrSLWDifferentResultsAction (read/write)

SLWSupplierInvalidDifferentClsResults

This property sets or returns if a Supplier Field is made invalid when the template classification and supplier extraction have different results.

Description	Definition
Syntax	SLWSupplierInvalidIfDifferentClsResults As Boolean (read/write)

Unlock

This method unlocks the Project after updating.

Description	Definition
Syntax	Unlock ()

UpdateAddressPool

This method updates the address analysis pool.

Description	Definition
Syntax	UpdateAddressPool ()

ValidationSettingsColl

This property returns a collection of all activated validation engines.

Description	Definition
Syntax	ValidationSettingsColl As ISCBCroCollection (read only)

ValidationTemplates

This returns a collection of all available validation templates.

Description	Definition
Syntax	ValidationTemplates As ISCBCroCollection (read only)

VersionCount

This property returns the number of versions available for a specified file name.

Description	Definition	
Syntax	VersionCount (Filename As String) As Long (read only)	
Parameter	Filename:	Name of the file.

WordSegmentationChars

This property sets or returns a string that contains all characters used for Word segmentation.

Description	Definition
Syntax	WordSegmentationChars As String (read/write)

SCBCdrDocClasses

Description

This Collection contains all defined DocClass objects of the Cedar Project.

Methods and Properties

Collection

This property returns the Collection that is internally used to store the DocClasses.

Description	Definition
Syntax	Collection As ISCBCroCollection (read only)

Count

This property returns the number of items within the Collection.

Description	Definition
Syntax	Count As Long (read only)

Item

This property returns a specified item from the Collection.

Description	Definition	
Syntax	Item (Index As Variant) As SCBCdrDocClass (read only)	
Parameter	Index:	[in] The index can either be a Long value specifying the index within the collection or a String specifying the item by name.

ItemByIndex

This property returns an item from the Collection specified by the index.

Description	Definition	
Syntax	ItemByIndex (Index As Long) As ISCBCdrDocClass (read only)	
Parameter	Index:	[in] Index of the item to retrieve from the Collection, valid range from 1 to Count

ItemByName

This property returns an item from the Collection specified by name.

Description	Definition	
Syntax	ItemByName (Name As String) As ISCBCdrDocClass (read only)	
Parameter	Name:	[in] Name of the item to retrieve from the Collection.

ItemExists

This method returns TRUE if an item with a specified names exists inside the Collection or FALSE is returned.

Description	Definition	
Syntax	ItemExists (Name As String) As Boolean	
Parameter	Name:	[in] Name of item to search for.

ItemIndex

This property returns the index of an item specified by name.

Description	Definition	
Syntax	ItemIndex (Name As String) As Long (read only)	
Parameter	Name:	[in] Name specifying an item in the Collection.

ItemName

This property returns the name of an item specified by the index.

Description	Definition	
Syntax	ItemName (Index As Long) As String (read only)	
Parameter	Index:	[in] Index specifying an item in the Collection, valid range from 1 to Count

Tag

Use this property to store a variant for each item of the Collection.

Description	Definition	
Syntax	Tag (Index As Long) As Variant (read/write)	
Parameter	Index:	Specifies the item index, valid from 1 to Count

SCBCdrDocClass

Description

A Cedar DocClass object represents a single document class within a Cedar project class hierarchy.

Type Definitions

CdrFocusChangeReason

This enumeration defines the reason for the focus change of a Verifier field edit.

Available Types	Description
CdrEnterPressed	Focus changed by pressing Enter
CdrFcrCandidateCopied	Focus changed (refreshed) because a candidate and its location was copied to the field
CdrFcrRefreshed	Focus changed (refreshed) because the selection area and its location was copied to the field
CdrFcrSelectionCopied	Focus changed (refreshed) because the selection area and its location was copied to the field
CdrFcrWordCopied	Focus changed (refreshed) because a word and its location was appended to the field
CdrFormLoaded	Focus changed because of loading form
CdrMouseClicked	Focus changed because of mouse click

Available Types	Description
CdrSelectedOutside	Focus changed because of some selection outside
CdrTableCellSelected	Focus changed because of the selection of a Table cell
CdrTabPressed	Focus changed because of pressing Tab key
CdrUnknownReason	Focus changed because of an unknown reason

CdrVerifierClassifyReason

These are the reasons for classification for the document.

Available Types	Description
CdrChangedReason	The user selected a new class without leaving the classification view.
CdrInitReason	Manual classification view has just been displayed.
CdrValidatedReason	The document class has been changed.

CDRsiModule

This type defines the module in which the smart index definition is used.

Available Types	Description
CDRsiModuleDistiller	Use smart indexing in automatic Field extraction
CDRsiModuleDistVer	Use smart indexing in automatic Field extraction and manual Field validation
CDRsiModuleVerifier	Use smart indexing in manual Field validation

CdrForceValidationMode

This enumeration defines the different options for the ForceValidation.

Available Types	Description
CdrForceValDefault	CdrForceValidationModeDefault: ForceValidationMode inherited
CdrForceValForbidden	CdrForceValidationModeForbidden: ForceValidation (3*return) not allowed
CdrForceValPermitted	CdrForceValidationModePermitted: ForceValidation (3*return) allowed

Methods and Properties

ClassificationField

Use this property to read or write the name of the field that is used for the classification.

Description	Definition
Syntax	<code>ClassificationField As String (read/write)</code>

ClassificationRedirection

This property returns the name of the target DocClass

Description	Definition
Syntax	<code>ClassificationRedirection As String (read/write)</code>

ClassifySettings

This is a collection of chosen classification engines and their settings for this DocClass.

Description	Definition
Syntax	<code>ClassifySettings As ISCBCroCollection (read only)</code>

DerivedDocClasses

This property returns a collection of all DocClasses derived directly from this DocClass

Description	Definition
Syntax	<code>DerivedDocClasses As ISCBCdrDocClasses (read only)</code>

DisplayName

This property sets or returns the display name of the DocClass currently not used. If nothing is inserted here, the DocClass name is used.

Description	Definition
Syntax	<code>DisplayName As String (read/write)</code>

Fields

This property provides access to FieldDefs of a DocClass.

Description	Definition
Syntax	<code>Fields As ISCBCdrFieldDefs (read only)</code>

ForceSubtreeClassification

This property sets or returns if the classification to the subtree of this DocClass is forced.

Description	Definition
Syntax	<code>ForceSubtreeClassification As Boolean (read/write)</code>

ForceValidation

If ForceValidation is set to 'permitted' then the user can overrule the validation by pressing three times on the Return key. If it is set to 'forbidden' then the user cannot change the content of the field disregarding the validation rules.

Description	Definition
Syntax	<code>ForceValidation As CdrForceValidationMode (read/write)</code>

GetFieldAnalysisSettings

This method returns the analysis settings for the document class.

Description	Definition	
Syntax	<code>GetFieldAnalysisSettings (FieldName As String, Language As String, ppAnalysisSettings As ISCBCdrAnalysisSettings)</code>	
Parameters	FieldName:	The name of the field for which the analysis settings are retrieved.
	ppAnalysisSettings:	The name of the analysis settings object that is used in the code to assign the settings to, see script sample.

Description	Definition
<p>Example</p>	<pre>'This script samples shows how to retrieve the analysis settings 'to assign them for example to be used for the associative 'search engine Dim theDocClass As SCBCdrDocClass Dim theAnalysisSettings As ISBCdrAnalysisSettings Dim theSupplierSettings As Object Set theDocClass=Project.AllClasses.ItemByName (pWorkdoc.DocClassName) 'Get the settings for the field VendorName theDocClass.GetFieldAnalysisSettings "VendorName", "German", theAnalysisSettings Set theSupplierSettings = theAnalysisSettings</pre>

Hidden

This property specifies whether the DocClass should be visible in the Project designer.

Description	Definition
Syntax	Hidden As Boolean (read/write)

InitField

This method reinitializes a required field in Workdoc.

Description	Definition	
Syntax	InitField (pWorkdoc As ISBCdrWorkdoc, pField As ISBCdrField)	
Parameters	pWorkdoc:	Current workdoc.
	pField:	Field to be cleared.

ManualTableTrainingMode

This property sets or returns the option for manual Table Extraction training mode.

Description	Definition
Syntax	ManualTableTrainingMode As Boolean (read/write)

Name

This property reads or writes the name of the Document Class

Description	Definition
Syntax	Name As String (read/write)

Page

This property returns the Page object of this DocClass with all defined zones and their OCR settings.

Description	Definition
Syntax	Page As ISBCCroPage (read only)

Parent

This property returns the parent DocClass of the actual DocClass.

Description	Definition
Syntax	Parent As ISBCDrvDocClass (read only)

ShowClassValidationDlg

This method displays the property page validation settings for this document class.

Description	Definition
Syntax	ShowClassValidationDlg (pContainer As ISBCDrvPPGContainer)
Parameter	pContainer: Container in which the property page displays.

ShowFieldValidationDlg

This method displays the property page of the validation settings for the specified field name.

Description	Definition
Syntax	ShowFieldValidationDlg (FieldName As String, pContainer As ISBCDrvPPGContainer)
Parameters	FieldName: Field for which the dialog box is shown.
	pContainer: Container in which the property page displays.

ShowGeneralFieldPPG

This method starts the field settings property page specifying the active tab.

Description	Definition	
Syntax	ShowGeneralFieldPPG (FieldName As String, TabIndexActive As Long, pContainer As ISCBCdrPPGContainer)	
Parameters	FieldName:	Field for which the dialog box is shown.
	TabIndexActive:	Zerobased Index for the tab that displays.
	pContainer:	Container in which the property page displays.

SubtreeClsMinDist

This property returns the minimal distance to the classification weight of the derived DocClasses.

Description	Definition
Syntax	SubtreeClsMinDist As Double (read/write)

SubtreeClsMinWeight

This property sets or returns the minimal classification weight of the derived DocClasses.

Description	Definition
Syntax	SubtreeClsMinWeight As Double (read/write)

UseDerivedValidation

This property sets or returns a Boolean value, when derived validation rules are used.

Description	Definition
Syntax	UseDerivedValidation As Boolean (read/write)

ValidationSettingsColl

This property returns a collection of all activated validation engines

Description	Definition
Syntax	ValidationSettingsColl As ISCBCroCollection (read only)

ValidationTemplateName

This property sets or returns the name of the validation template.

Description	Definition
Syntax	ValidationTemplateName As String (read/write)

ValidClassificationResult

This property sets or returns if this DocClass is a valid classification result or if it is omitted for classification.

Description	Definition
Syntax	ValidClassificationResult As Boolean (read/write)

VisibleInCorrection

This property determines if a project class is available for classification. You can modify this property prior to classification correction for a Verifier by setting the property to TRUE if the class is available for classification correction, or FALSE if the class is unavailable for classification correction.

Dynamic modification of this property is managed through the ScriptModule_VerifierClassify event. Dynamic modification of the class visibility overrides the default Designer class property.

Classification	Document Class	Validation
DocClass Name		
BOLZ Company 1234561		
Display Name		
BOLZ Company 1234561		
<input checked="" type="checkbox"/> This DocClass is a valid Classification Result		
<input checked="" type="checkbox"/> Visible in correction (Verifier)		

Description	Definition
Syntax	VisibleInCorrection As Boolean (read/write)

Description	Definition
<p>Example</p>	<p>The script sample below shows how to dynamically modify the property of classes prior to showing the classification view.</p> <p>The example below hides Invoices, BOLZ and UNICOM classes from verification availability.</p> <pre> Public Function fnShouldHideClass(ByVal strClassNameToCheck As String, pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc) As Boolean Select Case UCase (strClassNameToCheck) Case "BOLZ COMPANY 1234561" fnShouldHideClass = False Case "UNICOM CORPORATION 1234563" fnShouldHideClass = False Case "INVOICES" fnShouldHideClass = False Case Else fnShouldHideClass = True End Select End Function Private Sub ScriptModule_VerifierClassify(pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc, ByVal Reason As SCBCdrPROJLib.CdrVerifierClassifyReason, ClassName As String) Dim i As Long Dim strNextClassName As String If Reason = CdrInitReason Then For i = 1 To Project.AllClasses.Count Step 1 strNextClassName = Project.AllClasses.ItemName(i) Project.AllClasses.ItemByIndex(i).VisibleInCorrection = fnShouldHideClass(strNextClassName, pWorkdoc) Next i End If End Sub </pre>

FillRectangle

This method allows the developer to draw a square on the image (White/Black), which is used to blank out a certain area on the invoice.

By utilizing the FillRectangle method of the SCBCrolImage object, you can perform image redaction.

Description	Definition	
Parameters	Color to use	0: black , 1: white
	Left, Top, Width, Height:	Dimensions of the rectangle

SCBCdrFieldDefs

Description

This Collection contains all defined FieldDef objects of a single DocClass.

Methods and Properties

Collection

This property returns the Collection that is internally used to store the FieldDefs.

Description	Definition
Syntax	Collection As ISCBCroCollection (read only)

Count

This property returns the number of items within the FieldDef Collection.

Description	Definition
Syntax	Count As Long (read only)

Item

This property returns a specified item from the Collection

Description	Definition	
Syntax	Item (Index As Variant) As ISCBCdrFieldDef (read only)	
Parameter	Index:	[in] The index can either be a Long value specifying the index (1 to Count) within the Collection or a String specifying the item by name.

ItemByIndex

This property returns an item from the Collection specified by index.

Description	Definition	
Syntax	<code>ItemByIndex (Index As Long) As ISCBCdrFieldDef (read only)</code>	
Parameter	Index:	[in] Index of the item to retrieve from the Collection.

ItemByName

This property returns an item from the Collection specified by name.

Description	Definition	
Syntax	<code>ItemByName (Name As String) As ISCBCdrFieldDef (read only)</code>	
Parameter	Name:	[in] Name of the item to retrieve from the Collection.

ItemExists

This method returns TRUE if an item with specified name exists inside the Collection or FALSE is returned.

Description	Definition	
Syntax	<code>ItemExists (Name As String) As Boolean</code>	
Parameter	Name:	[in] Name of item to search for.

ItemIndex

This property returns the index of an item specified by name.

Description	Definition	
Syntax	<code>ItemIndex (Name As String) As Long (read only)</code>	
Parameter	Name:	[in] Name specifying an item in the Collection.

ItemNames

This property returns the name of an item specified by index.

Description	Definition	
Syntax	ItemName (Index As Long) As String (read only)	
Parameter	Index:	[in] Index specifying an item in the Collection, valid range from 1 to Count

Tag

Use this property to store a variant for each item of the Collection.

Description	Definition	
Syntax	Tag (Index As Long) As Variant (read/write)	
Parameter	Index:	Specifies the item index, valid range from 1 to Count

SCBCdrFieldDef

Description

A Cedar FieldDef object represents the definition of a single FieldDef inside a Cedar DocClass

Type Definitions

CdrFieldFormat

This type defines the default format of a certain field. (Not yet implemented.)

Available Types	Descriptions
CdrFieldFormatCurrency	CdrFieldFormatCurrency
CdrFieldFormatDate	CdrFieldFormatDate
CdrFieldFormatExtNumber	CdrFieldFormatExtNumber
CdrFieldFormatNone	CdrFieldFormatNone
CdrFieldFormatNumber	CdrFieldFormatNumber

CDRFieldType

This type defines the type of a FieldDef

Available Types	Description
CDRFieldTypeTable	The Field type is Table.
CDRFieldTypeText	The Field type is text, which may be single or multi-line text.

CdrForceValidationMode

This enumeration defines the different options for the ForceValidation.

Available Types	Description
CdrForceValDefault	CdrForceValidationModeDefault: ForceValidationMode inherited
CdrForceValForbidden	CdrForceValidationModeForbidden: ForceValidation (3*return) not allowed
CdrForceValPermitted	CdrForceValidationModePermitted: ForceValidation (3*return) allowed

CdrValFieldType

This enumeration contains different validation types for fields.

Available Types	Description
CdrAmountValidation	Used for amount values or general numeric values.
CdrChkboxValidation	Field as used check box.
CdrCustomValidation	TBD
CdrDateValidation	Used for date values.
CdrListValidation	Used for lists.
CdrTableValidation	Used for tables.
CdrTextValidation	Used for text values, strings.

Methods and Properties

AlwaysValid

This property sets or returns if the content of this FieldDef is always valid.

Description	Definition
Syntax	<code>AlwaysValid As Boolean (read/write)</code>

AnalysisTemplate

This property returns the name of the analysis template if used.

Description	Definition		
Syntax	<code>AnalysisTemplate (Language As String) As String (read only)</code>		
Parameter	<table border="1"> <tr> <td>Language:</td> <td>Language parameter</td> </tr> </table>	Language:	Language parameter
Language:	Language parameter		

AppendListItem

This method adds a new list item and returns a new item index for it.

Description	Definition		
Syntax	<code>AppendListItem (bstrItem As String) As Long</code>		
Parameter	<table border="1"> <tr> <td>bstrItem:</td> <td>String inserted into the list.</td> </tr> </table>	bstrItem:	String inserted into the list.
bstrItem:	String inserted into the list.		

ColumnCount

This property returns the number of Table columns if FieldType is Table.

Description	Definition
Syntax	<code>ColumnCount As Long (read only)</code>

ColumnName

This property returns the name of the Table column specified by an index if FieldType is Table.

Description	Definition	
Syntax	ColumnName (ColumnIndex As Long) As String (read only)	
Parameter	ColumnIndex:	Zero-based index of the Table column

DefaultValidationSettings

This property returns the validation settings with the default language.

Description	Definition	
Syntax	DefaultValidationSettings As ISCBCdrValidationSettings (read only)	

Derived

This property returns TRUE if the FieldDef properties are derived from an upper DocClass.

Description	Definition	
Syntax	Derived As Boolean (read only)	

DisplayName

The DisplayName can be different from the FieldDef name and does not have any restrictions about the used character set while the FieldDef name must be a valid basic name. An application may use the DisplayName instead of the FieldDef name to show a more readable name of the FieldDef

Description	Definition	
Syntax	DisplayName As String (read/write)	

EvalSetting

This property sets or returns the activated evaluation engine and its settings.

Description	Definition	
Syntax	EvalSetting (Language As String) As Object (read/write)	
Parameter	Language:	Language parameter

EvalTemplate

This property returns the name of the evaluation template if used.

Description	Definition	
Syntax	EvalTemplate (Language As String) As String (read only)	
Parameter	Language:	Language of Project

FieldID

This read-only property returns the internally used FieldID.

Description	Definition
Syntax	FieldID As Long (read only)

FieldType

This property sets or returns the type of the FieldDef.

Description	Definition
Syntax	FieldType As CDRFieldType (read/write)

ForceValidation

This property sets or returns the mode for the ForceValidation.

Description	Definition
Syntax	ForceValidation As CdrForceValidationMode (read/write)

ListItem

This property sets or returns a list item string for a given index.

Description	Definition	
Syntax	ListItem (lIndex As Long) As String (read/write)	
Parameter	lIndex:	Zero-based index.

ListItemCount

This property returns the number of strings in the ListItem list.

Description	Definition
Syntax	ListItemCount As Long (read only)
Example	<pre>Dim lngItem As Long For lngItem = Project.AllClasses.ItemByName("Invoice").Fields("Currency").ListItemCount - 1 To 0 Step -1</pre>

MaxLength

This property returns the maximum number of characters permitted for this FieldDef.

Description	Definition
Syntax	MaxLength As Long (read/write)

MinLength

This property sets or returns the minimal number of characters for this FieldDef.

Description	Definition
Syntax	MinLength As Long (read/write)

Name

This property sets or returns the name of the FieldDef.

Description	Definition
Syntax	Name As String (read/write)

NoRejects

This property sets or returns if rejects are permitted.

Description	Definition
Syntax	NoRejects As Boolean (read/write)

OCRConfidence

This property sets or returns the confidence level for OCR. The value must be between 0 and 100.

Description	Definition
Syntax	<code>OCRConfidence As Long (read/write)</code>

RemoveListItem

This method removes a list item by its index.

Description	Definition		
Syntax	<code>RemoveListItem (lIndex As Long)</code>		
Parameter	<table border="1"> <tr> <td>lIndex:</td> <td>Index of entry to be removed from the list.</td> </tr> </table>	lIndex:	Index of entry to be removed from the list.
lIndex:	Index of entry to be removed from the list.		
Example	<code>Project.AllClasses.ItemByName("Invoice").Fields("Currency").RemoveListItem(lngItem)</code>		

SmartIndex

This property contains all definitions about smart indexing.

Description	Definition
Syntax	<code>SmartIndex As ISCBCdrSmartIndex (read/write)</code>

UseDerivedOCRSettings

This property sets or returns if OCR settings of the parent DocClass are used.

Description	Definition
Syntax	<code>UseDerivedOCRSettings As Boolean (read/write)</code>

UseDerivedValidation

This property sets or returns if the derived validation rules are used for validation of this FieldDef.

Description	Definition
Syntax	<code>UseDerivedValidation As Boolean (read/write)</code>

UseMaxLen

This property sets or returns if the maximum number of characters is limited to the value given by MaxLength.

Description	Definition
Syntax	UseMaxLen As Boolean (read/write)

UseMinLen

This property sets or returns if the usage of the minimal number of characters given by the property MinLength is activated.

Description	Definition
Syntax	UseMinLen As Boolean (read/write)

ValidationSettings

This property sets or returns the chosen validation engine and its settings.

Description	Definition
Syntax	ValidationSettings (Language As String) As ISCBCdrValidationSettings (read/write)
Parameter	Language: Defines the language for classification, extraction and validation.

ValidationTemplate

This property returns the name of the validation template.

Description	Definition
Syntax	ValidationTemplate (Language As String) As String (read only)
Parameter	Language: Defines the language for classification, extraction and validation.

ValidationType

This property returns the type of validation.

Description	Definition
Syntax	ValidationType As CdrValFieldType (read only)

VerifierColumnWidth

This property sets or returns the width of the specified column of the Table.

Description	Definition	
Syntax	<code>VerifierColumnWidth (ColumnIndex As Long) As Long (read/write)</code>	
Parameter	ColumnIndex:	Zero-based Index of the Table column

SCBCdrSettings

Description

The Cedar Settings object stores arbitrary strings for usage in script.

Methods and Properties

ActiveClient

This property sets or returns the name of the currently active client.

Description	Definition	
Syntax	<code>ActiveClient As String (read/write)</code>	

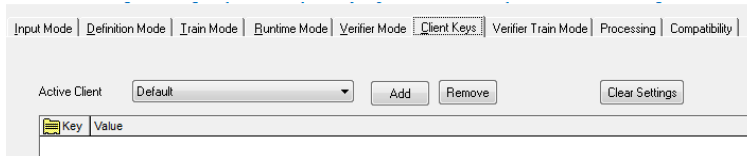
AddClient

This method adds a new client with the specified name to the current Settings object.

Description	Definition	
Syntax	<code>AddClient (newVal As String)</code>	
Parameter	newVal:	Name of new client

AddKey

This method adds a new key specified by its name and its Parent. See the *Perceptive Intelligent Capture Designer Help* for more information.



Description	Definition	
Syntax	<code>AddKey (newVal As String, Parent As String)</code>	
Parameters	newVal:	New key name
	Parent:	Name of the parent key, in case of a new base key use an empty string for the Parent.

Clear

This method clears all clients and keys from the Settings object.

Description	Definition
Syntax	<code>Clear ()</code>

Client

This property returns the name of the specified client

Description	Definition	
Syntax	<code>Client (Index As Long) As String (read only)</code>	
Parameters	Index:	Zero-based client index.

ClientCount

This property returns the number of clients

Description	Definition
Syntax	<code>ClientCount As Long (read only)</code>

GlobalLearnsetPath

This property sets or returns the global Learnset path.

Description	Definition
Syntax	GlobalLearnsetPath As String (read/write)

Key

This property returns the key name specified by index.

Description	Definition
Syntax	Key (Index As Long) As String (read only)
Parameter	Index: Zero-based index of the key.

Key Count

This property returns the number of keys.

Description	Definition
Syntax	KeyCount AS Long (read only)

KeyIcon

This property sets the new value for the specified key or returns the key's value.

Description	Definition
Syntax	KeyIcon (Key As String) As String (read/write)
Parameter	Key: Name of the key.

KeyParent

This property returns the parent name of the specified key index.

Description	Definition
Syntax	KeyParent (Index As Long) As String (read only)
Parameter	Index: Zero-based key index.

MoveKey

This method moves a key specified by its name to the NewParent specified by its name.

Description	Definition	
Syntax	MoveKey (Key As String, NewParent As String)	
Parameter	Key:	Name of key that should be moved
	NewParent:	Name of new parent, empty string in case of moving it as a base key

ProjectFileName

This property sets or returns the file name of the Project.

Description	Definition
Syntax	ProjectFileName As String (read/write)

RemoveClient

This method removes a client specified by its name.

Description	Definition	
Syntax	RemoveClient (ClientName As String)	
Parameter	ClientName:	Name of client that should be removed

RemoveKey

This method removes a key specified by its name.

Description	Definition	
Syntax	RemoveKey (KeyName As String)	
Parameter	KeyName:	Name of key that is removed.

SupervisedLearningDisabled

This property sets or returns the state of supervised learning in *Designer* and local *Verifier* workstations.

Description	Definition
Syntax	SupervisedLearningDisabled As Boolean (read/write)

TopDownEventSequence

This property sets or returns the value of a top-down event sequence.

Description	Definition
Syntax	<code>TopDownEventSequence As Boolean (read/write)</code>

Value

This property returns the value of the specified key.

Description	Definition	
Syntax	<code>Value (Key As String, Parent As String, Client As String) As String (read/write)</code>	
Parameters	Key:	Key name, which is assigned to the value.
	Parent:	Parent name of the key.
	Client:	Name of the client. Can be an empty string. In that case the active client is used.
Example	<pre>MyDBPath = Settings.Value(" DatabaseName ", " ", " ") 'now we can open the database DB.Open(MyDBPath, ...)</pre>	

SCBCdrScriptModule

Description

This is a global object at the project level. All script module events occurred at project level belongs to this object.

Methods and Properties

ModuleName

This property returns the name of the module that initialized ScriptModule.

The full list of values and under what circumstances they are set are detailed below:

- Runtime Server - ModuleName = Server
- Web Verifier Client (v5 and above) - ModuleName = Verifier
- Verifier Thick Client (v3 and above) - ModuleName = Verifier
- Local Verifier Project - ModuleName = LocalVerifier
- Learnset Manager Tool - ModuleName = PlainVerifier
- Designer Runtime mode = Server

- Designer Verifier test mode = Verifier
- Designer Verifier train mode = Verifier
- Designer Normal train mode = Designer
- Designer Definition mode = Designer

Description	Definition
Syntax	ModuleName As String (read only)
Example	<pre> `This example sets the global variable gblVerifierAsServer to true if the Modulename contains VERIFIER Private Sub Document_PreExtract(pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc) If InStr(UCASE(ScriptModule.ModuleName), "VERIFIER") Then gblVerifierAsServer = True Else gblVerifierAsServer = False end if End Sub `This example is a function which returns true if the Modulename contains VERIFIER Public Function fnIsVerifier As Boolean If InStr(UCASE(ScriptModule.ModuleName), "VERIFIER") Then fnIsVerifier = True Else fnIsVerifier = False end if End Function </pre>

ReadZone

Use this method to read a zone on a CrolImage object, the settings of which were previously saved in the ScanJObs' definition.

Description	Definition	
Syntax	ReadZone (Image As ISBCCroImage, ZoneName As String) As String	
Parameters	Image:	[in] SCBCrolImage object
	ZoneName:	[in] Name of Zone that is read

ReadZoneEx

Use this method to read a zone on a Crolmage object, the settings of which were previously saved in the ScanJobs' definition.

Description	Definition	
Syntax	ReadZoneEx (Image As ISBCCroImage, ZoneName As String, Result As ISBCCroWorktext)	
Parameters	Image:	[in] SCBCrolmage object
	ZoneName:	[in] Name of read zone
	Result:	[out] Result of reading returned as SCBCroWorktext object

SCBCdrScriptProject

Description

Methods and Properties

CurrentClient

This property retrieves and sets the "Client" attribute of the batch.

Description	Definition
Syntax	CurrentClient As String (read/write)

GetHostProperties

This method allows the user to retrieve information about the current machine, application, and Perceptive Intelligent Capture user.

Description	Definition
Syntax	GetHostProperties(appType as CDRApplicationName, appSubtype as Long, appInstance as String, appUserName as String, appIP as String, appMachineName as String, appLicensee as String)

Description	Definition	
Parameters	appType	<p>Applicationname represents the calling application by a CDRApplicationName type. The parameter can be read from script.</p> <p>CDRApplicationName:</p> <p>TANDesigner:</p> <ul style="list-style-type: none"> - represents Perceptive Intelligent Capture Designer <p>TANLearnSetManager:</p> <ul style="list-style-type: none"> - represents Perceptive Intelligent Capture Learn Set Manager <p>TANLocalVerifier:</p> <ul style="list-style-type: none"> - represents Perceptive Intelligent Capture Verifier used as local project for SLW <p>TANRuntimeServer:</p> <ul style="list-style-type: none"> - represents Perceptive Intelligent Capture Runtime Service Instance <p>TANUnknown:</p> <ul style="list-style-type: none"> - unknown application <p>TANVerifier:</p> <ul style="list-style-type: none"> - represents Perceptive Intelligent Capture Verifier <p>TANWebVerifier:</p> <ul style="list-style-type: none"> - represents Perceptive Intelligent Capture Web Verifier
	appSubType	Only used for internal purposes
	appInstance	<p>The name of the Perceptive Intelligent Capture Runtime Service Instancename, if ApplicationName is TANRuntimeServer.</p> <p>Not used for other applications.</p>
	appUsername	<p>Login Name of the current Perceptive Intelligent Capture user</p> <p>Perceptive Intelligent Capture user for Designer, Verifier, LSM, Web Verifier</p> <p>Windows user for Runtime Server</p>
	appIP	IP address of the computer
	appMachineName	Machine name that is running the script
	appLicensee	Customer name of the used license file

Description	Definition
<p>Example</p>	<p>The script below calls the GetHostProperties in the initialize event. The method than returns information into variables as to where the script is executed, who is executing it, and what application module is executing it.</p> <pre> Private Sub ScriptModule_Initialize(ByVal ModuleName As String) Dim appInstance As String Dim appSubtype As Long Dim appUserName As String Dim appIP As String Dim appMachineName As String Dim appLicensee As String Dim appType As CDRApplicationName Project.GetHostProperties(appType, appSubtype, appInstance, appUserName, appIP, appMachineName, appLicensee) End Sub </pre>

SCBCdrScriptAccess

Description

Perceptive Intelligent Capture provides a public interface “SCBCdrScriptAccess” for external access to the project and class level custom script pages. The interface can be queried from the main “SCBCdrProject” interface available in Perceptive Intelligent Capture custom script. Using this interface it is possible to retrieve, modify and dump project and class level scripts.

Methods and Properties

DumpAllPages

This method dumps all script pages available in the project as a Unicode text file.

Description	Definition	
Syntax	DumpAllPages(FileName As String)	
Parameters	FileName:	[in] name of the dump file.
Example	<pre> 'Export all script pages to a file (Project and Classes) theScriptAccess.DumpAllPages("Script Export.txt") </pre>	

ExportAllPages

CURRENTLY NOT SUPPORTED. This method exports all available script pages in a reimportable format to the specified folder.

Description	Definition	
Syntax	ExportAllPages(FolderName As String)	
Parameter	FolderName:	[in] name of the folder to save the script pages to.

ExportClassPage

CURRENTLY NOT SUPPORTED. This method exports the specified script page to a script dump file.

Description	Definition	
Syntax	ExportClassPage(FolderName As String, ClassName As String)	
Parameters	FolderName:	[in] name of the folder to save the script page to.
	ClassName:	[in] name of the class to export the script for.

GetPageCode

This method retrieves the project or specified class level script code.

Description	Definition	
Syntax	GetPageCode(ClassName As String, ScriptCode As String)	
Parameters	ClassName:	[in] name of the class.
	ScriptCode:	[out] class script code.

ImportAllPages

CURRENTLY NOT SUPPORTED. This method imports all available script pages using script dumps from the specified folder.

Description	Definition	
Syntax	ImportAllPages(FolderName As String)	
Parameter	FolderName:	[in] name of the folder to load the script pages from.

ImportClassPage

CURRENTLY NOT SUPPORTED. This method imports the specified script page from a script dump file.

Description	Definition	
Syntax	ImportClassPage(FolderName As String, ClassName As String)	
Parameters	FolderName:	[in] name of the folder to load the script page from.
	ClassName:	[in] name of the class to import the script for.

SetPageCode

This method assigns the project or specified class level script code.

Description	Definition	
Syntax	SetPageCode(ClassName As String, ScriptCode As String)	
Parameters	ClassName:	[in] name of the class.
	ScriptCode:	[out] class script code.
Example	<pre>theScriptAccess.SetPageCode(strClassName, " ") 'Set new script code (blank" ")</pre>	

CDRADSLib

SCBCdrSupExSettings

Description

This collection contains the functions for the Associative Search engine.

Methods and Properties

ClearFilterAttributes

This method clears all existing filters of the Multi-column Attribute Search.

Description	Definition
Syntax	ClearFilterAttributes()

Description	Definition
Example	<pre>Dim theSupplierSettings As Object Set theSupplierSettings = FieldAnalysisSettings Dim theAdsSettings As CDRADSLib.SCBCdrSupExSettings Set theAdsSettings = theSupplierSettings theAdsSettings.ClearFilterAttributes</pre>

AddFilterAttributes

This method adds new filters for chosen attribute of the Multi-column Attribute search. Choose attributes from the data source of the Associative Search Engine.

Note The first two attributes are combined as logical OR, and the additional ones that may be added are combined with logical AND.

Description	Definition	
Syntax	AddFilterAttribute("Attribute Name", "Attribute Value")	
Parameters	Attribute Name:	Name of the attribute to be filtered
	Attribute Value	Value of the attribute that is searched for in the datasource

Description	Definition
<p>Example</p>	<p>This example configures the Multi-column Attribute Search for use with the Vendor search button of the Verifier Thick Client. The VendorSearch button in Verifier is related to the Object: General, Process: DialogFunc:</p> <pre> Dim theSupplierSettings As Object Set theSupplierSettings = FieldAnalysisSettings Dim theAdsSettings As CDRADSLib.SCBCdrSupExSettings Set theAdsSettings = theSupplierSettings theAdsSettings.ClearFilterAttributes theAdsSettings.AddFilterAttribute "SupplierName", "VAN" theAdsSettings.AddFilterAttribute "SupplierName", "VAN3" The following example configures the extension for the filtering with RTS in the VendorName (or VendorASSA) object preExtract event: Private Sub VendorName_PreExtract(pField As SCBCdrPROJLib.SCBCdrField, pWorkdoc As SCBCdrPROJLib.SCBCdrWorkdoc) Dim theSupplierSettings As CDRADSLib.SCBCdrSupExSettings Dim theDocClass As SCBCdrDocClass Dim theAnalysisSettings As ISCBCdrAnalysisSettings Dim theObject As Object Set theDocClass=Project.AllClasses.ItemByName(pWorkdoc.DocClassName) theDocClass.GetFieldAnalysisSettings "VendorName", "German", theAnalysisSettings Set theObject = theAnalysisSettings Set theSupplierSettings = theObject theDocClass.GetFieldAnalysisSettings "VendorName", "German", theAnalysisSettings Set theObject = theAnalysisSettings Set theSupplierSettings = theObject theSupplierSettings.ClearFilterAttributes() theSupplierSettings.AddFilterAttribute "SupplierName", "VAN" theSupplierSettings.AddFilterAttribute "SupplierName", "VAN3" End Sub </pre>

Analysis Engines Object Reference

SCBCdrAssociativeDbExtractionSettings

Description

This interface covers all methods and properties that are required for controlling and accessing the new universal format of the ASSA engine's pool.

Type Definitions

CdrAutoUpdateType

This enumeration is used to specify the automatic import property.

Available Types	Description
CdrAUTFile	Automatic import from file for associative search field.
CdrAUTNone	No automatic import for associative search field.
CdrAUTODBC	Automatic import from ODBC source for associative search field.

Method and Properties

AddColumn

This method adds a new column field to the pool.

Description	Definition	
Syntax	AddColumn (ColumnName As String, IsSearchField As Boolean, NewColumnIndex As Long)	
Parameters	ColumnName:	[in] Name of the column field.
	IsSearchField:	[in] Boolean value that has to be set to true when the inserted column field is a search field.
	NewColumnIndex:	[out] Index of the newly created entry in the pool.

AddPhrase

This method appends a new phrase to the list of phrases to use for the address.

Description	Definition
Syntax	AddPhrase (Phrase As String, IsIncludePhrase As Boolean)

Description	Definition	
Parameters	Phrase:	[in] This string variable contains the phrase that is added to the list.
	IsIncludePhrase:	[in] If the value of the Boolean variable is true and the phrase is found, then the resulting address is accepted. If the value of the Boolean variable is false and the phrase is found, then the address is not accepted

ChangeEntry

This method updates or inserts the content of the entry data to the specified column.

Description	Definition	
Syntax	<code>ChangeEntry (ColumnName As String, EntryData As String)</code>	
Parameters	ColumnName:	[in] Name of the column that is changed.
	EntryData:	[in] The content of the specified column is updated with this data.

ClassNameFormat

This property sets or reads the format definition for a document class name.

Description	Definition
Syntax	<code>ClassNameFormat As String (read/write)</code>

ColumnCount

This property returns the number of columns of a currently opened pool.

Description	Definition
Syntax	<code>ColumnCount As Long (read only)</code>

ColumnName

This property returns or sets the name of the column by its index.

Description	Definition	
Syntax	<code>ColumnName (ColumnIndex As Long) As String (read/write)</code>	
Parameter	ColumnIndex:	[in] Index of the column to retrieve [zero-based].

CommitAddEntry

This method takes effect after execution of StartAddEntry and ChangeEntry.

Use this method only in context with the StartUpdate, StartAddEntry, ChangeEntry, CommitAddEntry, and CommitUpdate methods.

Description	Definition	
Syntax	CommitAddEntry (NewIndex As Long)	
Parameter	NewIndex:	[out] Index of new entry.

CommitUpdate

This method closes and save the currently opened pool.

Description	Definition
Syntax	CommitUpdate ()

EnableCandidateEvaluation

This property sets or returns if a candidate evaluation (so called Second Pass) is permitted.

For Enable Candidate Evaluation, the following three options are available.

- Above configured search areas: EvalOverSearchAreas is set to TRUE.
- First page only. EvalFirstPageOnly is set to TRUE.
- All pages of document. Evaluation is performed using the entire text of the document, which is performed if neither of the above restrictions is TRUE. Both are FALSE.

The EvalOverSearchAreas or EvalFirstPageOnly restrictions are mutually excluding, therefore when setting one to TRUE, the other one automatically becomes FALSE.

Note If candidate evaluation (Second Pass) is switched off, then candidates, returned after the first pass, typically have very low confidence.

Description	Definition
Syntax	EnableCandidateEvaluation As Boolean (read/write)

EntryCount

This property returns the number of entries of the pool.

Description	Definition
Syntax	EntryCount As Long (read only)

EvalFirstPageOnly

This property sets or returns if candidate evaluation is performed using the text from first page only.

When EvalFirstPageOnly is set to TRUE, EvalOverSearchAreas becomes FALSE automatically.

Description	Definition
Syntax	EvalFirstPageOnly As Boolean (read/write)

EvalOverSearchAreas

This property sets or returns if the candidate evaluation is processed using only the text from configured search areas.

When EvalOverSearchAreas is set to TRUE, EvalFirstPageOnly becomes FALSE automatically.

Description	Definition
Syntax	EvalOverSearchAreas As Boolean (read/write)

FieldContentsFormat

This property sets or returns the format definition for the representation of the engine.

Description	Definition
Syntax	FieldContentsFormat As String (read/write)

FindLocation

This property sets or returns if address analysis is enabled. If TRUE, the position of the address is found.

Description	Definition
Syntax	FindLocation As Boolean (read/write)

GeneratePool

This method imports the pool from the specified source by the property AutomaticImportMethod.

Description	Definition
Syntax	GeneratePool ()

GeneratePoolFromCsvFile

This method removes the previous pool and generates a new one using CSV file designed in the new format.

Description	Definition
Syntax	GeneratePoolFromCsvFile ()

GeneratePoolFromODBC

This method removes the previous pool and generates a new one using the ODBC source with the parameters set on the property page.

Description	Definition
Syntax	GeneratePoolFromODBC ()

GetClassNameByID

This method returns the formatted document class name for the pool entry specified by its unique ID.

Description	Definition	
Syntax	GetClassNameByID (IDHigh As Long, IDLow As Long, ClassName As String)	
Parameters	IDHigh:	[in] Upper part of 64 bit unique IDs.
	IDLow:	[in] Lower part of 64 bit unique IDs.
	ClassName:	[out] Formatted document class name for the specified entry.

GetEntry

This method returns the content of a field that is specified by its index and the column name.

Description	Definition	
Syntax	GetEntry (Index As Long, FieldName As String) As String	
Parameters	Index:	[in] Index of the entry to be retrieved.
	FieldName:	[in] Name of the column to be retrieved.

GetFormattedValueByID

This method returns the formatted entry representation for the pool entry specified by its unique ID

Description	Definition	
Syntax	GetFormattedValueByID (IDHigh As Long, IDLow As Long, FormattedValue As String)	
Parameters	IDHigh:	[in] Upper part of 64-bit unique ID.
	IDLow:	[in] Lower part of 64-bit unique ID.
	FormattedValue:	[out] Formatted entry representation for the specified entry.

GetIDByIndex

This method returns the unique ID of an entry by index.

Description	Definition	
Syntax	GetIDByIndex (Index As Long, IDHigh As Long, IDLow As Long)	
Parameter	Index:	[in] Zero-based index.
	IDHigh:	[out] Upper part of 64-bit unique ID.
	IDLow:	[out] Lower part of 64-bit unique ID.

GetIndexById

This method returns the index of an entry by its unique ID.

Description	Definition	
Syntax	GetIndexByID (IDHigh As Long, IDLow As Long, Index As Long)	
Parameters	IDHigh:	[in] Upper part of 64-bit unique ID.
	IDLow:	[in] Lower part of 64-bit unique ID.
	Index:	[out] Zero-based index.

GetSearchArea

This method returns an area on the document in which to search.

Description	Definition	
Syntax	GetSearchArea (SearchAreaIndex As Long, Left As Long, Top As Long, Width As Long, Height As Long)	

Description	Definition	
Parameters	SearchAreaIndex:	Index of the area, accepts values from 0 to 5 with following meaning: 0 – First Page Header 1 – First Page Footer 2 – Subsequent Pages Header 3 – Subsequent Pages Footer 4 – Last Page Header 5 – Last Page Footer
	Left:	Distance in % from left border of document.
	Top:	Distance in % from top of document.
	Width:	Width in % of search area.
	Height:	Height in % of search area.

IdentityColumn

This property sets or returns the unique ID of a column name.

Description	Definition
Syntax	<code>IdentityColumn As String (read/write)</code>

ImportFieldNames

This property sets or returns if the column names are taken from the first line of a CSV file.

Description	Definition
Syntax	<code>ImportFieldNames As Boolean (read/write)</code>

ImportFileName

This property sets or returns if column names are taken from the first line of a CSV file.

Description	Definition
Syntax	<code>ImportFileName As String (read/write)</code>

ImportFileNameRelative

This property sets or returns if the name of a CSV file is stored relative to the path of the project file.

Description	Definition
Syntax	ImportFileNameRelative As Boolean (read/write)

IsPhraseIncluded

This property sets or returns if a phrase to find the address is sufficient.

Description	Definition
Syntax	IsPhraseIncluded (PhraseIndex As Long) As Boolean (read/write)
Parameter	PhraseIndex: [in] Index of phrase [zero-based].

IsSearchField

This property sets or returns if a field is used for an associative search.

Description	Definition
Syntax	IsSearchField (ColumnIndex As Long) As Boolean (read/write)
Parameter	ColumnIndex: [in] Index of column [zero-based]

LastImportTimeStamp

This property returns the timestamp for the last import.

Description	Definition
Syntax	LastImportTimeStamp As Date (read only)

MaxCandidates

This property sets or returns the maximum number of results of the associative search engine.

Description	Definition
Syntax	MaxCandidates As Long (read/write)

MinDistance

This property sets or returns the required minimum distance to the next best candidate for a valid result.

Description	Definition
Syntax	MinDistance As Double (read/write)

MinRelevance

This property sets or returns the minimum relevance for search results, default value is 0.0.

Description	Definition
Syntax	MinRelevance As Double (read/write)

MinThreshold

This property sets or returns the required minimum value for a valid engine result.

Description	Definition
Syntax	MinThreshold As Double (read/write)

ODBCName

This property sets or returns the name of the ODBC source.

Description	Definition
Syntax	ODBCName As String (read/write)

Passwords

This property sets or returns the password of the ODBC source.

Description	Definition
Syntax	Password As String (read/write)

Phrase

This property sets or returns the phrase by its index.

Description	Definition
Syntax	Phrase (PhraseIndex As Long) As String (read/write)

PhrasesCount

This property returns the number of phrases used for address analysis.

Description	Definition
Syntax	PhrasesCount As Long (read only)

PoolName

This property sets or returns the name of the associative search pool.

Description	Definition
Syntax	PoolName As String (read/write)

PoolPath

This property sets or returns the name of the path of the associative search pool.

Description	Definition
Syntax	PoolPath As String (read/write)

PoolPathRelative

This property sets or returns if the pool should be saved relative to the path of the project.

Description	Definition
Syntax	PoolPathRelative As Boolean (read/write)

ProjectPath

This property returns the path of the project file.

Description	Definition
Syntax	ProjectPath As String (read only)

RemovePhrase

This method removes a phrase from a list of phrases for address analysis specified by its index.

Description	Definition
Syntax	RemovePhrase (PhraseIndex As Long)
Parameter	PhraseIndex: [in] Index of the phrase that should be deleted [zero-based].

SavePoolInternal

This property sets or returns if a pool should be saved within the project file or as separate files

Description	Definition
Syntax	<code>SavePoolInternal As Boolean (read/write)</code>

SearchAreaActive

This property sets or returns if the corresponding search area is active or not.

Description	Definition		
Syntax	<code>SearchAreaActive(SearchAreaIndex As Long) As Boolean (read/write)</code>		
Parameters	<table border="0"> <tr> <td style="vertical-align: top;">SearchAreaIndex:</td> <td>Index of the area, accepts values from 0 to 5 with following meaning: 0 – First Page Header 1 – First Page Footer 2 – Subsequent Pages Header 3 – Subsequent Pages Footer 4 – Last Page Header 5 – Last Page Footer</td> </tr> </table>	SearchAreaIndex:	Index of the area, accepts values from 0 to 5 with following meaning: 0 – First Page Header 1 – First Page Footer 2 – Subsequent Pages Header 3 – Subsequent Pages Footer 4 – Last Page Header 5 – Last Page Footer
SearchAreaIndex:	Index of the area, accepts values from 0 to 5 with following meaning: 0 – First Page Header 1 – First Page Footer 2 – Subsequent Pages Header 3 – Subsequent Pages Footer 4 – Last Page Header 5 – Last Page Footer		

Separator

This property sets or returns a separator, either a semicolon or comma, that is used for CSV files.

Description	Definition
Syntax	<code>Separator As String (read/write)</code>

SetSearchArea

This method sets the area on the document in which to search.

Description	Definition
Syntax	<code>SetSearchArea (SearchAreaIndex As Long, Left As Long, Top As Long, Width As Long, Height As Long)</code>

Description	Definition	
Parameters	SearchAreaIndex:	Index of the area, accepts values from 0 to 5 with following meaning: 0 – First Page Header 1 – First Page Footer 2 – Subsequent Pages Header 3 – Subsequent Pages Footer 4 – Last Page Header 5 – Last Page Footer
	Left:	Distance in % from left border of document.
	Top:	Distance in % from top of document.
	Width:	Width in % of search area.
	Height:	Height in % of search area.

SQLQuery

This property sets or returns an SQL statement used to import ODBC source.

Description	Definition
Syntax	SQLQuery As String (read/write)

StartAddEntry

This method prepares the insertion of a new entry to the associative search pool.

Description	Definition
Syntax	StartAddEntry ()

StartUpdate

This method generates and opens a new empty pool, or opens an existing pool for the update.

Description	Definition	
Syntax	StartUpdate (RemoveExistingPool As Boolean)	
Parameter	RemoveExistingPool:	[in] When this Boolean variable is set to true, than the old pool is removed, otherwise the existing pool is supposed to be updated by further "AddPhrase" calls. Note that in this case, it should not be required to call "AddColumn" function, because the former column information has to be taken.

Description	Definition	
		Moreover, in case this parameter is true, and the "AddColumn" method is invoked, the "AddColumn" method will report an error because it must be prohibited to modify the existing column.

UserName

This property sets or returns the user name required to login in to the ODBC source.

Description	Definition
Syntax	Username As String (read/write)

VendorTypeColumn

This property sets or returns the column that defines the vendor type. The vendor Type column must contain a value in the 0 to 2 range.

- 0 = No class is created for this vendor through SLW.
- 1 = Allows one document for that vendor to be trained.
- 2 = Allows unlimited training.

Description	Definition
Syntax	VendorTypeColumn As String (read/write)

SCBCdrParaCheckLib

SCBCdrParaCheckAnalysisSettings

Description

This class contains the functions for the Check Analysis engine.

Methods and Properties

AddItemToVocabulary

This method adds a vocabulary entry to the list of possible items.

Description	Definition		
Syntax	AddItemToVocabulary (Name as string, Weight as long, caption as CroParaCheckVocabularyEntriesAddOptions, IsVisible as Boolean)		
Parameters	Name:	Payee Name	Provide the name of the possible payee candidates.

Description	Definition	
	Keyword	Provide a keyword for possible payee candidates. This can be a single word or part of the payee name.
	Weight	Set a value between 0 and 15. Usually, most words have a weight of zero. A weight value of 15 corresponds to a vocabulary entry that appears in approximately 50 % of the images.
	caOption	CroParaCheckVocabularyEntriesAddStatic: add entry to the static vocabulary CroParaCheckVocabularyEntriesAddDynamic: add entry to the dynamic vocabulary
	IsVisible	Defines if the entry is visible in the GUI table in Designer application.
Example	<pre> Dim theFieldCheckAnalysisSettings As Object Set theFieldCheckAnalysisSettings = FieldAnalysisSettings Dim ParaCheckSettings As SCBCdrParaCheckAnalysisSettings Set ParaCheckSettings = theFieldCheckAnalysisSettings ParaCheckSettings.AddItemToVocabulary "PayeeName1", 0, CroParaCheckVocabularyEntriesAddDynamic, True ParaCheckSettings.AddItemToVocabulary "PayeeName2", 15, CroParaCheckVocabularyEntriesAddDynamic, True ParaCheckSettings.AddItemToVocabulary "PayeeName3", 0, CroParaCheckVocabularyEntriesAddDynamic, False ParaCheckSettings.AddItemToVocabulary "PayeeName4", 5, CroParaCheckVocabularyEntriesAddDynamic, True </pre>	

ClearVocabulary

Use this method to remove vocabulary entries.

Description	Definition
Syntax	ClearVocabulary (clOption as CroParaCheckVocabularyEntriesClearOptions)
Parameter	clOption: <ul style="list-style-type: none"> - CroParaCheckVocabularyEntriesClearStaticOnly: Remove all static entries - CroParaCheckVocabularyEntriesClearDynamicOnly: Remove all dynamic entries - CroParaCheckVocabularyClearEntriesAll: Remove all dynamic and static entries

Description	Definition
Example	<pre>Dim theFieldCheckAnalysisSettings As Object Set theFieldCheckAnalysisSettings = FieldAnalysisSettings Dim ParaCheckSettings As SCBCdrParaCheckAnalysisSettings Set ParaCheckSettings = theFieldCheckAnalysisSettings ParaCheckSettings.ClearVocabulary CroParaCheckVocabularyEntriesClearDynamicOnly</pre>

FieldType

This property sets or returns the field type.

Description	Definition
Syntax	FieldType as Long (read/write)

MinDistance

This property sets or returns the minimum distance.

Description	Definition
Syntax	MinDistance as double (read/write)

MinWeight

This property sets or returns the minimum weight.

Description	Definition
Syntax	MinWeight as double (read/write)

PayeeLineRecMode

This property sets or returns the engine recognition mode.

Description	Definition
Syntax	<pre>PayeeLineRecMode as CROParaCheckPayeeRecognitionMode (read/write)</pre> <ul style="list-style-type: none"> • <code>CROParaCheckPayeeRecognitionModeAcc</code> - This is the default mode. In this mode the names in the vocabulary entries reflect the entire Payee Name information • <code>CROParaCheckPayeeRecognitionModeKeyWordSearch</code> - This mode restricts the search to keywords. This enhances the search performance. In this mode, the name entries in the vocabulary entries list contain one single word or a part of the Payee Name.

PayeeVocCoverage

Use this property to define the vocabulary coverage parameter for the payee line. This value expresses the occurrence likelihood of the predefined names on the checks.

Description	Definition
Syntax	<pre>PayeeVocCoverage as long (read/write)</pre>
Parameters	<p><i>LONG</i> Define the vocabulary coverage as a LONG value in the range between 1 and 100. The default value is 35.</p>
Example	<pre>Private Sub Document_PreExtract(pWorkdoc As SCBCdrPROJLib.ISCBCdrWorkdoc) : : Set ParaCheckSettings = Project.AllClasses(pWorkdoc.DocClassName).Fields("Payee").AnalysisSettin g(Project.DefaultLanguage) ParaCheckSettings.PayeeVocCoverage = 85 : : End Sub</pre>

PayeeVocEntries

This property returns the collection of vocabulary entries objects.

Description	Definition
Syntax	<pre>PayeeVocEntries as ISBCCroPayeeVocEntries (read)</pre>

PayeeVocEntries

Description

This is a collection of all vocabulary entry objects contained in the current ParaCheckSettings object.

Methods and Properties

Count

This property returns the number of items within the vocabulary entry collection.

Description	Definition
Syntax	Count As Long (read only)

Item

This read-only property returns a specified item from the collection.

Description	Definition		
Syntax	Item (Index As Variant) As ISCBCroPayeeVocEntry (read only)		
Parameter	<table border="1"> <tr> <td><i>Index:</i></td> <td>The index can either be a long value specifying the index within the collection, valid range from 1 to Count, or a string specifying the item by name.</td> </tr> </table>	<i>Index:</i>	The index can either be a long value specifying the index within the collection, valid range from 1 to Count, or a string specifying the item by name.
<i>Index:</i>	The index can either be a long value specifying the index within the collection, valid range from 1 to Count, or a string specifying the item by name.		

ItemByIndex

This property returns an item from the collection specified by the index.

Description	Definition		
Syntax	ItemByIndex (Index As Long) As ISCBCroPayeeVocEntry (read only)		
Parameter	<table border="1"> <tr> <td><i>Index:</i></td> <td>Index of the item to retrieve from the Collection, valid range from 1 to Count</td> </tr> </table>	<i>Index:</i>	Index of the item to retrieve from the Collection, valid range from 1 to Count
<i>Index:</i>	Index of the item to retrieve from the Collection, valid range from 1 to Count		

ItemByName

This property returns the specified item from the collection.

Description	Definition		
Syntax	ItemByName (Name As String) As ISCBCroPayeeVocEntry (read only)		
Parameter	<table border="1"> <tr> <td><i>Name:</i></td> <td>Name of the item to retrieve from the collection.</td> </tr> </table>	<i>Name:</i>	Name of the item to retrieve from the collection.
<i>Name:</i>	Name of the item to retrieve from the collection.		

ItemExists

This method returns TRUE if an item with the specified name exists inside the collection, or FALSE if not.

Description	Definition	
Syntax	ItemExists (Name As String) As Boolean	
Parameter	<i>Name:</i>	Name of item for which to search.

ItemIndex

Returns the index of an item specified by name.

Description	Definition	
Syntax	ItemIndex (Name As String) As Long (read only)	
Parameter	<i>Name:</i>	Name specifying an item in the collection.

ItemName

Returns the name of an item specified by index.

Description	Definition	
Syntax	ItemName (Index As Long) As String (read only)	
Parameter	<i>Index:</i>	Index specifying an item in the collection, valid range from 1 to Count.

MoveItem

This method moves an item specified by OldIndex from OldIndex to NewIndex.

Description	Definition	
Syntax	MoveItem (OldIndex As Long, NewIndex As Long)	
Parameters	<i>OldIndex:</i>	Index of item to move valid range from 1 to Count.
	<i>NewIndex:</i>	New index of the item after the move has occurred, valid range from 1 to Count.

Remove

This method removes the specified item from the collection and releases the reference count to this item.

Description	Definition	
Syntax	Remove (ItemName As String)	
Parameter	<i>ItemName:</i>	Name of item to remove.

Tag

This property stores a variant for each item of the collection.

Description	Definition	
Syntax	Tag (Index As Long) As Variant (read/write)	
Parameter	<i>Index:</i>	Specifies the item index, valid range from 1 to Count.

SCBCdrFormatEngine

This class provides methods and properties for the Format Analysis engine.

CdrFormatSettings

This table contains the properties for the Format Analysis engine.

Property	Description
MaxWordGap	Corresponds to the "Max. Compare Distance" property
MaxWordCount	Corresponds to the "Max. Wordcount" property
MaxWordLen	Maximum word length
FormatCount	Number of format strings
FormatString	The string to add to the Format Strings list.
Prefix	Corresponds to the "Prefix" property
Suffix	Corresponds to the "Suffix" property
AnalysisMethod	CdrAnalysisString corresponds to the "Find Format" property CdrAnalysisDesignator corresponds to the "Find Designator" property

Property	Description
DesignatorType	CdrDesignatorNextWord CdrDesignatorNextBlock CdrDesignatorNextParagraph CdrDesignatorThisBlock CdrDesignatorPrevWord CdrDesignatorWordBelow CdrDesignatorWordAbove CdrDesignatorNextLine CdrDesignatorEndOfLine
CompareType	See CdrCompareType
MaxDistance	Corresponds to the “Max. Compare Distance” property
CaseSensitive	Corresponds to the “Compare case sensitive” property
KeepSpaces	Corresponds to the “Keep spaces between connected words” property
UseRegions	Corresponds to the “Restrict engine to region” property
UseFirstPage	Corresponds to the “First Page” property
UseSubseqPage	Corresponds to the “Subseq” property
UseLastPage	Corresponds to the “Last Page” property
TopFirst	Corresponds to the “Top” property for the first page
BottomFirst	Corresponds to the “Bottom” property for the first page
LeftFirst	Corresponds to the “Left” property for the first page
RightFirst	Corresponds to the “Right” property for the first page
TopSubseq	Corresponds to the “Top” property for the subsequent page
BottomSubseq	Corresponds to the “Bottom” property for the subsequent page
LeftSubseq	Corresponds to the “Left” property for the subsequent page
RightSubseq	Corresponds to the “Right” property for the subsequent page
TopLast	Corresponds to the “Top” property for the last page
BottomLast	Corresponds to the “Bottom” property for the last page

Property	Description
LeftLast	Corresponds to the "Left" property for the last page
RightLast	Corresponds to the "Right" property for the last page
IgnoreCharacters	Corresponds to the "Ignore Characters" property

Methods and Properties

SrchFlag

Bit flag property to set or return the string construction rules for a **single** Format String.

Bit 1 contains the "Compare case sensitive" option. Use this option to make the candidate search case-sensitive.

Bit 2 contains the "Keep spaces between connected words" option. Use this option to keep the spaces in the format string.

Use the bit-wise operators `Or` and `And` `Not` to set or clear the options.

Note The options "Compare case sensitive" and "Keep spaces between connected words" on the General tab of the "Format Analysis Engine" settings in Designer refer to **all** Format Strings.

For more information, see "Rules for string construction from words" in the *Perceptive Intelligent Capture Designer Help*.

Description	Definition		
Syntax	<code>SrchFlag (index As Long) As Long (read/write)</code>		
Parameter	<table border="1"> <tr> <td>Index:</td> <td>The index parameter has a valid range from 0 to FormatCount-1.</td> </tr> </table>	Index:	The index parameter has a valid range from 0 to FormatCount-1.
Index:	The index parameter has a valid range from 0 to FormatCount-1.		
Example	<p>This example shows how to set "Compare case sensitive" and "Keep spaces between connected words" options for a specific Format String of the field MyField, and how to check if an option is active.</p> <pre>Private Sub MyField_PreExtract _ (pField As SCBCdrPROJLib.ISCBCdrField, _ pWorkdoc As SCBCdrPROJLib.ISCBCdrWorkdoc) Dim DocClass As SCBCdrDocClass Dim theAnalysisSettings As SCBCdrPROJLib.ISCBCdrAnalysisSettings Dim theSettings As Object Dim theFormatSettings As SCBCdrFormLib.SCBCdrFormatSettings Dim nFlag3 As Long 'Get current DocClass</pre>		

Description	Definition
	<pre> Set DocClass=Project.AllClasses.ItemByName(pWorkdoc.DocClassName) 'Get the settings for the field 'MyField' DocClass.GetFieldAnalysisSettings ("MyField","German", _ theAnalysisSettings) 'Convert them to the SCBCdrFormatSettings Set theSettings = theAnalysisSettings Set theFormatSettings = theSettings 'Set the "Case Sensitive" option (bit 1) for Format string 3 'Get the current settings for the Format String nFlag3 = theFormatSettings.SrchFlag(3) 'Set bit 1. To clear the option, use "nFlag3 And (Not 1)" theFormatSettings.SrchFlag(3) = nFlag3 Or 1 'Set the "Keep spaces..." option (bit 2) for Format string 3 'Get the current settings for the Format String nFlag3 = theFormatSettings.SrchFlag(3) 'Set bit 2. To clear the option, use "nFlag3 And (Not 2)" theFormatSettings.SrchFlag(3) = nFlag3 Or 2 'If "Keep spaces..." (bit 2) is enabled for Format String 3 'write a message in the Log If (theFormatSettings.SrchFlag(3) And 2) Then Project.LogScriptMessageEx (CDRTypeInfo, _ CDRSeverityLogFileOnly, _ "Keep Spaces option is active for Format String 3") End If End Sub </pre>

TestString

Use this method to test a particular search string (Format String) against an arbitrary text, using the settings assigned to that specific Format String.

The method returns the distance value. The distance calculates as follows:
 $(\text{StringLength} - \text{MatchedSubstrLength}) / \text{StringLength}$.

0.0 means that the exact search string was found.

1.0 means that not even a partial match was found.

Note The considered MatchedSubstrLength is the one with the maximum length in the Format String expression.

Description	Definition	
Syntax	<code>TestString(ByVal bstrText As String, ByVal nFormatIndex As Long, pFormatSettings As SCBCdrFormatSettings) As Single</code>	
Parameters	bstrText	The string to be tested against the Format String.
	nFormatIndex	Zero-based index of the Format String list.
	pFormatSettings	The settings of the specific Format String. See CdrFormatSettings for more information.
Example	<p>This example writes the confidence of the test string "1234567890" against the Format String #[3-5], which is the index 3 of the field MyField, into the log file.</p> <pre> Private Sub MyField_PreExtract _ (pField As SCBCdrPROJLib.ISCBCdrField, _ pWorkdoc As SCBCdrPROJLib.ISCBCdrWorkdoc) Dim DocClass As SCBCdrDocClass Dim theAnalysisSettings As SCBCdrPROJLib.ISCBCdrAnalysisSettings Dim theSettings As Object Dim theFormatSettings As SCBCdrFormLib.SCBCdrFormatSettings Dim AE As Object Dim theFormatEngine As SCBCdrFormLib.SCBCdrFormatEngine Dim TestStr As String Dim fDist As Single 'Get current DocClass Set DocClass=Project.AllClasses.ItemByName(pWorkdoc.DocClassName) 'Get the settings for the field 'MyField' DocClass.GetFieldAnalysisSettings("MyField","German", _ theAnalysisSettings) 'Convert them to the SCBCdrFormatSettings Set theSettings = theAnalysisSettings Set theFormatSettings = theSettings 'Get SCBCdrFormatEngine from the Project Set AE=Project.GetAnalysisEngineByName("Format Analysis Engine") Set theFormatEngine = AE </pre>	

Description	Definition
	<pre> 'Test a string against Format String index 2 of field MyField, 'using the current options for that specific search string TestStr = "1234567890" 'The Format String index 2 is #[3-5] (Simple Expression) fDist = theFormatEngine.TestString(TestStr, 2, theFormatSettings) Project.LogScriptMessageEx(CDRTypeInfo, CDRSeverityLogFileOnly, _ "Distance from the String 1234567890 using #[3-5] is: " _ & CStr(fDist)) 'Expected value is 0.5 End Sub </pre>

StringComp Object Reference (SCBCdrSTRCOMPLib)

SCBCdrStringComp

Description

This component provides several implementations of string compare algorithms.

Type Definitions

CdrCompareType

This table contains a list of string compare algorithms.

Available Types	Description
CdrTypeLevenShtein	Levenshtein algorithm
CdrTypeRegularExpression	Regular expression
CdrTypeSimpleExpression	Simple expression
CdrTypeStringComp	Exact string compare
CdrTypeTrigram	Trigram algorithm

Methods and Properties

CaseSensitive

This property controls if the compare algorithm should work case-sensitive.

Description	Definition
Syntax	<code>CaseSensitive As Boolean (read/write)</code>

CompTypes

This property selects the compare algorithm used for the next call of Distance.

Description	Definition
Syntax	<code>CompType As CdrCompareType (read/write)</code>

Distance

This method performs the selected string compare algorithm. You must first initialize the search expression and the compare method. The return value is the distance between the search expression and the string parameter, which is between 0.0 and 1.0. A distance of 0.0 means that the search expression matches the string parameter exactly and a distance of 1.0 means that there is no match at all. Most algorithms can also return a value between 0.0 and 1.0, which provides the possibility to compare strings in a fault tolerant way.

Description	Definition	
Syntax	<code>Distance (String As String, Distance As Double)</code>	
Parameters	String:	[in] Specifies the string that should be compared with the search expression.
	Distance:	[out] Contains the distance of the compare operation that is between 0.0 and 1.0.

LevDeletions

This property returns the count of deletions calculated by the last Distance function.

Description	Definition
Syntax	<code>LevDeletions As Single (read only)</code>

LevInsertions

This property returns the count of insertions calculated by the last Distance function.

Description	Definition
Syntax	LevInsertions As Single (read only)

LevRejects

This property returns the count of rejects calculated by the last Distance function.

Description	Definition
Syntax	LevRejects As Single (read only)

LevReplactments

This property returns the count of replacements calculated by the last Distance function.

Description	Definition
Syntax	LevReplacements As Single (read only)

LevSame

This property returns the count of equal characters calculated by the last Distance function.

Description	Definition
Syntax	LevSame As Single (read only)

LevTraceMatrix

This property returns the Levenshtein trace matrix calculated by the last Distance function.

Description	Definition
Syntax	LevTraceMatrix As String (read only)

LevTraceResult

This property returns the Levenshtein trace result calculated by the last Distance function.

Description	Definition
Syntax	LevTraceResult As String (read only)

MatchEndPosition

This property returns the matching end position calculated by the last Distance function.

Description	Definition
Syntax	MatchEndPosition As Single (read only)

MatchStartPosition

This property returns the matching start position calculated by the last Distance function.

Description	Definition
Syntax	MatchStartPosition As Single (read only)

SearchExpression

This property contains the search expression that should be used for the next compare operation.

Description	Definition
Syntax	SearchExpression As String (read/write)

ValidateSearchExpression

This method performs a syntax check for the specified compare method and search expression.

Description	Definition	
Syntax	ValidateSearchExpression (Type As CdrCompareType, SearchExpression As String) As Boolean	
Parameters	Type:	Compare method to use for validation.
	SearchExpression:	Search expression to validate.

SCBCdrEmailProperties

Description

When importing a MSG file into a Workdoc, the most important properties of the email are stored in the Workdoc and available in the custom script via the “**ISBCdrEmailProperties**” interface that can be queried from the SCBCdrWorkdoc interface.

The following properties are supported.

- Email subject
- List of email senders

- List of email recipients
- List of carbon copy recipients
- Date and time the email was sent
- Date and time the email was received at
- Priority the email was sent with
- Unique message identifier.

Properties

CdrMessageSeverity

This type defines the different message severities.

Available Types	Description
CDRSeverityLogFileOnly	Store the message to the application log file only.
CDRSeveritySystemMonitoring	Store the message in the log file and forward it to the host instance's MMC console and to the System Monitoring service of the Runtime Server. This option is applicable when the call is executed from within the Runtime Server application only.
CDRSeverityEmailNotification	Store the message in the log file and forward it to the MMC console / System Monitoring view and send as an email to the system administrators via System Monitoring service of Runtime Server. This option is applicable when the call is executed from within the Runtime Server application only.

Disable moving downloaded messages to the deleted items folder

Use this feature for testing purposes only to keep your messages in the Inbox to download the same documents set on each Runtime Server import iteration.

To disable moving downloaded messages to the deleted items folder on the Exchange server, complete the following steps.

1. Launch the Windows registry editor.
2. Complete one of the following substeps.
 - For a 32-bit machine, navigate to [HKEY_LOCAL_MACHINE\SOFTWARE\Lexmark\Cedar].
 - For a 64-bit machine, navigate to [HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Lexmark\Cedar].
3. In the right pane, right-click and then click **New > DWORD (32-bit) Value**.
4. In the **Name** field, type **LeaveDownloadedMessagesInInbox** and then click **OK**.
5. Right-click the **LeaveDownloadedMessagesInInbox** key and then click **Modify**.
6. In the **Edit DWORD (32-bit) Value** dialog box, in the **Value data** field, to disable moving the downloaded messages, type **1** and then click **OK**.

- 7. Optional. To enable moving the downloaded messages, delete the **LeaveDownloadedMessagesInInbox** key.

SCBCdrLicenseInfoAccess

Description

The Licensing Information Access object allows direct retrieval to the active licensing object. The Developer is able to directly query any licensing component in a custom script.

Methods

GetLicenseCounterByID

This method returns the license counter information for any given active or inactive license counter.

An active counter is one that is specifically identified in the license file and is enforced by the licensing mechanism.

Note This method is not available for engine-level counters. Use GetLicenseCounterByName instead.

Description	Definition	
Syntax	GetLicenseCounterByID(CounterID As SCBCdrPROJLib.CDRLicenseCounter, Count As Long, Active As Boolean)	
Parameters	CounterID:	The ID of the counter from which the value is retrieved. The ID is determined by the CdrLicenseCounter project data type.
	Count:	The returned utilization value from the licensing mechanism. This stores the value of usage.
	Active:	Identifies if the license counter should be active, or specified in the license file.

Description	Definition
<p>Example</p>	<p>An example to retrieve the OCRred count of documents in script.</p> <pre> Dim theLicensingInterface2 As SCBCdrPROJLib.SCBCdrLicenseInfoAccessDim theObject2 As Object Dim vValue2 As Long Dim vValue3 As Variant Dim LicInfoMsg2 As String vValue2=0 vValue3=0 Project.ActivateLicensing "Designer", "C:\Program Files (x86)\Lexmark\Components\Cairo" Set theObject2 = Project Set theLicensingInterface2 = theObject2 ' theLicensingInterface2.GetLicenseCounterByID(TLCPeriodPagesOCRred, vValue2, False) ' theLicensingInterface2.GetLicenseCounterByID(TLCTotalPagesOCRred, vValue3, False) ' theLicensingInterface2.GetLicenseCounterByID(TLCFineReaderRemainingUnits , vValue2, True) theLicensingInterface2.GetLicenseCounterByName ("Overall OCRred Pages", vValue2, True) LicInfoMsg2 = "OCRred count - " & CStr(vValue2) MsgBox(LicInfoMsg2, vbOkOnly, "Get License Count By ID") </pre>

For more information, see `CdrLicenseCounter`, `CdrLicenseFeatureName`, `GetLicenseCounterByName`, `GetLicenseValueByID`, `GetLicenseValueByName`, and `ActivateLicensing`.

GetLicenseCounterByName

This method returns the license counter information for any given active or inactive license counter.

An active license counter is one that is specifically identified in the license file and is enforced by the licensing mechanism.

Description	Definition	
<p>Syntax</p>	<pre>GetLicenseCounterByName(CounterName As String, Count As Long, Active As Boolean)</pre>	
<p>Parameters</p>	<p>CounterName:</p>	<p>The Name of the counter from which the value is retrieved. The Name is the same as shown in the license file.</p>
	<p>Count:</p>	<p>The returned utilization value from the licensing mechanism. This stores the value of usage.</p>
	<p>Active:</p>	<p>Identifies if the license counter should be active, or specified in the license file.</p>
<p>Example</p>	<p>An example to retrieve the OCRred count of documents in script.</p> <pre>Dim theLicensingInterface As SCBCdrPROJLib.SCBCdrLicenseInfoAccess</pre>	

Description	Definition
	<pre> Dim theObject As Object Dim vValue1 As Variant Dim LicInfoMsg As String Project.ActivateLicensing "Designer", "" Set theObject = Project Set theLicensingInterface = theObject theLicensingInterface.GetLicenseCounterByName("OCRed Pages per Day", vValue1, True) LicInfoMsg = "OCRed count - " & CStr(vValue1) </pre>

For more information, see [CdrLicenseCounter](#), [CdrLicenseFeatureName](#), [GetLicenseCounterByID](#), [GetLicenseValueByID](#), [GetLicenseValueByName](#), and [ActivateLicensing](#).

GetLicenseValueById

This method returns the license counter information for any given item in the license file.

Note This method is not available for engine-level counters. Use [GetLicenseValueByName](#) instead.

Description	Definition	
Syntax	<pre> GetLicenseValueByID(PropertyID As SCBCdrPROJLib.CDRLicenseFeatureName, Value As Variant) </pre>	
Parameters	PropertyID:	Depicts the item for which to retrieve the values. Various options can be found in CdrLicenseFeatureName .
	Value:	The returned value from the licensing mechanism. The data type varies depending on the item being returned.
Example	<p>An example to retrieve the Email Importing flag in the license file.</p> <pre> Dim theLicensingInterface As SCBCdrPROJLib.SCBCdrLicenseInfoAccess Dim theObject As Object Dim vValue1 As Variant Dim LicInfoMsg As String Project.ActivateLicensing "Designer", "" Set theObject = Project </pre>	

Description	Definition
	<pre> Set theLicensingInterface = theObject theLicensingInterface.GetLicenseValueByID(CDRLfnEMailsImporting, vValue1) LicInfoMsg = "Email Importing - " & CStr(vValue1) MsgBox(LicInfoMsg, vbOkOnly,"Get License Value By ID") </pre>

For more information, see `CdrLicenseCounter`, `CdrLicenseFeatureName`, `GetLicenseCounterByID`, `GetLicenseCounterByName`, `GetLicenseValueByName`, and `ActivateLicensing`.

GetLicenseValueByName

This method returns the license counter information for any given item in the license file.

Description	Definition	
Syntax	<pre>GetLicenseValueByName(PropertyName As String, Value As Variant)</pre>	
Parameters	PropertyName:	Depicts the name on which to retrieve values. Various options can be found in the license file. The text to be entered for this parameter should be the exact same text as appears in the license file.
	Value:	The returned value from the licensing mechanism. The data type varies depending on the item being returned.
Example	<p>An example to retrieve the Email Importing flag in the license file.</p> <pre> Dim theLicensingInterface As SCBCdrPROJLib.SCBCdrLicenseInfoAccess Dim theObject As Object Dim vValue1 As Variant Dim LicInfoMsg As String Project.ActivateLicensing "Designer", "" Set theObject = Project Set theLicensingInterface = theObject theLicensingInterface.GetLicenseValueByName("Serial", vValue1) LicInfoMsg = "Primary Dongle Serial Number - " & CStr(vValue1) </pre>	

Description	Definition
	<code>MsgBox(LicInfoMsg, vbOkOnly, "Get License Value By Name")</code>

For more information, see `CdrLicenseCounter`, `CdrLicenseFeatureName`, `GetLicenseCounterByID`, `GetLicenseCounterByName`, `GetLicenseValueByName`, and `ActivateLicensing`.

Cedar Verifier Component Library

SCBCdrVerificationForm

Description

This interface is used to set properties specific for verification form object, as well as to set default properties for embedded elements, like verification fields, labels, tables, buttons, and so on.

For the Web Verifier, use this method in the `VerifiedFormatLoad` event.

Methods and Properties

DefaultLabelFont

This property sets or returns the default font for all label elements available on this verification form.

Description	Definition
Syntax	<code>DefaultLabelFont As StdFont</code>

DefaultLabelFontColor

This property sets or returns the default color for all label elements available on this verification form.

Description	Definition
Syntax	<code>DefaultLabelFontColor As OLE_COLOR</code>
Example	<pre>Dim clrDefaultColor As OLE_COLOR clrDefaultColor = -1 the Form.VerificationLabels.ItemByIndex(1NextLabelIndex).FontColor = clrDefaultColor</pre>

DefaultLabelBackgroundColor

This property sets or returns the default background color for all label elements available on this verification form.

Description	Definition
Syntax	<code>DefaultLabelBackgroundColor As OLE_COLOR</code>

DefaultFieldFont

This property sets or returns the default font for all verification field elements available on this verification form.

Description	Definition
Syntax	DefaultFieldFont As StdFont

DefaultFieldFontColor

This property sets or returns the default color for all verification field elements available on this verification form.

Description	Definition
Syntax	DefaultFieldFontColor As OLE_COLOR

DefaultElementBackgroundColorValid

This property sets or returns the default color for all valid (valid in terms of validation status) field elements available on this verification form.

Description	Definition
Syntax	DefaultElementBackgroundColorValid As OLE_COLOR

DefaultElementBackgroundColorInvalid

This property sets or returns the default color for all invalid (invalid in terms of validation status) field elements available on this verification form.

Description	Definition
Syntax	DefaultElementBackgroundColorInvalid As OLE_COLOR

FormBackgroundColor

This property sets or returns the background color for the form.

Description	Definition
Syntax	FormBackgroundColor As OLE_COLOR

FormBackgroundColorDI

This property sets or returns the background color for the Direct Input control on the form, i.e. for the area around the Direct Input field.

Description	Definition
Syntax	FormBackgroundColorDI As OLE_COLOR

SCBCdrVerificationField

Description

This interface is used to identify verification properties specific for header fields' validation elements, like drop down lists, check-boxes, and normal edit fields.

Note To get the OLE_COLOR or StdFont object for the properties below, add **OLE Automation** as a reference.

Type Definitions

CdrVerifiedFieldType

This is the Verifier Field type. This type interface is a member of the Cedar Verifier Project library.

Available Type	Description
CDRVerifierFieldTypeCheckbox	Check box field type
CDRVerifierFieldTypeCombobox	Combo box field type
CDRVerifierFieldTypeTableCheckBoxCell	Table check box cell field type
CDRVerifierFieldTypeTextMultiline	Multiline Text field type
CDRVerifierFieldTypeTextSingleline	Single Line Text field type

Methods and Properties

AutoCompletionEnabled

This property enables or disables Auto Completion for a verification field.

Description	Definition
Attribute	Read/Write

Description	Definition
<p>Example</p>	<p>The example below turns Auto Completion on for the Invoice Number field.</p> <pre> Dim theVerificationProject As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationProject Dim theVerificationForm As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationForm ' Request the main form Project.GetVerifierProject theVerificationProject Set theVerificationForm = theVerificationProject.AllVerificationForms.ItemByName("Invoices") theVerificationForm.VerificationFields.ItemByName("Field_InvoiceNo").AutoCompletionEnabled = True </pre>

BackgroundColorInvalid

This property sets the color for the verification field to display to the user when the field required manual verification. When the field is Invalid in Verifier, the color that is set displays to the user. By default, the invalid background color of the field is red.

Description	Definition
<p>Attribute</p>	<p>Read/Write</p>
<p>Example</p>	<p>The example below turns the color for Invoice Number field to gray if it is Invalid.</p> <pre> Dim theVerificationProject As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationProject Dim theVerificationForm As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationForm ' Request the main form Project.GetVerifierProject theVerificationProject Set theVerificationForm = theVerificationProject.AllVerificationForms.ItemByName("Invoices") theVerificationForm.VerificationFields.ItemByName("InvoiceNo").BackgroundColorInvalid = RGB(192, 129, 129) </pre>

For more information, see [BackgroundColorValid](#).

BackgroundColorValid

This property sets the color for the verification field to display to the user when the field does not require manual verification. When the field is Valid in Verifier, the color that is set displays to the user. By default, the valid background color of the field is green.

Description	Definition
<p>Attribute</p>	<p>Read/Write</p>

Description	Definition
<p>Example</p>	<p>The example below turns the color for Invoice Number field to gray if it is Valid.</p> <pre> Dim theVerificationProject As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationProject Dim theVerificationForm As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationForm ' Request the main form Project.GetVerifierProject theVerificationProject Set theVerificationForm = theVerificationProject.AllVerificationForms.ItemByName("Invoices") theVerificationForm.VerificationFields.ItemByName("Field_InvoiceNo").Ba ckgroundColorValid = RGB (192, 129, 129) </pre>

For more information, see `BackgroundColorInvalid`.

Font

This property sets the Font for the content of the verification field.

Description	Definition
<p>Attribute</p>	<p>Read/Write</p>
<p>Example</p>	<p>The example below sets the Font for Invoice Number field.</p> <pre> Dim theVerificationProject As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationProject Dim theVerificationForm As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationForm Dim DefaultFieldFont As New StdFont DefaultFieldFont.Bold = False 'Set Font attributes ' Request the main form Project.GetVerifierProject theVerificationProject Set theVerificationForm = theVerificationProject.AllVerificationForms.ItemByName("Invoices") theVerificationForm.VerificationFields.ItemByName("Field_InvoiceNo").Fo nt = DefaultFieldFont </pre>

For more information, see `FontColor`.

FontColor

This property sets the Font Color for the content of the verification field.

Description	Definition
Attribute	Read/Write
Example	<p>The example below sets the FontColor for Invoice Number field to gray.</p> <pre>Dim theVerificationProject As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationProject Dim theVerificationForm As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationForm ' Request the main form Project.GetVerifierProject theVerificationProject Set theVerificationForm = theVerificationProject.AllVerificationForms.ItemByName("Invoices") theVerificationForm.VerificationFields.ItemByName("Field_InvoiceNo").FontColor = RGB (192, 129, 129)</pre>

For more information, see [Font](#).

Invisible

This property determines if the field is visible or hidden from the Verifier or WebVerifier form. The developer uses script options to hide or display the field from the verifier user. For the WebVerifier, this property can only be used in the VerifierFormload event.

Description	Definition
Attribute	Read/Write
Example	<p>The example below hides the Invoice Number field from the verifier user.</p> <pre>Dim theVerificationProject As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationProject Dim theVerificationForm As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationForm ' Request the main form Project.GetVerifierProject theVerificationProject Set theVerificationForm = theVerificationProject.AllVerificationForms.ItemByName("Invoices") theVerificationForm.VerificationFields.ItemByName("Field_InvoiceNo").Invisible = True ' Update the form theVerificationForm.RepaintControls</pre>

Left

This property provides the left position of the field on the Verifier form.

Description	Definition
Attribute	Read/Write
Example	<p>The example below retrieves the Left position of the Invoice Number field from the Verifier form.</p> <pre> Dim theVerificationProject As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationProject Dim theVerificationForm As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationForm Dim LeftPos As Integer ' Request the main form Project.GetVerifierProject theVerificationProject Set theVerificationForm = theVerificationProject.AllVerificationForms.ItemByName("Invoices") LeftPos = theVerificationForm.VerificationFields.ItemByName("Field_InvoiceNo").Left </pre>

For more information, see [Top](#) and [Width](#).

Name

This property provides the Name of the field on the Verifier form.

Description	Definition
Attribute	Read
Example	<p>The example below retrieves the Name of the Invoice Number field from the Verifier form.</p> <pre> Dim theVerificationProject As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationProject Dim theVerificationForm As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationForm Dim FieldName As String ' Request the main form Project.GetVerifierProject theVerificationProject Set theVerificationForm = theVerificationProject.AllVerificationForms.ItemByName("Invoices") FieldName = theVerificationForm.VerificationFields.ItemByName("Field_InvoiceNo").Name </pre>

ReadOnly

This property determines if the verification field on the Verifier or Web Verifier form is editable or Read Only. For the Web Verifier, use this method in the VerifiedFormatLoad event.

Set the property to TRUE to make the field non-editable.

Description	Definition
Attribute	Read/Write
Example	<p>The example below sets the Invoice Number field as Read Only on the Verifier form.</p> <pre>Dim theVerificationProject As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationProject Dim theVerificationForm As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationForm ' Request the main form Project.GetVerifierProject theVerificationProject Set theVerificationForm = theVerificationProject.AllVerificationForms.ItemByName("Invoices") theVerificationForm.VerificationFields.ItemByName("Field_InvoiceNo").ReadOnly = True theVerificationForm.RepaintControls ' Update the form UI</pre>

TabIndex

This property allows the scripter to set the tab sequence number of the verification field on the Verifier form.

The Tab sequence is typically configured on the verification form in Designer. This script method allows the scripter to change the sequence number to re-ordering Tab sequence of fields.

Description	Definition
Attribute	Read/Write
Example	<p>The example below sets the Invoice Number field tab sequence on the Verifier form.</p> <pre>Dim theVerificationProject As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationProject Dim theVerificationForm As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationForm ' Request the main form Project.GetVerifierProject theVerificationProject Set theVerificationForm = theVerificationProject.AllVerificationForms.ItemByName("Invoices") theVerificationForm.VerificationFields.ItemByName("Field_InvoiceNo").TabIndex = 5</pre>

Top

This property provides the top position coordinates of the field on the Verifier form.

The scripter may choose to reorder positional information of the field if another element is being hidden. Using the `RepaintControls` method, the form UI is updated with the changes made.

Description	Definition
Attribute	Read/Write
Example	<p>The example below retrieves the Top position of the Invoice Number field from the Verifier form.</p> <pre> Dim theVerificationProject As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationProject Dim theVerificationForm As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationForm Dim TopPos As Integer ' Request the main form Project.GetVerifierProject theVerificationProject Set theVerificationForm = theVerificationProject.AllVerificationForms.ItemByName("Invoices") TopPos = theVerificationForm.VerificationFields.ItemByName("Field_InvoiceNo").Top </pre>

For more information, see `Left`, `Width`, and `RepaintControls`.

Type

This property provides the Field Type information of the field on the Verifier form.

The scripter may choose to review information based on the field type.

Description	Definition
Attribute	Read

Description	Definition
<p>Example</p>	<p>The example below retrieves the Field Type Information of the Invoice Number field from the Verifier form.</p> <pre> Dim theVerificationProject As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationProject Dim theVerificationForm As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationForm Dim FieldInfo As CdrVerifierFieldType ' Request the main form Project.GetVerifierProject theVerificationProject Set theVerificationForm = theVerificationProject.AllVerificationForms.ItemByName("Invoices") FieldInfo = theVerificationForm.VerificationFields.ItemByName("Field_InvoiceNo").Type </pre>

For more information, see `CdrVerifierFieldType`.

Width

This property provides the Width size information of the field on the Verifier form.

The scripter may choose to reorder or resize positional information of the field if another element is being hidden. Using the `RepaintControls` method, the form UI is updated with the changes made.

Description	Definition
<p>Attribute</p>	<p>Read/Write</p>
<p>Example</p>	<p>The example below retrieves the Width Information of the Invoice Number field from the Verifier form.</p> <pre> Dim theVerificationProject As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationProject Dim theVerificationForm As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationForm Dim WidthInfo As Integer ' Request the main form Project.GetVerifierProject theVerificationProject Set theVerificationForm = theVerificationProject.AllVerificationForms.ItemByName("Invoices") WidthInfo = theVerificationForm.VerificationFields.ItemByName("Field_InvoiceNo").Width </pre>

For more information, see `Left`, `Top`, and `RepaintControls`.

SCBCdrVerificationTable

Description

This interface is used to identify verification properties specific for table validation elements.

Methods and Properties

FontFont

This property sets or returns the font settings for the individual table field element.

Description	Definition
Syntax	FontFont As StdFont

BackgroundColorValid

This property sets or returns the background color for the individual verification table element, when the table cell is valid in terms of current validation status.

Description	Definition
Syntax	BackgroundColorValid As OLE_COLOR

BackgroundColorInvalid

This property sets or returns the background color for the individual verification table element, when the table cell is invalid in terms of current validation status.

Description	Definition
Syntax	BackgroundColorInvalid As OLE_COLOR

HeaderFont

This property sets or returns the font settings for all header buttons of the table field element, including row header buttons, column header buttons and the table header button (small control in the left-top corner of the table).

Description	Definition
Syntax	HeaderFont As StdFont

HeaderFontColor

This property sets or returns the font color for the header buttons of the table field element, including row header buttons and column header buttons.

Description	Definition
Syntax	HeaderFontColor As OLE_COLOR

HeaderBackgroundColor

This property sets or returns background color for all header buttons of the table field element, including row header buttons, column header buttons, and the table header button.

Description	Definition
Syntax	HeaderBackgroundColor As OLE_COLOR

SCBCdrVerificationButton

Description

Use this interface to set verification properties specific for all custom buttons defined on a verification form.

Methods and Properties

Font

This property sets or returns the font settings, such as name, type and style, for the individual custom button control.

Description	Definition
Syntax	Font As StdFont

FontColor

This property sets or returns the font color for the individual custom button control.

Description	Definition
Syntax	FontColor As OLE_COLOR

BackgroundColor

This property sets or returns background color for the individual custom button control.

Description	Definition
Syntax	<code>BackgroundColor As OLE_COLOR</code>

SCBCdrVerificationLabel

Description

This object is part of the Cedar Verifier Component Library. It enables the scripter to manipulate the verifier form labels.

Cedar Verifier Component Library is not enabled by default. This component can be added to the script references for any project class.

Properties

BackgroundColor

This property sets the color for the verification text label to display to the user. By default, the background color of the field is gray.

Description	Definition
Syntax	<code>BackgroundColor As OLE_COLOR</code>
Attribute	Read/Write
Example	<p>The example below turns the color for Invoice Number label to gray.</p> <pre>Dim theVerificationProject As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationProject Dim theVerificationForm As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationForm ' Request the main form Project.GetVerifierProject theVerificationProject Set theVerificationForm = theVerificationProject.AllVerificationForms.ItemByName("Invoices") theVerificationForm.VerificationLabels.ItemByName("Label_InvoiceNo").Ba ckgroundColor = RGB (192, 129, 129)</pre>

Font

This property sets the Font for the content of the verification field label.

Note To get the StdFont object, add OLE Automation as a reference.

Description	Definition
Attribute	Read/Write
Example	<p>The example below sets the Font for Invoice Number field label.</p> <pre> Dim theVerificationProject As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationProject Dim theVerificationForm As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationForm Dim DefaultLabelFont As New StdFont DefaultLabelFont.Bold = False 'Set Font attributes 'Request the main form Project.GetVerifierProject theVerificationProject Set theVerificationForm = theVerificationProject.AllVerificationForms.ItemByName("Invoices") theVerificationForm.VerificationLabels.ItemByName("Label_InvoiceNo").Font = DefaultLabelFont </pre>

For additional information see the `FontColor` property.

FontColor

This property sets the Font Color for the content of the verification field label.

Note To get the OLE_COLOR object, add OLE Automation as a reference.

Description	Definition
Attribute	Read/Write
Example	<p>The example below sets the FontColor for Invoice Number field label to blue.</p> <pre> Dim theVerificationProject As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationProject Dim theVerificationForm As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationForm 'Request the main form Project.GetVerifierProject theVerificationProject Set theVerificationForm = theVerificationProject.AllVerificationForms.ItemByName("Invoices") theVerificationForm.VerificationLabels.ItemByName("Label_InvoiceNo").FontColor = RGB(0, 0, 255) </pre>

For additional information, see the `Font` property.

Invisible

This property determines if the field label is visible or hidden on the Verifier form. The developer may script options to hide or display the field label from the verifier user.

Description	Definition
Attribute	Read/Write
Example	<p>The example below hides the Invoice Number field label from the verifier user.</p> <pre> Dim theVerificationProject As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationProject Dim theVerificationForm As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationForm ' Request the main form Project.GetVerifierProject theVerificationProject Set theVerificationForm = theVerificationProject.AllVerificationForms.ItemByName("Invoices") theVerificationForm.VerificationLabels.ItemByName("Label_InvoiceNo").Invi sible = True ' Update the form theVerificationForm.RepaintControls </pre>

Left

This property provides the left position of the field on the Verifier form.

Description	Definition
Attribute	Read/Write
Example	<p>The example below retrieves the Left position of the Invoice Number field label from Verifier Form.</p> <pre> Dim theVerificationProject As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationProject Dim theVerificationForm As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationForm Dim LeftPos As Integer ' Request the main form Project.GetVerifierProject theVerificationProject Set theVerificationForm = theVerificationProject.AllVerificationForms.ItemByName("Invoices") LeftPos = theVerificationForm.VerificationLabels.ItemByName("Label_InvoiceNo").Left </pre>

For additional information, see the `Top` and `width` properties.

Name

This property provides the Name of the field label on the Verifier form.

Description	Definition
Attribute	Read
Example	<p>The example below retrieves the Name of the Invoice Number field label from Verifier Form.</p> <pre> Dim theVerificationProject As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationProject Dim theVerificationForm As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationForm Dim FieldName As String ' Request the main form Project.GetVerifierProject theVerificationProject Set theVerificationForm = theVerificationProject.AllVerificationForms.ItemByName("Invoices") FieldName = theVerificationForm.VerificationLabels.ItemByName("Label_InvoiceNo").Name </pre>

Text

This property allows the scripter to set the text of the verification field label on the Verifier form.

Description	Definition
Attribute	Read/Write
Example	<p>The example below sets the Invoice Number field label text on the Verifier Form.</p> <pre> Dim theVerificationProject As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationProject Dim theVerificationForm As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationForm ' Request the main form Project.GetVerifierProject theVerificationProject Set theVerificationForm = theVerificationProject.AllVerificationForms.ItemByName("Invoices") theVerificationForm.VerificationLabels.ItemByName("Label_InvoiceNo").Text = "Invoice Number" </pre>

Top

This property provides the top position coordinates of the field label on the Verifier form.

The scripter may choose to reorder positional information of the field label if another element is being hidden. Using the RepaintControls method, the form UI is updated with the changes made.

Description	Definition
Attribute	Read/Write
Example	<p>The example below retrieves the Top position of the Invoice Number field label from Verifier Form.</p> <pre> Dim theVerificationProject As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationProject Dim theVerificationForm As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationForm Dim TopPos As Integer ' Request the main form Project.GetVerifierProject theVerificationProject Set theVerificationForm = theVerificationProject.AllVerificationForms.ItemByName("Invoices") TopPos = theVerificationForm.VerificationLabels.ItemByName("Label_InvoiceNo").Top </pre>

For additional information , see the [Left](#), [Width](#), and [RepaintControls](#).

Width

This property provides the Width size information of the field label on the Verifier form.

Description	Definition
Attribute	Read/Write
Example	<p>The example below retrieves the Width Information of the Invoice Number field label from the Verifier form.</p> <pre> Dim theVerificationProject As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationProject Dim theVerificationForm As DISTILLERVERIFIERCOMPLib.SCBCdrVerificationForm Dim WidthInfo As Integer 'Request the main form Project.GetVerifierProject theVerificationProject Set theVerificationForm = theVerificationProject.AllVerificationForms.ItemByName("Invoices") WidthInfo = theVerificationForm.VerificationLabels.ItemByName("Label_InvoiceNo").Width </pre>

For additional information, see the [Left](#), [Top](#), [RepaintControls](#) properties.

Password Encryption for Database Connection Strings

The application architecture of Perceptive Intelligent Capture makes it very important to be able to hide sensitive security information, such as DB access password, stored in Perceptive Intelligent Capture or custom project configuration files.

The same requirement also applies to the database connection strings in the Perceptive Intelligent Capture project INI files that often contain multiple connection strings to different database instances (like for Visibility reporting or custom databases) with unencrypted password info. These INI files may not reside directly on the local Verifier workstation, but still can be easily accessed by the Verifier users, because at least the read-only access to the Perceptive Intelligent Capture project directory is a requirement for Perceptive Intelligent Capture applications.

Note: Maximum number of characters allowed to encrypt is 30. Passwords longer than 30 characters would not be encrypted correctly. Below are the steps to implement password encryption for custom configuration files used when loading Perceptive Intelligent Capture projects.

Master Project Side (Project Primary Developer)

About encryption keys

Before you start, request a pair of RSA encryption keys from Lexmark Customer Support. In terms of testing you can though use the pair of test keys below. However, do request a new pair before releasing your master project to the others.

Important Keep your private key safe - do NOT share it with anyone else. Only the public key should be distributed to those who use your project for custom implementations.

Test Public Key

```
<RSAKeyValue><Modulus>vJ+W7SuXuvOrWVoy4tPrbflCuoHElo750cpTuEzLpK6iz6bHAodPVgLFaOEK+XMMS2G5z+6961v
uQsDGUt+01Ag1PiTXCa6rrAaeCaaDO4HI8Mmpw00kUZefCzPTTYCYQPFzlgokwomF6VDSB9d1US430IT0gctQY1b5iM4MqT0=
</Modulus><Exponent>AQAB</Exponent></RSAKeyValue>
```

Test Private Key

```
<RSAKeyValue><Modulus>vJ+W7SuXuvOrWVoy4tPrbflCuoHElo750cpTuEzLpK6iz6bHAodPVgLFaOEK+XMMS2G5z+6961v
uQsDGUt+01Ag1PiTXCa6rrAaeCaaDO4HI8Mmpw00kUZefCzPTTYCYQPFzlgokwomF6VDSB9d1US430IT0gctQY1b5iM4MqT0=
</Modulus><Exponent>AQAB</Exponent><P>8SRHEvT5Bn2paRHSDR9yCQb7WGYE9PbeHzuqwh6iWa0LNYJrSrhhUeCEpw1
PLQWQq10KmMZgG0+Br4nuBMmMHQ==</P><Q>yD719fjB/MJWYav3LcEzy286Q+Xvo74i6THvHkKqB1NKYGcN9xF9d8XbiUQNg
BZ/4F02T6mFeYDO32KFVRXHoQ==</Q><DP>nRDTFn7nwRmSgfRwi8minkyk5DQ3IFO35EIZ+x3Ao4Z52ZWkStwDz6/c12vR3X
JVg7irkU0NB1zoDK1bk1Sw5Q==</DP><DQ>B3xieGmORva05/2ZkPpSA3ubAALOjJ6FC5a0S7tOQ+vXMfdoTD45JIsfA+ipYI
p2yVpyt10tC7fHBA7Y0S95QQ==</DQ><InverseQ>4S1xqlXK9f1rawGCbFWOVp6lz1fCoQ8RfyDE87/G/pUi1HRJV2acBAcn
gY3c/MRMKrXQb8lx99k7dENUYc8ywQ==</InverseQ><D>KAL6cwkCQKgbuvKFRNSLZmFOqV2JpB5kI/p1U+0GWAs6Qi4wnPq
y+5303na0a2faPctXLSKJqvlvSz21VDMUCsyphvOSxBtc1cZHJp4ueQPA7u+qrIJaDY1Rh1AVoqNfCJFX6+McVJ+I/X+mZOCt
dUaCuAoNn014UYOamuJYDQE=</D></RSAKeyValue>
```


Implementation Guidelines

1. Split the connection string in your configuration files to encrypted and non-encrypted parts.

Example of connection string of *[ProjectName].ini* before splitting:

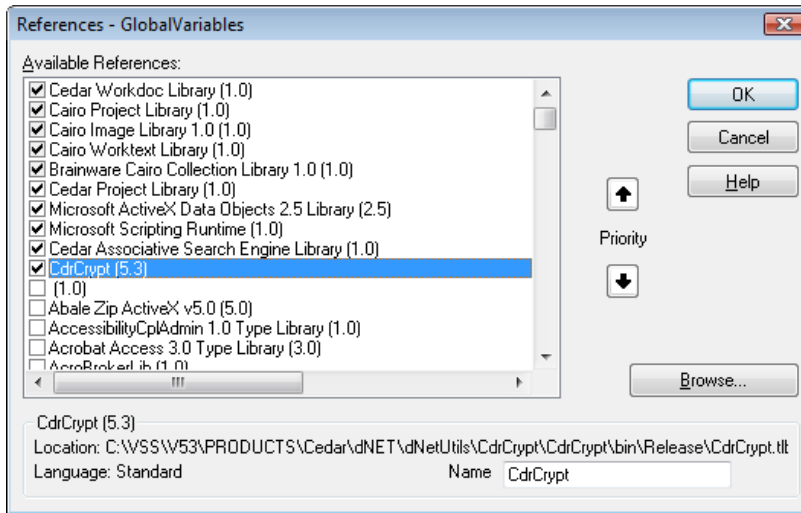
```
SQL_VL_01_ConnectionString=Provider=SQLOLEDB.1;Password=alexey 123456789;Persist Security Info=True;User ID=alexey;Initial Catalog=Visibility;Data Source=KIR-AE-NB-03\SQLSERVER2008R2
```

Example of connection string of *[ProjectName].ini* before splitting (the red part of the example below is now packaged as an extended new variable - see the red part below):

```
SQL_VL_01_ConnectionString=Provider=SQLOLEDB.1;Persist Security Info=True;User ID=alexey;Initial Catalog=Visibility;Data Source=KIR-AE-NB-03\SQLSERVER2008R2
```

```
SQL_VL_01_ConnectionPassword=encrypted_password_is_to_be_placed_here
```

2. Open your master project in Designer, run script editor, open the script page, where you would like to implement connection string encryption and add the Reference to "CdrCrypt (5.3)" type library:



- At the place of the same script page, where connection string is supposed to be read from the configuration (INI) file and then used further to connect to the database add a script code, similar to the one below:

```
Dim theCedarCryptographyHelper As New CdrCrypt.RSACodecInt
Dim strEncryptedPassword As String
Dim strOpenPassword As String
Dim strPrivateKey As String

strPrivateKey =
"<RSAKeyValue><Modulus>vJ+W7SuXuvOrWVoy4tPrbflCuoHElo750cpTuEzLPk6 iz6bHAodPVgLFaOEK+XMMS2G5z+
6961vuQsDGUt+01Ag1PiTXCa6rrAaeCaaDO4HI8Mmpw00kUZefCZpTTYCYQPfZlgokwomF6VDSB9d1US430IT0gctQY1b
5iM4MqT0=</Modulus><Exponent>AQAB</Exponent><P>8SRHEvT5Bn2paRHSDR9yCQb7WGYE9PbeHzuqwh6iWa0LNY
JrSrhhUeCEpw1PLQWQq10KmMZgG0+Br4nuBMmMHQ==</P><Q>yD719fjb/MJWYaV3LcEzY286Q+Xvo74i6THvHkKqB1NK
YGcN9xF9d8XbiUQNgbZ/4F02T6mFeYDO32KFVRXHoQ==</Q><DP>nRDTFn7nwRmSgfRwi8minkyk5DQ3IFO35EIZ+x3Ao
4Z52ZWkStwDz6/c12vR3XJVg7irkU0NB1zoDK1bk1Sw5Q==</DP><DQ>B3xieGmORva05/2ZkPpSA3ubAAL0jJ6FC5a0S
7tOQ+vXmfdoTD45JIsfA+ipYIp2yVpyt10tC7fHBA7Y0S95QQ==</DQ><InverseQ>4S1xqlXK9f1rawGCbFWOVp61z1f
CoQ8RfyDE87/G/pUilHRJV2acBAcngY3c/MRMKrxQb8lx99k7dENUYc8ywQ==</InverseQ><D>KAL6cwkCQKgbuvKFRN
SLZmFOqV2JpB5kI/p1U+0GWas6Qi4wnPqy+5303naOa2faPctXLSKJqvlvSz21VDMUCsyphvOSxBtc1cZHJp4ueQPA7u+
qrIJJaDY1Rh1AVoqNfCJFX6+McVJ+I/X+mZOCtdUaCuAoNn014UYOaMuJYDQE=</D></RSAKeyValue>"

strEncryptedPassword = DicVal("01" & "ConnectionPassword", "SQL")

If Len(strEncryptedPassword) > 0 Then
    strOpenPassword = theCedarCryptographyHelper.Decode(strEncryptedPassword, strPrivateKey)
End If

If Len(strOpenPassword) > 0 Then
    strConnection = strConnection + ";Password=" + strOpenPassword
End If
```

- Make sure you encrypt the script page that contains the code above via standard script code encryption feature.
Alternatively, you leave the code above unencrypted, but place the "strPrivateKey" variable and its initialization on another encrypted page available from the code above.
- When you release your master project to the others, distribute the public key along with the project release. The professional services representatives who are installing your project on site, use this public key to encrypt their custom passwords.

Advanced troubleshooting

To provide script dumps for script issues, such as compilation problems, that occur on the PIC Web Verifier, complete the following steps.

Note This feature requires advanced product knowledge.

- Launch the Windows registry editor.
- Complete one of the following substeps.
 - For a 32-bit machine, navigate to [HKEY_LOCAL_MACHINE\SOFTWARE\Lexmark\Cedar].
 - For a 64-bit machine, navigate to [HKEY_LOCAL_MACHINE\SOFTWAREWow6432Node\Lexmark\Cedar].
- In the right pane, right-click and then click **New > DWORD (32-bit) Value**.
- In the **Name** field, type **DumpProjectScriptCode** and then click **OK**.

5. Right-click the **DumpProjectScriptCode** key and click **Modify**.
6. In the **Edit DWORD (32-bit) Value** dialog box, in the **Value data** field, complete one of the following steps and then click **OK**.
 - To disable the feature, type **0**.
 - To enable the feature, type **3**.