Oracle® Fusion Middleware

WebCenter Forms Recognition Scripting User's Guide

14c (14.1.1.0.0)

F73580-01

August 2023

Describes how to write simple scripts to customize your processing

Oracle Fusion Middleware Oracle WebCenter Forms Recognition Scripting User’s Guide, 14c (14.1.1.0.0)

F73580-01

Table of Contents

About Scripting **18**
ScriptModule Events **18**
About the ScriptModule Event Interface **18**
Implement an Event 18
AppendWorkdoc 19
Syntax *19*
BatchClose 19
Syntax *19*
Script Code *20*
BatchOpen 21
Syntax *21*
Sample Code *22*
ExportDocument 22
Syntax *22*
Create a Script for Document Export *22*
Initialize 23
Syntax *23*
Sample Code *23*
MoveDocument 23
Syntax *24*
Sample Code *24*
PostClassify 24
Syntax *24*
Sample Code *24*
Sample Code *25*
Sample Code *25*
PostImport 26
Syntax *26*
See also *26*
PostImportBatch 26
Syntax *26*
Sample Code *28*
PostOCR 28
Syntax *28*

PreClassify 28
Syntax *28*
Sample Code *28*
PreClassifyAnalysis 28
Syntax *28*
PreImport 29
Syntax *29*
PreOCR 29
Syntax *29*
ProcessBatch 30
Syntax *30*
Sample Code *30*
RouteDocument 33
Syntax *33*
Sample Code *34*
Terminate 34
Syntax *34*
Sample Code *34*
UpdateSystemSecurity 35
Syntax *35*
Sample Code *35*
See also *35*
VerifierClassify 36
Syntax *36*
VerifierException 36
Syntax *36*
VerifierFormLoad 37
Syntax *37*
Sample Code *37*
See also *38*
Cedar DocClass Event Interface 38
FocusChanged 38
Syntax *38*
Sample Code *39*
Sample code *39*
OnAction 39

Syntax ... *39*
Sample Code ... *39*
PostExtract ... 40
Syntax ... *40*
Sample Code ... *40*
PreExtract ... 40
Syntax ... *40*
Sample Code ... *40*
PreVerifierTrain ... 40
Syntax ... *41*
Sample Code ... *41*
Validate ... 41
Syntax ... *41*
Sample Code ... *41*
VerifierTrain ... 42
Syntax ... *42*
Cedar [FieldName] Event Interface ... 42
CellChecked ... 42
Syntax ... *43*
Sample Code ... *43*
CellFocusChanged ... 43
Syntax ... *43*
Sample Code ... *44*
Format ... 44
Syntax ... *44*
Sample Code ... *45*
FormatForExport ... 45
Syntax ... *45*
Sample Code ... *45*
PostAnalysis ... 45
Syntax ... *45*
Sample Code ... *46*
PostEvaluate ... 46
Syntax ... *46*
Sample Code ... *46*
PreExtract ... 46

Syntax ... *46*
Sample Code ... *47*
SmartIndex ... 47
Syntax ... *47*
Sample Code ... *47*
TableHeaderClicked ... 47
Syntax ... *47*
Sample Code ... *48*
Validate ... 48
Syntax ... *48*
Sample Code ... *49*
ValidateCell ... 49
Syntax ... *49*
Sample Code ... *49*
ValidateRow ... 49
Syntax ... *50*
Sample Code ... *50*
ValidateTable ... 50
Syntax ... *50*
Sample Code ... *50*
Working with Workdoc Objects (SCBCdrWKDOCLib) ... 51
SCBCdrWKDOCLib Type Definitions ... 51
CDRClassifyResult ... *51*
CDRDocFileType ... *52*
CDRDocState ... *52*
CdrEdgeSide ... *52*
CdrExportType ... *53*
CDRFieldState ... *53*
CDRHighlightMode ... *53*
CDRLocation ... *54*
CDRPageAssignment ... *55*
CDRPageSource ... *55*
CDRPDFExportStyle ... *55*
CDRTableHighlightMode ... *55*
CDRTranslationLanguage ... *57*
CroLinesDir ... *57*

CroLinesKooType......57
SCBCdrCandidate......58
SCBCdrCandidate Methods......58
SCBCdrCandidate Properties......58
SCBCdrDocPage......63
SCBCdrDocPage Methods......63
SCBCdrDocPage Properties......63
SCBCdrEmailProperties......67
SCBCdrEmailProperties Properties......67
SCBCdrField......68
SCBCdrField Methods......68
SCBCdrField Properties......74
SCBCdrFields......82
SCBCdrFields Methods......82
SCBCdrFields Properties......85
SCBCdrFolder......87
SCBCdrFolder Methods......87
SCBCdrFolder Properties......89
SCBCdrTable......90
SCBCdrTable Methods......90
SCBCdrTable Properties......95
SCBCdrTextblock......111
SCBCdrTextblock Properties......111
SCBCdrWord......113
SCBCdrWord Properties......113
SCBCdrWorkdoc......115
SCBCdrWorkdoc Methods......115
SCBCdrWorkdoc Properties......130
SCBCdrWorkdoc Named Properties......143
SCBCdrPROJLib......150
SCBCdrPROJLib Type Definitions......150
CDRApplicationName......150
CDRBatchReleaseAction......150
CDRClassifyMode......151
CDRDatabaseWorkflowTypes......151
CdrDocumentBinarizationMode......153

CDRFieldType ... *154*
CdrFocusChangeReason ... *154*
CdrForceValidationMode ... *155*
CdrLicenseCounter ... *155*
CdrLicenseFeatureName ... *156*
CdrLocalTrainingReason ... *159*
CdrMessageSeverity ... *160*
CdrMessageType ... *160*
CdrMPType ... *160*
CDRsiModule ... *161*
CdrSLWDifferentResultsAction ... *161*
CdrTableFocusChangeReason ... *162*
CdrTableHeaderClickType ... *163*
CdrValFieldType ... *163*
CdrVerifierClassifyReason ... *163*
CDRVerifierExceptionReason ... *164*
SCBCdrProject Methods ... 164
ExtractClassificationField ... *164*
GetHostProperties ... *165*
GetVerifierProject ... *166*
Lock ... *166*
LogScriptMessage ... *166*
LogScriptMessageEx ... *166*
MoveDocClass ... *167*
PerformScriptCommandRTS ... *167*
ReleaseAllAdsPools ... *168*
ReportLicensingStatus ... *169*
SecurityUpdateAddUserGroup and SecurityUpdateAddUserGroupPwd ... *170*
SecurityUpdateCommit ... *172*
SecurityUpdateStart ... *172*
SecurityUpdateUserParameter ... *173*
ShowValidationTemplates ... *174*
Unlock ... *174*
UpdateAddressPool ... *174*
SCBCdrProject Properties ... 175
AllClasses ... *175*

BaseClasses *175*
ClassificationMode *175*
CurrentClient *175*
DefaultClassifyResult *176*
DefaultLanguage *176*
Filename *176*
ForceValidation *176*
MinClassificationDistance *177*
MinClassificationWeight *177*
MinParentClsDistance *177*
MinParentClsWeight *177*
NoUI *177*
Page *178*
ParentWindow *178*
SLWDifferentResultsAction *178*
SLWSupplierInvalidDifferentClsResults *178*
ValidationSettingsColl *178*
ValidationTemplates *178*
VersionCount *179*
WordSegmentationChars *179*
SCBCdrDocClass 179
SCBCdrDocClass Methods *179*
SCBCdrDocClass Properties *181*
SCBCdrDocClasses 184
SCBCdrDocClasses Properties *184*
SCBCdrFieldDef 186
SCBCdrFieldDef Methods *186*
SCBCdrFieldDef Properties *188*
SCBCdrFieldDefs 194
SCBCdrFieldDefs Properties *194*
SCBCdrLicenseInfoAccess 196
SCBCdrLicenseInfoAccess Methods *196*
SCBCdrSettings 199
SCBCdrSettings Methods *199*
SCBCdrSettings Properties *201*
SCBCdrScriptModule 204

SCBCdrScriptModule Methods *204*
SCBCdrScriptModule Properties *204*
SCBCdrScriptAccess 205
SCBCdrScriptAccess Methods *205*
Working with the Associative Search Engine (CDRADSLib) 208
SCBCdrSupExSettings Type Definitions 208
CdrAutoUpdateType *208*
SCBCdrSupExSettings Methods 208
AddColumn *208*
AddFilterAttribute *208*
AddPhrase *209*
ChangeEntry *210*
ClearFilterAttributes *210*
CommitAddEntry *210*
CommitUpdate *211*
GeneratePool *211*
GeneratePoolFromCsvFile *211*
GeneratePoolFromODBC *211*
GetClassNameByID *212*
GetEntry *213*
GetFormattedValueByID *213*
GetIDByIndex *213*
GetIndexById *214*
GetSearchArea *214*
RemovePhrase *215*
SetConnectionPropertiesRMS .... ***Error! Bookmark not defined.***
SetSearchArea *215*
StartAddEntry *215*
StartUpdate *216*
SCBCdrSupExSettings Properties 216
ClassNameFormat *216*
ColumnCount *216*
ColumnName *216*
EnableCandidateEvaluation *217*
EntryCount *217*
EvalFirstPageOnly *217*

EvalOverSearchAreas ... *217*
FieldContentsFormat ... *218*
FindLocation ... *218*
IdentityColumn ... *218*
ImportFieldNames ... *218*
ImportFileName ... *218*
ImportFileNameRelative ... *218*
IsPhraseIncluded ... *218*
IsSearchField ... *219*
LastImportTimeStamp ... *219*
MaxCandidates ... *219*
MinDistance ... *219*
MinRelevance ... *219*
MinThreshold ... *220*
NumberKeepLocalCopies ... *220*
ODBCName ... *220*
Passwords ... *220*
Phrase ... *220*
PhrasesCount ... *221*
PoolName ... *221*
PoolPath ... *221*
PoolPathRelative ... *221*
PoolVersion ... *221*
ProjectPath ... *221*
SavePoolInternal ... *222*
SearchAreaActive ... *222*
Separator ... *222*
SQLQuery ... *222*
UserName ... *222*
VendorTypeColumn ... *223*
SCBCdrSupplierExtractionEngine Methods ... 223
SearchField ... *223*
Working with the Check Analysis Engine (SCBCdrParaCheckLib) ... 224
SCBCdrParaCheckLib Type Definitions ... 224
SCBCdrParaCheckAnalysisSettings Methods ... 224
AddItemToVocabulary ... *224*

ClearVocabulary *225*
SCBCdrParaCheckAnalysisSettings Properties 226
FieldType *226*
MinDistance *226*
MinWeight *226*
PayeeLineRecMode *226*
PayeeVocEntries *227*
Vocabulary Entry Objects (PayeeVocEntries) 227
PayeeVocEntries Methods *227*
PayeeVocEntries Properties *228*
Working with the Format Analysis Engine (SCBCdrFormatEngine) 230
Sample code 230
Sample Code 231
Sample Code 231
SCBCdrFormatEngine Methods 232
FindStringFirst *232*
FindStringNext *232*
GetBlockID *233*
TestString *234*
SCBCdrFormatEngine Properties 234
SrchFlag *234*
Working with Format Settings (SCBCdrFormatSettings) 235
Sample code 235
SCBCdrFormatSettings Type Definitions 235
CdrAnalysisMethod *235*
CdrDesignatorType *236*
SCBCdrFormatSettings Methods and Properties to Modify the List of Format Strings .. 236
AddFormat *236*
DeleteAll *237*
DeleteFormat *237*
FormatCount *237*
MoveFormat *237*
SCBCdrFormatSettings Methods and Properties to Specify the Settings for each Format String 237
AllowMultilineCandidates *238*
AnalysisMethod *238*

FormatValid *239*
IgnoreCharacters *239*
Prefix *240*
Suffix *240*
SCBCdrFormatSettings Methods and Properties to Specify the General Search Settings 240
CaseSensitive *240*
KeepSpaces *240*
MaxDistance *241*
MaxWordCount *241*
MaxWordGap *241*
MaxWordLen *241*
ResetTranslationLanguage *241*
SettingsCheckSum *241*
SetTranslationLanguage *242*
SCBCdrFormatSettings Methods and Properties to Specify the Settings for the Search Regions 242
BottomFirst *242*
BottomLast *242*
BottomSubseq *242*
LeftFirst *243*
LeftLast *243*
LeftSubseq *243*
RightFirst *243*
RightLast *243*
RightSubseq *244*
UseFirstPage *244*
UseLastPage *244*
UseRegions *244*
UseSubseqPage *244*
TopFirst *244*
TopLast *245*
TopSubseq *245*
Working with Batches (SCBCdrBATCHLib) 245
SCBCdrBatchRoot Methods 245
Connect *245*
Disconnect *246*

SetConnectionProperties *246*
SetFilter *246*
SetLoginProperties *247*
SCBCdrBatchRoot Properties 247
ActivateSupport *247*
BatchCount *248*
Comparing Strings (SCBCdrSTRCOMPLib) 248
SCBCdrSTRCOMPLib Type Definitions 248
CdrCompareType *248*
SCBCdrStringComp Methods 249
Distance *249*
ValidateSearchExpression *249*
SCBCdrStringComp Properties 250
CaseSensitive *250*
CompTypes *250*
LevInsertions *250*
LevRejects *250*
LevReplacements *250*
LevSame *251*
LevTraceMatrix *251*
LevTraceResult *251*
MatchEndPosition *251*
MatchStartPosition *251*
SearchExpression *251*
Working with Images (SCBCroIMGLib) 251
SCBCroIMGLib Type Definitions 252
CroDocToView *252*
CROImgIOMode *252*
CROImgFileTypes *252*
CROImgType *258*
CROLDFLayer *258*
CROTiffTagType *258*
CROViewAlignmentX *259*
CROViewAlignmentY *259*
CROViewMode *259*
CroViewerSelectionFunctionality *260*

CroViewToDoc ... *260*
Image Object SCBCroImage ... 260
SCBCroImage Methods ... *260*
SCBCroImage Properties ... *271*
Sample Code ... *278*
Viewer Object SCBCroViewer ... 278
SCBCroViewer Methods ... *278*
SCBCroViewer Properties ... *281*
SCBCroViewer Events ... *286*
Sample Code ... *291*
Working with CI Documents (SCBCroCiDocLib) ... 292
SCBCroCiDocLib Type Definitions ... 292
CROViewDim ... *292*
CI Document Object SCBCroCIDOC ... 292
SCBCroCIDOC Methods ... *292*
SCBCroCIDOC Properties ... *295*
Sample Code ... *296*
CI Document Conversion Object SCBCroCIDOCConv ... 296
SCBCroCIDOCConv Methods ... *296*
Working with Cairo Projects (SCBCroPROJLib) ... 297
SCBCroPROJLib Type Definitions ... 297
CROcPagePrepStep ... *297*
CROcPageState ... *298*
CROcZoneState ... *298*
CroEngineType ... *298*
CroPrepDirection ... *299*
CROResetContext ... *299*
CROZoneSelectState ... *299*
Project Object SCBCroProject ... 299
SCBCroProject Methods ... *299*
SCBCroProject Properties ... *304*
SCBCroProject Events ... *305*
Page Object SCBCroPage ... 308
SCBCroPage Methods ... *308*
SCBCroPage Properties ... *310*
Pages Object SCBCroPages ... 313

SCBCroPages Methods *313*

SCBCroPages Properties *316*

Zone Object SCBCroZone 318

SCBCroZone Methods *318*

SCBCroZone Properties *320*

Zones Object SCBCroZones 324

SCBCroZones Methods *324*

SCBCroZones Properties *326*

Working with Collections (SCBCroCollLib) 328

Collection Object SCBCroCollection 328

SCBCroCollection Methods *328*

SCBCroCollection Properties *331*

SCBCroCollection Events *334*

Working with Worktext Objects (SCBCroWorktextLib) 335

SCBCroWorktextLib Type Definitions 335

CroWorktextDirection *335*

DimensionInfo_t *336*

PosInfo_t *336*

WktConversionType *336*

SCBCroWorktext 337

SCBCroWorktext Methods *337*

SCBCroWorktext Properties *355*

SCBCroWorktext Events *359*

Working with Verification Forms (DISTILLERVERIFIERCOMPLib) 360

DISTILLERVERIFIERCOMPLib Type Definitions 360

CdrVerifierFieldType *360*

SCBCdrVerificationForm 360

SCBCdrVerificationForm Methods *360*

SCBCdrVerificationForm Properties *361*

SCBCdrVerificationField 363

SCBCdrVerificationField Properties *363*

SCBCdrVerificationTable 368

SCBCdrVerificationTable Properties *368*

SCBCdrVerificationButton 369

SCBCdrVerificationButton Properties *369*

SCBCdrVerificationLabel 369

SCBCdrVerificationLabel Properties *369*
Printing Documents (SCBCroPrintLib) 373
SCBCroPrintLib Type Definitions 373
CROImgFileTypes *373*
CroPrintPageRange *377*
CroPrintPageSelection *378*
Orientation *378*
ResizeImage *378*
Print Object SCBCroPrint 378
SCBCroPrint Properties *378*
SCBCroPrint Methods *382*
Error Handling 385
On Error 385
Notes *385*
ErrObject 386
Sample Code 386
Troubleshoot Scripting Issues 386
Encryption 387
Password Encryption for Database Connection Strings 387
Password length *387*
Encrypt a password *387*
Decrypt a password *388*
Encryption keys for testing purposes *389*
Script Encryption 390

About Scripting

Oracle WebCenter Forms Recognition provides a scripting language that enables you to write simple scripts to customize your processing.

ScriptModule Events

The ScriptModule object provides the following events.

- AppendWorkdoc
- BatchClose
- BatchOpen
- ExportDocument
- Initialize
- MoveDocument
- PostClassify
- PostImport
- PostImportBatch
- PostOCR
- PreClassify
- PreClassifyAnalysis
- PreImport
- PreOCR
- ProcessBatch
- RouteDocument
- Terminate
- UpdateSystemSecurity
- VerifierClassify
- VerifierException
- VerifierFormLoad

About the ScriptModule Event Interface

Project events are specific for one project. However, within a WebCenter Forms Recognition 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.

Implement an Event

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

1. In **Designer**, open the project.
2. Switch to **Definition Mode**.
3. In the left pane, on the **Classes** tab, select the project.

4. On the toolbar, click **Show/hide script** .
5. In the **Script View for Project** dialog box, from the **Object** list, select **Script Module**.
6. From the **Proc** list, select the required event.
7. Modify the script according to your needs.

AppendWorkdoc

This event triggers when processing a document separation workflow step in Runtime Server. It appends a given workdoc after the last workdoc based on the append type.

Syntax

```
ScriptModule_AppendWorkdoc (pLastWorkdoc as ISCBCdrWorkdoc,
pCurrentWorkdoc as ISCBCdrWorkdoc, pAppendType as CdrMPType)
```

Parameter	Description
pLastWorkdoc	The last workdoc.
pCurrentWorkdoc	The current workdoc.
pAppendType	Defines the possible results of the multi-page classification. **Possible values** CdrMPType

BatchClose

This event triggers when a Verifier or Web Verifier user exits a batch in one of the following ways.

- When verifying a batch and selecting Return to batch list.
- Batch verification completion.
- Partial batch verification completion.
- The user exits Verifier while in a batch.

Syntax

```
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)
```

Parameter	Description
UserName	The currently logged in user name who has closed the batch.

BatchDatabaseID	The unique Batch ID within the database. For the file system, this batch ID is not used.
ExternalGroupID	The Group ID assigned to a batch. 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. For example, a batch belonging to Group ID 80 is only accessible by a user who is assigned to group 80. You can change this read or write parameter to any long value.
ExternalBatchID	The External Batch ID assigned to a batch. 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. You can change this read or write parameter to any long value.
TransactionID	The Transaction ID assigned to a batch. 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. You can change this read or write parameter to any long value.
WorkflowType	Corresponds to CDRDatabaseWorkflowTypes data type.
BatchState	The current status of an open batch, such as status 550 (Extraction Verification).
BatchReleaseAction	Batch Release Action represents the action taken when the last document of the batch verifies. The parameter can be set or read from a script. By default, it is set to `CDRBatchReleaseActionUserDefined` (as a user always makes a selection). If you use a registry value to hide the batch release dialog box in Verifier, then the last action taken prior to the dialog box being hidden is the one showing in this parameter. The scripter can set an override value to this parameter, such as every time batch verification completes, always goes to the next invalid batch. **Possible values** CDRBatchReleaseAction

Script Code

```
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
```

BatchOpen

This event triggers when a user opens a batch.

Syntax

```
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)
```

Parameter	**Description**
UserName	The user name currently logged in who opened the batch.
BatchDatabaseID	The unique batch ID in the database displayed as a numeric value. For the file system, this batch ID is not used. For example, batch 00000061 returns the value 61.
ExternalGroupID	The Group ID assigned to a batch. The Group ID used with the scripting security methods that enable the developer to assign a batch to a security group. Only those users belong to the same Group ID are able to access batches. For example, a batch belonging to Group ID 80 is only accessible by a user assigned to group 80. You can change this read or write parameter to any long value.
ExternalBatchID	The External Batch ID assigned to a batch. 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. You can change this read or write parameter to any long value.
TransactionID	The Transaction ID assigned to a batch. 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. You can change this read or write parameter to any long value.
WorkflowType	Corresponds to CDRDatabaseWorkflowTypes data type.
BatchState	The current status of the batch being opened, such as status 550 (Extraction Verification).

Sample Code

The following sample code logs the Batch ID and User name that opened a batch with date and time.

LogMessage is a custom function that writes a text line into a log file with Date/Time as a prefix.

```
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
```

ExportDocument

This event 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.

Syntax

```
ScriptModule_ExportDocument (pWorkdoc as ISCBCdrWorkdoc, ExportPath as
String, pCancel as Boolean)
```

Description	Definition
pWorkdoc	Workdoc to export.
ExportPath	Export path as configured in the Runtime Server settings (no changes possible).
pCancel	Set this variable to TRUE to cancel the export.

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** .
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.

   Example

   ```
   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.

Initialize

This event triggers when a batch opens for processing.

Syntax

```
ScriptModule_Initialize (ModuleName as String)
```

Parameter	Description
ModuleName	Name of the current module. **Possible values** - Server - Designer - Verifier - Thin Client Verifier

Sample Code

```
Public Sub ScriptModule_Initialize(ByVal ModuleName as String)
DBname=Project.Filename DBname=Left(DBname,InStrRev(DBname,''\'')) &
''InvoiceBestellNo.mdb'' Set DB=OpenDatabase(DBname) End Sub
```

MoveDocument

This event triggers when a Verifier or Web Verifier user places a document in the exception batch and the document moves out of the batch.

The ScriptModule provides the following event information.

- Old Batch ID
- New Batch ID
- Reason
- Document state

For the event to trigger, the condition must be set within the program settings that a new exception batch is created when a user places a document to exception.

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

After placing a document into the exception batch, the event triggers in the following ways.

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

Syntax

```
ScriptModule_MoveDocument(pWorkdoc as SCBCdrPROJLib.SCBCdrWorkdoc, ByVal
OldBatchID as String, ByVal NewBatchID as String, ByVal Reason as
SCBCdrPROJLib.CDRMoveDocumentReason)
```

Parameter	Description
pWorkdoc	The workdoc object used. No changes happen to the workdoc within this event.
OldBatchID	The batch ID of the document prior to moving a document to exception.
NewBatchID	The new batch ID where the document moved.
Reason	The reason the event triggers. Possible value • CDRMoveDocumentToExceptionBatch: The document moved to exception.
DocState	The workflow state of the document.

Sample Code

The following sample code logs a general message for each document placed into exception, showing the old batch ID and the new batch ID.

```
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
```

PostClassify

This event triggers after all defined classification methods are executed by the Cedar Project.

Syntax

```
ScriptModule_PostClassify (pWorkdoc as SCBCdrPROJLib.SCBCdrWorkdoc)
```

Parameter	Description
pWorkdoc	Classified workdoc object.

Sample Code

```
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 works only 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 '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
```

Sample Code

```
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
```

Sample Code

The following script retrieves OMR blackness.

```
Private Sub ScriptModule_PostClassify(pWorkdoc As
SCBCdrPROJLib.ISCBCdrWorkdoc) Dim oWorktext As New SCBCroWorktext Dim
lCharIndex, lTagType As Long Dim vTagValue As Variant Dim
strBlacknessValue As String Dim oImage As SCBCroImage Dim lImageWidth,
lImageHeight As Long Dim oPage As SCBCroPage Dim strMessage As String On
Error Resume Next Set oImage = pWorkdoc.Image(0) If oPage Is Nothing Then
Project.LogScriptMessageEx CDRTypeError, CDRSeverityLogFileOnly, "Image
object is Null" End If lImageWidth = oImage.Width lImageHeight =
oImage.Height strMessage = "Image Width == " & lImageWidth & ", while
Image Height == " & lImageHeight If ScriptModule.ModuleName = "Designer"
Then MsgBox strMessage Else Project.LogScriptMessageEx(CDRTypeInfo,
CDRSeveritySystemMonitoring, strMessage) End If Set oPage =
Project.AllClasses.ItemByName("Invoices").Page If oPage Is Nothing Then
Project.LogScriptMessageEx CDRTypeError, CDRSeverityLogFileOnly, "Page
```

```
object is Null" End If ' Enable retrieval of OMR blackness in RTS Set
oPage.Image = oImage oPage.Zones.ItemByName("MyOmrZone").Recognize
(oWorktext) 'Retrieve the blackness info for the OMR zone oWorktext.GetTag
(0, lCharIndex, lTagType, vTagValue) strBlacknessValue = CStr(CDbl
(vTagValue)) strMessage = "OMR Recognition Result == " & oWorktext.Text &
Chr$(13) & "OMR Blackness == " & strBlacknessValue If
ScriptModule.ModuleName = "Designer" Then MsgBox strMessage Else
Project.LogScriptMessageEx(CDRTypeInfo, CDRSeveritySystemMonitoring,
strMessage) End If End Sub
```

PostImport

In general, this event triggers after the import.

Processing for failed documents imported from the file system

- The event only triggers if **Perform advanced import failure processing** is active in Runtime Server.
- The event does not trigger if the `pCancel` parameter was set to `true` in the PreImport event.

Processing for failed documents imported from external batches

- The event always triggers.

Syntax

```
ScriptModule_PostImport(pWorkdoc As SCBCdrWorkdoc)
```

Parameter	Description
pWorkdoc	Workdoc object to import.

See also

"Activate advanced import failure processing" in the *Oracle WebCenter Forms Recognition Runtime Server User's Guide*.

PostImportBatch

The Runtime Server triggers this event after it finishes importing the batch. Use this event to set batch properties such as batch name and external group ID.

Syntax

```
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)
```

Parameter	Description
BatchDatabaseID	The unique Batch ID from the database that corresponds to the numeric BatchID within the associated database tables. **Note:** You cannot change this read-only parameter.
BatchName	The Batch Name assigned by the Runtime Server instance. The name comes from the Import settings of the Runtime Server instance. You can change this read or write parameter to any string value.
Priority	The Batch priority assigned by the Runtime Server instance. The priority comes from the Import settings of the Runtime Server instance. You can change this read or write parameter to any long value between 1 and 9.
State	The Batch State assigned by the Runtime Server instance. The status comes from the Import Success workflow setting in the Runtime Server instance. You can change this read or write parameter to any long value between 100 and 999.
ExternalGroupID	The Group ID assigned to a batch. The Group ID 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. For example, a batch belonging to Group ID 80 is only accessible by another user assigned to group 80. You can change this read or write parameter to any numeric value.
ExternalBatchID	The External Batch ID assigned to a batch. 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. You can change this read or write parameter to any numeric value.
TransactionID	The Transaction ID assigned to a batch. The Transaction ID allows the developer to synchronize a newly created batch of documents with another external system in order to identify originators of a batch of documents. You can change this read or write parameter to any long value.
TransactionType	The Transaction Type assigned to a batch. The Transaction Type allows the developer to synchronize a newly created batch of documents with another external system. It identifies the types of documents (Invoices, Claim forms etc.) in batches or the source of a document, such as Email or Scanned. You can change this read or write parameter to any long value.

Sample Code

The following sample code updates the batch priorities after the import has been done. It changes the name, state and adds a group ID as well as a transaction type and an ID.

```
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
```

PostOCR

This event triggers after the OCR process.

Syntax

```
ScriptModule_PostOCR(pWorkdoc As SCBCdrWorkdoc)
```

Parameter	Description
pWorkdoc	Workdoc object to OCR.

PreClassify

This event triggers before any defined classification method executes by the Cedar project. During this event, it is possible to apply an existing name of a `DocClass` to the workdoc.

Syntax

```
ScriptModule_PreClassify (pWorkdoc as SCBCdrWorkdoc)
```

Parameter	Description
pWorkdoc	Workdoc object to classify.

Sample Code

```
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
```

PreClassifyAnalysis

This event triggers 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.

Syntax

```
ScriptModule_PreClassifyAnalysis(pWorkdoc As SCBCdrWorkdoc)
```

Parameter	Description
pWorkdoc	Workdoc object to classify.

PreImport

This event triggers before the import occurs.

Syntax

```
ScriptModule_PreImport(pWorkdoc As SCBCdrWorkdoc, FilePath As String,
FileType As CDRDocFileType, pCancel As Boolean)
```

Parameter	Description
pWorkdoc	Workdoc object
FilePath	Path of the document to import.
FileType	File type **Possible values** See CDRDocFileType
pCancel	True: Skip the import of the current document. **Notes:** • When importing documents from an external batch, the PostImport event triggers. • When importing documents from the file system, the PostImport event does not trigger. False: Import the current document.

PreOCR

This event triggers before the OCR process occurs.

Syntax

```
ScriptModule_PreOCR(pWorkdoc As SCBCdrWorkdoc, pCancel As Boolean)
```

Parameter	Description
pWorkdoc	Workdoc object to OCR.
pCancel	Set this parameter to `True` to skip the OCR process for the current document.

ProcessBatch

The ProcessBatch event triggers when the Runtime Server instance begins processing during the custom processing workflow step.

Syntax

```
ScriptModule_ProcessBatch(pBatch as SCBCdrPROJLib.ISCBCdrBatch, ByVal
InputState as Long, DesiredOutputStateSucceeded as Long,
DesiredOutputStateFailed as Long)
```

Parameter	Description
pBatch	The Batch Object that is processed.
InputState	The input state of the batch when custom processing was activated on it.
DesiredOutputStateSucceeded	The output state of the batch if the workflow step succeeds.
DesiredOutputStateFailed	The output state of the batch if the workflow step failed.
	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 makes it impossible to load the ProcessBatch script.

Sample Code

Add the following sample code to the very beginning of the ProcessBatch event to stop an indefinite looping process of state 0 batches.

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 program uses the default value, which is zero.

```
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 foldersdirectories: [" &
pBatch.BatchID & "]" pBatch.BatchState = 987 End If On Error Resume Next
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 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 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 ' [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 ' 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
 ' 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 Next lDocIndex Next
lFolderIndex LABEL_SUCCESS: Project.LogScriptMessageEx CDRTypeInfo,
CDRSeveritySystemMonitoring, "ScriptModule_ProcessBatch finished
successfully, batch <" & CStr(pBatch.BatchID) & ">, new state <" & CStr
(DesiredOutputStateSucceeded) & ">, old state <" & CStr(InputState) & ">"
Exit Sub LABEL_ERROR: Project.LogScriptMessageEx CDRTypeError,
CDRSeveritySystemMonitoring, "ScriptModule_ProcessBatch, finished with
Error: " & Err.Description End Sub
```

RouteDocument

This event triggers when the system extracts a document.

Syntax

```
ScriptModule_RouteDocument (pWorkdoc as ISCBCdrWorkdoc, State as Single)
```

Parameter	Description
pWorkdoc	Workdoc object that was classified and extracted.
State	This parameter contains the current state assigned to the workdoc. The value can be changed from script.

Sample Code

```
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
```

For example, in an environment where the batch directory is shared between multiple organizations (either country groups, or departments), it is possible to allocate verifiers their own workflow configurations.

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).

```
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
```

Terminate

This event triggers before a batch closes after processing.

Syntax

```
ScriptModule_Terminate (ModuleName as String)
```

Parameter	Description
ModuleName	Name of the current module. **Possible values** - Designer - Verifier - Server

Sample Code

```
Private Sub ScriptModule_Terminate(ByVal ModuleName as String) DB.Close
Set DB = nothing End Sub
```

This script when added to one of the real projects triggers the Terminate event in Runtime Server. This script erases all state 0 batches that contain zero directories. Do not use this script piece together with the corresponding ProcessBatch event script within one project, because this Terminate event script makes 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("Job
name", "Zero Folder Batch Terminator", False) pBatchRoot.Connect("Job
name", "", "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 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
```

UpdateSystemSecurity

This event triggers when running security updates in Runtime Server.

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.

Syntax

```
ScriptModule_UpdateSystemSecurity(ByVal InstanceName as String)
```

Parameter	Description
InstanceName	The Runtime Server instance name that calls the UpdateSystemSecurity event.

Sample Code

The following sample code updates the database user security on a regular basis. The script can be updated to lookup users/roles and update the WebCenter Forms Recognition user table.

```
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, "VER|SET",
"BDomain" Project.SecurityUpdateAddUserGroup "User6", 777, "VER|FLT",
"BDomain" Project.SecurityUpdateAddUserGroup "User7", 333, "AEB",
"BDomain" Project.SecurityUpdateAddUserGroup "User10", 777, "ADM",
"BDomain" Project.SecurityUpdateCommit End Sub
```

See also

- SecurityUpdateStart
- SecurityUpdateCommit

- SecurityUpdateUserParameter
- PostImportBatch

VerifierClassify

This event triggers in Verifier in the following cases.

- When the user opens the **Class** list for a document, the event triggers with `Reason = CdrInitReason` and `ClassName = initial Class name`.
- When the user closes the **Class** list, for example, by pressing Enter or by selecting a new document, the event triggers with `Reason = CdrValidatedReason` and `ClassName = new Class name`.

Syntax

```
ScriptModule_VerifierClassify (pWorkdoc as ISCBCdrWorkdoc, Reason as
CdrVerifierClassifyReason, ClassName as String)
```

Parameter	Description
pWorkdoc	Reference to the currently processed document.
Reason	The reason why the script routine decided to reject or accept the document. **Possible values** CdrVerifierClassifyReason
ClassName	The name of the document class to which it is classified manually.

VerifierException

This event triggers when a document or batch is moved to exception state in Verifier.

Syntax

```
ScriptModule_VerifierException(pWorkdoc As SCBCdrWorkdoc, Reason As
SCBCdrPROJLib.CDRVerifierExceptionReason, CreateNewBatch As Boolean,
BatchName As String, BatchDocumentState As Long, BatchPriority As Long,
BatchFolderName As String, ApplyExceptionHandling As Boolean)
```

Parameter	Description
pWorkdoc	Workdoc object.
Reason	Exception reason. **Possible values** See CDRVerifierExceptionReason

CreateNewBatch	True: Create a new exception batch. False: Do not create a new exception batch.
BatchName	Name of the new exception batch.
BatchDocumentState	State of the new exception batch.
BatchPriority	Priority of the new exception batch.
BatchFolderName	Name of the folder in the exception batch.
ApplyExceptionHandling	True: Move the document/batch to exception. False: Do not move the document/batch to exception.

VerifierFormLoad

This event triggers optionally in Designer before the Verifier form is loaded.

To trigger this event in Designer's Verifier Test Mode or Verifier Train Mode, enable "Allow firing of VerifierFormLoad event when in Verifier Test/Train Modes" in the Designer settings on the Compatibility tab.

You can use the VerifierFormLoad event for different purposes, including the following examples.

- To switch verification forms between different types of classes
- 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 language different from Windows Region and Language settings.
- To load the required verification form according to the Windows Region and Language settings of the current system.
- To default Verifier to display a specific page of a document instead of the first one.
- To modify the form before it displays to the user.

Syntax

```
ScriptModule_VerifierFormLoad (pWorkdoc as ISCBCdrWorkdoc, FormName as
String, FormClassName as String)
```

Parameter	Description
pWorkdoc	Reference to the currently processed document.
FormName	Name of the form that Verifier loads.
FormClassName	Name of the class that contains the verification form.

Sample Code

The following code example demonstrate how to replace the standard class name and the form name that Verifier loads.

Note: If the VerifierFormLoad event script handler is not implemented or the script assigns a non-existing class name or a non-existing form name, the application loads the standard verification form.

```
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
```

See also

- Implement an event
- DisplayPage
- Invisible

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.

The Cedar DocClass event interface provides the following events.

FocusChanged

This event triggers each time before the focus inside the verification form changes. You can modify the focus change by modifying the `pNewFieldIndex` parameter. You can write a different field index into that parameter, which causes 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 dialog box, on the Compatibility tab.

Syntax

```
Document_FocusChanged (pWorkdoc as ISCBCdrWorkdoc, Reason as
CdrFocusChangeReason, OldFieldIndex as Long, pNewFieldIndex as Long)
```

Description	Definition
pWorkdoc	Reference to the currently displayed workdoc.
Reason	Reason of the current focus change, which can be Tab key, Enter key, mouse click, or initial loading. **Possible values** CdrFocusChangeReason
OldFieldIndex	Index of the currently selected field. In case of initial loading this -1.
pNewFieldIndex	Index of the currently selected field. This parameter is modified during the script event to keep the focus in the previous field or set it to another field.

Sample Code

The following script example demonstrates how to skip the table data validation in Verifier for a table with 2 columns.

```
Private Sub Document_FocusChanged(pWorkdoc as SCBCdrWorkdoc, Reason as
CdrFocusChangeReason, OldFieldIndex as Long, pNewFieldIndex as Long) Dim
theEmptyTable as SCBCdrPROJLib.SCBCdrTable Dim theEmptyTableField as
SCBCdrPROJLib.SCBCdrField 'Initializes table and field references Set
theEmptyTable = _ pWorkdoc.Fields("EmptyTable").Table(pWorkdoc.Fields
("EmptyTable").ActiveTableIndex) 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
```

Sample code

To trigger the event before the assigned verification form loads, but after the VerificationFormLoad event, enable the "Allow firing of FocusChanged event when loading the verification form" in Designer. For more information, see "Specify compatibility settings" in the *Oracle WebCenter Forms Recognition Designer User's Guide*.

The following sample code shows how to implement the handler for the `CdrBeforeFormLoaded` reason.

```
Private Sub Document_FocusChanged(pWorkdoc as SCBCdrPROJLib.SCBCdrWorkdoc,
ByVal Reason as SCBCdrPROJLib.CdrFocusChangeReason, ByVal OldFieldIndex as
Long, pNewFieldIndex as Long) If Reason = CdrBeforeFormLoaded Then ' Do
something... End If End Sub
```

OnAction

This event triggers if any of the configured actions in the Verifier Design Mode were caused by the user clicking a button or pressing any of the configured keyboard short cuts.

Syntax

```
Document_OnAction (pWorkdoc as ISCBCdrWorkdoc, ActionName as String)
```

Definition	Description
pWorkdoc	Reference to the currently displayed workdoc.
ActionName	Name of the action assigned to the pressed button or short cut key.

Sample Code

```
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
```

PostExtract

This event triggers after all defined analysis or evaluation methods execute 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.

Syntax

```
Document_PostExtract (pWorkdoc as ISCBCdrWorkdoc)
```

Description	Definition
pWorkdoc	Current workdoc object

Sample Code

```
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
```

PreExtract

This event triggers before any defined analysis or evaluation method executes by the Cedar DocClass.

Syntax

```
Document_PreExtract (pWorkdoc as ISCBCdrWorkdoc)
```

Parameter	Description
pWorkdoc	Current workdoc object

Sample Code

```
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
```

PreVerifierTrain

The PreVerifierTrain event is added to control SLW training in Verifier, Learnset Manager, and Designer.

This event triggers when a program starts learning for a document in the Supervised Learning Workflow (SLW).

Syntax

```
Document_PreVerifierTrain(pWorkdoc as SCBCdrPROJLib.SCBCdrWorkdoc, pMode
as Long)
```

Parameter	Description
pWorkdoc	The current workdoc
pMode	Reserved for future use, should not be used in the current version.

Sample Code

The following script example demonstrates how the new script event can be used to apply a substitution of the primary Associative Search Engine field with another result referring to a different pool.

```
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
```

Validate

Use the Validate event to perform validation on document level. At this point, the validation of all single fields executes. 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.

Syntax

```
Document_Validate (pWorkdoc as ISCBCdrWorkdoc, pValid as Boolean)
```

Parameter	Description
pWorkdoc	The current workdoc
pValid	Parameter containing the current valid state of the workdoc.

Sample Code

```
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
```

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 trigger an event that adds a document to the local learnset.

Syntax

```
Document_VerifierTrain (pWorkdoc as ISCBCdrWorkdoc, ProposedClassName as
String, WillTrain as Boolean, VerifierReason as CdrLocalTrainingReason,
ScriptReason as String)
```

Parameter	Description
pWorkdoc	Contains the reference to the currently processed document.
ProposedClassName	The proposed class name.
WillTrain	Boolean value for the current learning state. - True: The document will be learned - False: The document will not be learned
VerifierReason	The reason why the document is taken for training or why it was rejected. **Possible values** CdrLocalTrainingReason
ScriptReason	Contains the reason why the script routine decided to reject or accept the document.

Cedar [FieldName] Event Interface

Field events are specific for each Cedar field of each document class (DocClass). Field events appear within the script page of their DocClass. For example, all events for the field **Number** of the document class **Invoice** must be implemented within the script page of the document class **Invoice**.

Within the script, the field names are used as specifier for the event.

Example: The **Validate** event for the field **Number** is named `Number_Validate`.

Note: During this documentation, [FieldName] is used as a placeholder for the name of the field, such as `[FieldName]_Validate`.

The Cedar [FieldName] event interface provides the following events.

CellChecked

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

Syntax

```
<Fieldn>_CellChecked (pTable as ISCBCdrTable, pWorkdoc as ISCBCdrWorkdoc,
Row as Long, Column as Long, Checked as Boolean)
```

Parameter	Description
pTable	Current table object.
pWorkdoc	Current workdoc object.
Row	This parameter contains the index of the current row on which the user clicked.
Column	This parameter contains the index of the current column on which the user clicked.
Checked	True: The cell is checked. False: The cell is not checked.

Sample Code

```
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
```

CellFocusChanged

This event triggers each time the focus inside the verification table changes.

Syntax

```
[FieldName]_CellFocusChanged (pTable as ISCBCdrTable, pWorkdoc as
ISCBCdrWorkdoc, Reason as CdrTableFocusChangeReason, OldRow as Long,
OldColumn as Long, pNewRow as Long, pNewColumn as Long)
```

Parameter	Description
pTable	Current table object.
pWorkdoc	Current workdoc object.
Reason	Reason for the focus change. **Possible values** CdrTableFocusChangeReason
OldRow	This parameter contains the index of the derivation row.
OldColumn	This parameter contains the index of the derivation column.

pNewRow	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.
pNewColumn	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.

Sample Code

```
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,
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 sets the 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
program 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
```

Format

This event is used to reformat the content of a field, for example to unify a date or amount format or removing prefixes and suffixes. This event prepares the field data for validation.

Note: The content of `pField.Text` is typically used for learning within the Scripting Guide engines. To change the output format for the field content, use the FormatForExport script event.

Syntax

```
[FieldName]_Format (pField as ISCBCdrField)
```

Parameter	Description
pField	Field object

Sample Code

```
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
```

FormatForExport

Use this event to reformat the content of a field, for example to unify a date or amount format or to remove prefixes and suffixes.

Formatting information can be kept within `pField.FormattedText` rather than `pField.Text`. In this way, the original `pField.Text` is used for learning within the Scripting Guide engines. The formatted `pField.FormattedText` can be used via scripting for export.

Syntax

```
[FieldName]_FormatForExport (pField as ISCBCdrField)
```

Parameter	Description
pField	Current field.

Sample Code

```
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. Then
use pField.FormattedText also for the Export, instead of pField.Text
pField.FormattedText = NewAmount End if End Sub
```

PostAnalysis

This event triggers after the analysis step performs. It is possible to examine the list of all candidates and to add further candidates to the field.

Syntax

```
<Fieldn>_PostAnalysis (pField as ISCBCdrField, pWorkdoc as ISCBCdrWorkdoc)
```

Parameter	Description

pField	The object containing the field.
pWorkdoc	The current workdoc object.

Sample Code

```
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
```

PostEvaluate

This event triggers after the evaluation step performs. It is possible to examine the list of all candidates and to change their weights.

Syntax

```
<Fieldn>_PostEvaluate (pField as ISCBCdrField, pWorkdoc as ISCBCdrWorkdoc)
```

Parameter	Description
pField	Object containing the field
pWorkdoc	Current workdoc object

Sample Code

```
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
```

PreExtract

This event triggers before any defined analysis or evaluation method for this field executes by the Cedar DocClass.

Syntax

```
[FieldName]_PreExtract (pField as ISCBCdrField, pWorkdoc as
ISCBCdrWorkdoc)
```

Parameter	Description
pField	Object containing the field

pWorkdoc	Current workdoc object

Sample Code

```
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
```

SmartIndex

This event triggers for the field where the smart indexing is defined. This field typically provides the key for the select statement.

Syntax

```
<Fieldn>_SmartIndex (pField as ISCBCdrField, pWorkdoc as ISCBCdrWorkdoc)
```

Parameter	Description
pField	Object containing the current field
pWorkdoc	Current workdoc object

Sample Code

```
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
```

TableHeaderClicked

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

Syntax

```
[FieldName]_TableHeaderClicked (pTable as ISCBCdrTable, pWorkdoc as
ISCBCdrWorkdoc, ClickType as CdrTableHeaderClickType, Row as Long, Column
as Long, pSkipDefaultHandler as Boolean)
```

Parameter	Description
pTable	The current table object.
pWorkdoc	The current workdoc object.

ClickType	The click type of the mouse. **Possible values** CdrTableHeaderClickType
Row	This parameter contains the index of the current row on which the user clicked.
Column	This parameter contains the index of the current column on which the user clicked.
pSkipDefaultHandler	The default value is False. When the user wants to skip the default handling it has to be set to True.

Sample Code

```
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
```

Validate

Use this event to perform project specific validation rules. Use the pValid parameter to return the validation decision. If the parameter remains unchanged or if the event is not implemented, the document state gets valid if all fields are valid.

Syntax

```
<Fieldn>_Validate (pField as ISCBCdrField, pWorkdoc as ISCBCdrWorkdoc,
pValid as Boolean)
```

Parameter	Description

pField	Object containing the current field
pWorkdoc	Current workdoc object
pValid	Parameter containing the current valid state of the field

Sample Code

```
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
```

ValidateCell

This event method triggers for each cell of the table. You can implement validation checks specific for a single cell.

Syntax

```
<Fieldn>_ValidateCell (pTable as ISCBCdrTable, pWorkdoc as ISCBCdrWorkdoc,
Row as Long, Column as Long, pValid as Boolean)
```

Parameter	**Description**
pTable	The current table object.
pWorkdoc	The current workdoc object.
Row	The row of the table.
Column	The column of the table.
pValid	The current valid state of the table cell.

Sample Code

```
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
```

ValidateRow

Use this event to implement validation rules, which combine two or more cells of a row.

Syntax

```
<Fieldn>_ValidateRow (pTable as ISCBCdrTable, pWorkdoc as ISCBCdrWorkdoc,
Row as Long, pValid as Boolean)
```

Parameter	Description
pTable	Table object for which row is to be validated.
pWorkdoc	Current workdoc object.
Row	Row of the table to validate.
pValid	Parameter that contains the current valid state of the row.

Sample Code

```
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
```

ValidateTable

Use this event to implement a validation rule for the entire table.

Syntax

```
<Fieldn>_ValidateTable (pTable as ISCBCdrTable, pWorkdoc as
ISCBCdrWorkdoc, pValid as Boolean)
```

Parameter	Description
pTable	Table object
pWorkdoc	Current workdoc object.
pValid	Parameter containing the current valid state of the table.

Sample Code

```
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 pTabler.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
```

Working with Workdoc Objects (SCBCdrWKDOCLib)

The workdoc object reference library SCBCdrWKDOCLib provides the following objects.

- SCBCdrCandidate
- SCBCdrDocPage
- SCBCdrField
- SCBCdrFields
- SCBCdrFolder
- SCBCdrTable
- SCBCdrTextblock
- SCBCdrWord
- SCBCdrWorkdoc

SCBCdrWKDOCLib Type Definitions

The SCBCdrWKDOCLib object provides the following type definitions.

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.

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

CDRDocFileType

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

Type	Description
CDRDocFileTypeCroCIDoc	Cairo CIDocument
CDRDocFileTypeCroImage	Cairo image object
CDRDocFileTypeRawText	Created from plain text without document
CDRDocFileTypeUnknown	Unknown file type, maybe attachment

CDRDocState

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

Type	Description
CDRDocStateAnalyzed	Document is analyzed
CDRDocStateBlocks	Blocks are analyzed in document
CDRDocStateClassified	Document is classified
CDRDocStateDeleted	Document is deleted
CDRDocStateEvaluated	Document is evaluated
CDRDocStateExported	Document is exported
CDRDocStateHaveDocs	Images or CI docs are assigned to documents
CDRDocStateLanguage	Language detection executed
CDRDocStateReset	Initial state of document
CDRDocStateValid	Validity state of document
CDRDocStateWorktext	Worktext is assigned to document

CdrEdgeSide

This type determines the type of alignment or edges.

Type	Description

CDREdgeLeft	Chooses left alignment (left edges) in analysis.
CDREdgeRight	Chooses right alignment (right edges) in analysis.

CdrExportType

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

Type	Description
CDRExportTypeOCRData	Exports OCR data of words and characters of a workdoc.

CDRFieldState

This enumeration contains the state of the field.

Type	Description
CDRFieldStateAnalyzed	Field is analyzed
DRFieldStateEvaluated	Field is evaluated
CDRFieldStateFormated	Field is formatted
CDRFieldStateReset	Initial state of a field
CDRFieldStateValid	Validity state of field

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.

Type	Description
CDRHighlightAttractors	Attractor highlighting
CDRHighlightBlocks	Block highlighting
CDRHighlightCandidates	Candidates highlighting
CDRHighlightCandidatesAdvanced	Highlights only candidates but according to their advanced highlighting type, also triggers all mouse events for all words
CDRHighlightCheckedWords	Verified words highlighting

CDRHighlightCheckedWordsAndCandidates	Verified words and candidate highlighting
CDRHighlightCheckedWordsAndField	Verified words and field highlighting
CDRHighlightCheckedWordsAndFields	Verified words and fields highlighting
CDRHighlightFields	Fields highlighting
CDRHighlightNothing	No highlighting
CDRHighlightParagraphs	Paragraph highlighting
CDRHighlightRectangles	Variable rectangle highlighting
CDRHighlightTables	Table highlighting
CDRHighlightTablesAdvanced	Highlights checked words and selected table cell, also shows tool-tips for all words and triggers all mouse events for all words
CDRHighlightTextLines	Text lines highlighting
CDRHighlightTextLinesAdvanced	Highlights text lines according their block number, show tool-tips with line confidences, also triggers all mouse events
CDRHighlightTrainedFields	Trained fields highlighting
CDRHighlightVerticalEdgesLeft	Left aligned edges highlighting
CDRHighlightVerticalEdgesRight	Right aligned edges highlighting
CDRHighlightWords	Word highlighting

CDRLocation

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

Type	Description
CDRLocationBottom	Bottom corner coordinate
CDRLocationLeft	Left corner coordinate
CDRLocationRight	Right corner coordinate
CDRLocationTop	Top corner coordinate

CDRPageAssignment

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

Type	Description
CDRPageAssignAllPages	Assign all DocPages of Image or CIDoc to workdoc
CDRPageAssignNewPage	First Page of Image or CIDoc appended as last DocPage to workdoc
CDRPageAssignNoPage	No DocPages assigned to workdoc

CDRPageSource

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

Type	Descriptions
CDRPageSourceFrontPage	Front page assigned to workdoc.
CDRPageSourceRearPage	Rear page assigned to workdoc.
CDRPageSourceUnknown	Assigned page to workdoc is unknown.

CDRPDFExportStyle

This data type specifies the type used to export the page to the PDF.

Type	Description
CDRPDF_ImgOnly	Export the page as "image only" to the PDF.
CDRPDF_ImgOnTxt	Default value Export the page with both image and text to the PDF.
CDRPDF_NoExport	Do not export this page to the PDF.
CDRPDF_NoThumbnails	This type is obsolete. Since version 5.9.1, all pages are exported without thumbnails.
CDRPDF_TxtOnly	Export the page as "text only" to the PDF.

CDRTableHighlightMode

This lists the enumerations that contain the highlighting mode of a table.

Type	Description
CDRTableHighlightAllCells	Highlight all cells of a table
CDRTableHighlightAllColumns	Highlight all columns of a table
CDRTableHighlightAllColumnsAdvanced	Advanced highlighting mode for both mapped and unmapped columns
CDRTableHighlightAllRows	Highlight all rows of a table
CDRTableHighlightCell	Highlight particular cell (as set by HighlightColumnIndex and HighlightRowIndex)
CDRTableHighlightColumn	Highlight column (as set by HighlightColumnIndex)
CDRTableHighlightNothing	Highlight nothing
CDRTableHighlightRow	Highlight row (as set by HighlightRowIndex)
CDRTableHighlightTable	Highlight whole table

CDRTranslationLanguage

This type defines the transliteration approach.

Type	Description
CDRTranslationLanguageDefault	Internal language translation is turned off.
CDRTranslationLanguageGreek	One to one character translation type representing former non-western languages support approach.
CDRTranslationLanguageRussian	Translation through extended CJKT methods, but leaving all one to many chars unchanged.
CDRTranslationLanguageCJKT	Full CJKT type of language translation with one to many encoding for both CJKT and non-western languages.
CDRTranslationLanguageCombined	Transformation for all Unicode characters.

CroLinesDir

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

Type	Descriptions
CroLinesDir_Horizontal	Horizontal line
CroLinesDir_Vertical	Vertical line

CroLinesKooType

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

Type	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

SCBCdrCandidate

Cedar candidates are generated during the analysis step and are representing possible results of a field.

SCBCdrCandidate Methods

The SCBCdrField object provides the following methods.

- CopyToField
- RemoveAttractor

CopyToField

This method copies all required properties from the candidate to the field result.

Syntax

```
CopyToField (pField as ISCBCdrField)
```

Parameter	Description
pField	Reference to the field containing the candidate. States which field gets the values from the candidate.

RemoveAttractor

This method removes the attractor specified by index.

Syntax

```
RemoveAttractor (AttractorIndex as Long)
```

Parameter	Description
AttractorIndex	Index of attractor to remove. **Possible values** 0 to AttractorCount-1

SCBCdrCandidate Properties

The SCBCdrField object provides the following properties.

Attractor

This read-only property returns the attractor of the candidate by a zero-based index.

Syntax

```
Attractor (index as Long) as ISCBCdrAttractor
```

Parameter	Description
Index	Specifies the index in the attractor array. **Possible values** 0 to AttractorCount - 1.

AttractorCount

This read-only property returns the number of attractors for this candidate.

Syntax

```
AttractorCount as Long
```

FilterID

This read-only property returns FilterID value as specified by the AddCandidate method of the field.

Syntax

```
FilterID as Long
```

Sample Code

```
Dim intNewCandidate as long Dim lngUniqueID as Long lngUniqueID =
pWorkdoc.Fields("VendorASSA").Candidate(intNewCandidate).FilterID
pWorkdoc.Fields("VendorASSA").PutUniqueEntryId(0, lngUniqueID)
```

FormatConfidence

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

Syntax

```
FormatConfidence as Double
```

Height

This read-only property returns the height of the candidate in pixels.

Syntax

```
Height as Long
```

KeepSpaces

This property sets or returns if the text created from several Words keeps the spaces between these Words.

Syntax

```
KeepSpaces as Boolean
```

Left

This read-only property returns the left border of the candidate in pixels.

Syntax

```
Left as Long
```

Line

This read-only property returns the text of a single line. A candidate can consist of one or more lines.

Syntax

```
Line (index as Long) as String
```

Parameter	Description
Index	Index of the line. **Possible values** 0 to LineCount-1

LineCaption

If a candidate has more than one line, you can use this property to set or return a caption to each line to provide information about the content of the line.

Syntax

```
LineCaption (index as Long) as String
```

Parameter	Description
Index	Index of the line. **Possible values** 0 to LineCount – 1

LineCount

This property sets or returns the number of lines of the candidate or the number of lines of a field.

Syntax

```
LineCount as Long
```

LineWordCount

This read-only property returns the number of words of the specified line.

Syntax

```
LineWordCount (index as Long) as Long
```

Parameter	Description
Index	Index of the line.

LineWordID

This read-only property returns the Word ID of the specified line and word index.

Syntax

```
LineWordID (LineIndex as Long, WordIndex as Long) as Long
```

Parameter	Description
LineIndex	Index of the Line **Possible values** 0 to LineCount-1
WordIndex	Index of the Word within the Line.

LineWorktext

This property sets or returns the worktext object of the single line specified by the zero-based index within a multi-line field.

Syntax

```
LineWorktext (index as Long) as ISCBCroWorktext
```

Parameter	Description
Index	Zero-based index of single line

PageNr

This read-only property returns the DocPage number where the candidate is located.

Syntax

```
PageNr as Long
```

Sample Code

```
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
```

Text

This read-only property returns the text of the candidate.

Syntax

```
Text as String
```

Top

This read-only property returns the top border of the candidate in pixels.

Syntax

```
Top as Long
```

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 percent) in case the sum of different single candidate weights resulting from position and environment of the candidate exceeds 100 percent. Candidates with more than 100 percent are also accounted for selection.

Syntax

```
Weight as Double
```

Width

This read-only property returns the width of the candidate in pixels.

Syntax

```
Width as Long
```

WordCount

This read-only property returns the word count of the candidate.

Syntax

```
WordCount as Long
```

WordID

This read-only property returns the word ID of the specified word index within the first line.

Syntax

```
WordID (index as Long) as Long
```

Parameter	Description
Index:	Zero-based index of the Word within the line.

Worktext

This read-only property returns the worktext object of the first line.

Syntax

```
Worktext as ISCBCroWorktext
```

SCBCdrDocPage

The SCBCdrDocPage object represents a single DocPage within a workdoc.

SCBCdrDocPage Methods

The SCBCdrDocPage object provides the following methods.

GetResolution

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

Syntax

```
GetResolution (ImageIndex as Long, pXRes as Long, pYRes as Long)
```

Parameter	Description
ImageIndex	Index of the image of the DocPage. **Possible values** 0 to ImageCount - 1.
pXRes	[out] X-resolution after execution of the method.
pYRes	[out] Y-resolution after execution of the method.

Rotate

This method rotates the underlying Images by the specified angle.

Syntax

```
Rotate (angle as Double)
```

Parameter	Description
Angle	Specifies the rotation angle in a range of -180.0 to +180.0.

SCBCdrDocPage Properties

The SCBCdrDocPage object provides the following properties.

DisplayImage

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

Syntax

```
DisplayImage as Long
```

DocIndex

This read-only property specifies the index of the document inside the workdoc to which this DocPage belongs.

Syntax

```
DocIndex (ImageIndex as Long) as Long
```

Parameter	Description
ImageIndex	ImageIndex of the DocPage. **Possible values** 0 to ImageCount-1

See also the following properties of the SCBCdrWordoc object:

- DocFileName
- DocFileType

DocPageIndex

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

Syntax

```
DocPageIndex (ImageIndex as Long) as Long
```

Parameter	Description
ImageIndex	Index of the Image of the DocPage. **Possible values** 0 to ImageCount-1

Height

This read-only property returns the height of the DocPage in millimeter.

Syntax

```
Height as Double
```

Image

This read-only property returns an image object for the specified index of the DocPage.

Syntax

```
Image (index as Long) as ISCBCroImage
```

Parameter	Description
Index	Index of the image of the DocPage. **Possible values** 0 to ImageCount - 1

ImageCount

This read-only property returns the number of images available for the DocPage.

Syntax

```
ImageCount as Long
```

ImportedFileName

This read-only property returns the name of the imported file.

Syntax

```
ImportedFileName as String
```

ImportedFilePageIndex

This read-only property returns the zero-based index of the imported file page.

Syntax

```
ImportedFilePageIndex as Long
```

Line

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

Syntax

```
Line (LineIndex as Long, LineDir as CroLinesDir, KooType as
CroLinesKooType) as Long
```

Parameter	Description
LineIndex	Zero-based index of the Line.
LineDir	Direction of Line (Horizontal or Vertical).
KooType	**Possible values** See CroLinesKooType

LinesCount

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

Syntax

```
LinesCount (LinesDir as CroLinesDir) as Long
```

Parameter	Description
LinesDir	Direction of Line (Horizontal or Vertical).

OriginalDocumentFileName

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

Syntax

```
pWorkdoc.Pages(0).OriginalDocumentFileName
```

Sample Code

```
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
CDRTypeInfo, CDRSeverityLogFileOnly, " Original File Name of Page: " &
CStr(CurPage+1) & " is [" & OrgFilename & "]" Next CurPage End Sub
```

PageSource

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

Syntax

```
PageSource as CDRPageSource
```

See also

CDRPageSource

Rotation

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

Syntax

```
Rotation as Double
```

Text

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

Syntax

```
Text as String
```

Width

This read-only property returns the width of the DocPage in millimeters.

Syntax

```
Width as Double
```

SCBCdrEmailProperties

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 through the "ISCBCdrEmailProperties" interface that can be queried from the SCBCdrWorkdoc interface.

SCBCdrEmailProperties Properties

The SCBCdrEmailProperties object provides the following properties.

CC

List of carbon copy recipients.

Syntax

```
CC as String
```

From

List of email senders.

Syntax

```
From as String
```

MessageID

Unique message identifier.

Syntax

```
MessageID as String
```

Priority

Priority the email was sent with.

Syntax

```
Priority as CDREmailPriority
```

Received

Date and time the email was received.

Syntax

```
Received as Date
```

Sent

Date and time the email was sent.

Syntax

```
Sent as Date
```

Subject

Subject of the email.

Syntax

```
Subject as String
```

To

List of email recipients.

Syntax

```
To as String
```

SCBCdrField

This object contains the evaluated data to extract from the document.

SCBCdrField Methods

The SCBCdrField object provides the following methods.

AddCandidate

This method adds a new candidate to the field based on the specified Word ID.

Syntax

```
AddCandidate (WordNr as Long, WordCount as Long, FilterID as Long, pIndex
as Long)
```

Parameter	Description

WordNr	Specifies the Word index within the Word array of the workdoc. **Possible values** 0 to pWorkdoc.WordCount - 1
WordCount	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.
FilterID	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.
pIndex	[out] Returns the index of the new candidate within the candidate array.

Sample Code

```
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
```

AddCandidate2

This method adds a new candidate to the field based on the specified Worktext.

Syntax

```
AddCandidate2 (pWorktext as ISCBCroWorktext, pIndex as Long)
```

Parameter	Description
pWorktext	[in] Must be an initialized Worktext as it was created calling a SCBCroZone.Recognize method.
pIndex	[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.

Syntax

```
AddTable ()
```

CandidateByFilterID

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

Syntax

```
CandidateByFilterID (ByVal FilterID as Long, ByVal CreateNew as Boolean,
pCandidateIndex as Long) as ISCBCdrCandidate
```

Parameter	Description
FilterID	The filter ID used to find the candidate.
CreateNew	Create a new candidate if set to True.
pCandidateIndex	The index of the found candidate

CandidateCount

This method returns the number of candidates for a field.

Syntax

```
CandidateCount as Long
```

DeleteLine

This method deletes a line from a specific index position.

Syntax

```
DeleteLine (LineIndex as Long)
```

Parameter	Description
LineIndex	Index of Line, zero-based indexing

Sample Code

```
'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"
```

DeleteTable

This method deletes a table from the table array of this field.

Syntax

```
DeleteTable (TableIndex as Long)
```

Parameter	Description
TableIndex	Zero-based index of the table

FindCandidate

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

Syntax

```
FindCandidate (WordID as Long, pCandIndex as Long)
```

Parameter	Description
WordID	[in] Specifies a WordID inside the Word array of the workdoc searched for.
pCandIndex	[out] Contains the index of the candidate if someone was found or -1 if no candidate was found.

FindCandidateByPos

This method finds a candidate by its position.

Syntax

```
FindCandidateByPos (Page as Long, Param1 as Long, Left as Long, Top as
Long, Width as Long, Height as Long, CandidateIndex as Long) as
ISCBCdrCandidate
```

Parameter	Description
Page	Page number
Param1	Parameter 1
Left	Left position
Top	Top position
Width	Width
Height	Height
CandidateIndex	Index of the candidate

GetFirstCandidatePropsByPage

This method returns the first candidate’s properties by page.

Syntax

```
GetFirstCandidatePropsByPage (Page as Long, Param1 as Long, Left as Long,
Top as Long, Width as Long, Height as Long, Text as String, Weight as
Double) as Long
```

Parameter	Description
Page	Page number

Param1	Parameter 1
Left	Left position
Top	Top position
Width	Width
Height	Height
Text	Text
Weight	Weight

GetNextCandidatePropsByPage

This method returns the next candidate’s properties by page.

Syntax

```
GetNextCandidatePropsByPage (Left as Long, Top as Long, Width as Long,
Height as Long, Text as String, Weight as Double) as Long
```

Parameter	**Description**
Left	Left position
Top	Top position
Width	Width
Height	Height
Text	Text
Weight	Weight

GetUniqueEntryID

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

Note: The alphanumeric indexes do not support this method. For these, use the UniqueIDproperty as demonstrated in the sample code below.

Syntax

```
GetUniqueEntryId (IdHigh as Long, IdLow as Long)
```

Parameter	Description
IdHigh	[out] Upper part of the 64-bit unique ID.
IdLow	[out] Lower part of the 64-bit unique ID.

Sample Code

```
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
```

InsertLine

This method inserts a line at the specified line index in a field.

Syntax

```
InsertLine (LineIndex as Long)
```

Parameter	Description
LineIndex	Zero-based line index that specifies the position of the line to insert.

Sample Code

Use the following sample code to insert new lines to a field.

```
'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" does not work.
```

PutUniqueEntryId

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

Note: The alphanumeric indexes do not support this method.

Syntax

```
PutUniqueEntryId (IdHigh as Long, IdLow as Long)
```

Parameter	Description
IdHigh	Upper part of the 64-bit unique ID.

IdLow	Lower part of the 64-bit unique ID.

Sample Code

```
Dim intNewCandidate as long Dim lngUniqueID as Long lngUniqueID =
pWorkdoc.Fields("VendorASSA").Candidate(intNewCandidate).FilterID
pWorkdoc.Fields("VendorASSA").PutUniqueEntryId(0, lngUniqueID)
```

RemoveCandidate

This method removes a candidate from the candidate array.

Syntax

```
RemoveCandidate (CandIndex as Long)
```

Parameter	Description
CandIndex	Zero-based candidate Index.

SortCandidatesByWeight

This method sorts evaluated field candidates by their weight.

Syntax

```
SortCandidatesByWeight(Ascending as Boolean)
```

Parameter	Description
Ascending	True: Sort the candidates ascending, that means by descending weight value. False: Sort the candidates descending, that means by ascending weight value.

SCBCdrField Properties

The SCBCdrField object provides the following properties.

ActiveTableIndex

This property sets or returns the position where the table is activated or activates the table at given zero-based index.

Syntax

```
ActiveTableIndex as Long
```

Sample Code

```
'Initializes table and field references Set theEmptyTable = _
pWorkdoc.Fields("EmptyTable").Table(pWorkdoc.Fields
("EmptyTable").ActiveTableIndex) Set theEmptyTableField = pWorkdoc.Fields
("EmptyTable")
```

BoostDigitsOnly

This property sets or returns if ’Boost digits only’ is enabled.

Syntax

```
BoostDigitsOnly as Boolean
```

BoostField

This property sets or returns if ’Boost field’ is enabled.

Syntax

```
BoostField as Boolean
```

Candidate

This read-only property returns a candidate of the field.

Syntax

```
Candidate (index as Long) as ISCBCdrCandidate
```

Parameter	Description
Index	Index of the candidate. **Possible values** 0 to candidateCount-1

Changed

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

Syntax

```
Changed as Boolean
```

CustomDetailsString

This property sets or returns the CustomDetailsString.

Syntax

```
CustomDetailsString as String
```

CustomStatusLong

This property sets or returns the CustomStatusLong.

Syntax

```
CustomStatusLong as Long
```

ErrorDescription

This property sets or returns the reason if a script validation could not be performed.

Syntax

```
ErrorDescription as String
```

Sample Code

```
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
```

ExternalText

This property sets or returns the extended text.

Syntax

```
ExternalText as String
```

FieldID

This read-only property returns the internally used FieldID.

Syntax

```
FieldID as Long
```

FieldState

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

Syntax

```
FieldState as CDRFieldState
```

Sample Code

```
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
```

See also

CDRFieldState

FieldVersion

This property returns the field data of the specified version.

Syntax

```
FieldVersion as String (ByVal index as Long)
```

Parameter	Description
Index	ByVal index as Long

FormattedText

This property is set during field validation if a output format has been defined for the field.

The value can be accessed in later events such as the FormatForExport field event.

For example, to apply the formatting to the exported field value, the `pField.Text` can be overwritten with the `FormattedText` value.

Height

This property sets or returns the height of the field in pixels.

Syntax

```
Height as Long
```

Sample Code

```
'copy the positional information to the new object pCopyField.Height =
pField.Height
```

IsIDAlphNum

This property sets or returns whether an Associative Search Engine's (ASE) unique ID is alphanumeric. If TRUE, then the field is alphanumeric, if FALSE then 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 configurations, 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 (for example alphanumeric) but the same field on a Child Class if of Type B (for example numeric). In this instance accessing the workdoc.IsIDAlphNum always returns 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 or set.

Syntax

```
IsIDAlphNum as Boolean
```

Sample Code

```
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
```

LastModificationEndDate

This property sets or returns the LastModificationEndDate.

Syntax

```
LastModificationEndDate as Date
```

LastModificationEndDateAsFileTimeUtc

This property sets or returns the LastModificationEndDateAsFileTimeUtc.

Syntax

```
LastModificationEndDateAsFileTimeUtcAs  Date
```

Left

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

Syntax

```
Left as Long
```

Line

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

Syntax

```
Line (index as Long) as String
```

Parameter	Description
Index	Index of the line. **Possible values** 0 to LineCount-1

LineCaption

This property sets or returns the LineCaption setting.

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.

Syntax

```
LineCaption (index as Long) as String
```

Parameter	Description
Index	Index of the line **Possible values** 0 to LineCount-1

LineCount

This property sets or returns the number of lines of a multi-line header field. This equals the number of Worktext objects.

Note: Each line of a multi-line header field is represented by a separate individual Worktext object.

Syntax

```
LineCount as Long
```

LineWorktext

This property sets or returns the worktext of each single line of the field. The line index corresponds to the worktext object.

Syntax

```
LineWorktext (LineIndex as Long) as ISCBCroWorktext
```

Parameter	Description
LineIndex	Index of the line. **Possible values** 0 to LineCount-1

MultilineText

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

Syntax

```
MultilineText as String
```

Name

This read-only property returns the name of the field as defined within the design environment.

Syntax

```
Name as String
```

PageNr

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

Syntax

```
PageNr as Long
```

SkipTrainingWithEngine

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

Syntax

```
SkipTrainingWithEngine (bstrEngineName as String) as Boolean
```

Parameter	Description
bstrEngineName	Name of the extraction engine.

Table

This read-only property returns the table object from an array of tables of this field at a specified index.

Syntax

```
Table (index as Long) as ISCBCdrTable
```

Parameter	Description
Index	Position of a table in an array of tables, zero-based indexing.

TableCount

This read-only property returns the number of tables according to the field.

Syntax

```
TableCount as Long
```

Tag

This property sets or returns an arbitrary variant in the field.

Syntax

```
Tag as Variant
```

Text

This property sets or returns 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.

Syntax

```
Text as String
```

Top

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

Syntax

```
Top as Long
```

TrainedWithEngine

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

Syntax

```
TrainedWithEngine (bstrEngineName as String) as Boolean
```

Parameter	Description
bstrEngineName	Name of the engine

UniqueId

This property sets or returns the unique alphanumeric id for the field content from the associative search pool.

Syntax

```
UniqueId as String
```

Value

This property sets or returns the content of the field.

Syntax

```
Value as Variant
```

Valid

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

Syntax

```
Valid as Boolean
```

Width

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

Syntax

```
Width as Long
```

Worktext

This property sets or returns 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.

Note: Each line of a multi-line field is treated as a separate worktext. To access subsequent lines, use `LineWorktext(LineIndex).`

Syntax

```
Worktext as ISCBCroWorktext
```

See also

- Line
- LineCount
- LineWorktext
- MultilineText

XmlExportEnabled

This property sets or returns whether the field is included in the exported XML file.

Set the property to False to exclude the field from the exported file.

Note: The `FieldCount` attribute of the XML `Fields` element contains the total number of fields in the workdoc, not the number of exported fields.

The default value is `True`.

Syntax

```
XmlExportEnabled as Boolean
```

Sample Code

The following sample code disables the XML export of field data for field names that start with "tmp".

```
Dim i As Long Dim currentField As ISCBCdrField For i = 1 To
pWorkdoc.Fields.Count Set currentField = pWorkdoc.Fields.ItemByIndex(i) If
Left$(currentField.Name,3) = "tmp" Then currentField.XmlExportEnabled =
False End If Next i
```

See also

- ExportDocumentToXml
- "Export to XML" in the *Oracle WebCenter Forms Recognition Runtime Server User's Guide*.

SCBCdrFields

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

SCBCdrFields Methods

The SCBCdrFields object provides the following methods.

Add

This method adds a new field with the specified name to the field collection.

Syntax

```
Add (NewItem as ISCBCdrField, ItemName as String) as Long
```

Parameter	Description
NewItem	Pointer to a SCBCdrField object to be added to the collection.
ItemName	Name of the field item inside the collection. Use this name to access the item inside the collection.

Clear

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

Syntax

```
Clear ()
```

ItemExists

This method returns TRUE if an item with the specified name exists inside the collection.

Syntax

```
ItemExists (Name as String) as Boolean
```

Parameter	Description
Name	Name of item to search for.

MoveItem

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

Syntax

```
MoveItem (OldIndex as Long, NewIndex as Long)
```

Parameter	Description

OldIndex	Index of the item to move. **Possible values** 1 to Count
NewIndex	New index of the item after moving it. **Possible values** 1 to Count

Remove

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

Syntax

```
Remove (ItemName as String)
```

Parameter	Description
ItemName	[in] Name of item to remove.

RemoveByIndex

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

Syntax

```
RemoveByIndex (Index as Long)
```

Parameter	Description
Index	Index of the item to remove. **Possible values** 1 to Count

Rename

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

Syntax

```
Rename (OldName as String, NewName as String)
```

Parameter	Description
OldName	[in] Name of item to rename
NewName	[in] New name of item in collection.

SCBCdrFields Properties

The SCBCdrFields object provides the following properties.

Collection

This read-only property returns the collection used internally to store the fields.

Syntax

```
Collection as ISCBCroCollection
```

Count

This read-only property returns the number of items within the field collection.

Syntax

```
Count as Long
```

Item

These read-only property returns a specified item from the collection. The Item property is the default property of the ISCBCdrFields collection.

Syntax

```
Item (Index as Variant) as ISCBCdrField
```

Parameter	Description
Index	The index can either be a long value specifying the index within the collection, or a string specifying the item by name. **Possible values** • 1 to Count • Item name

ItemByIndex

This read-only property returns an item from the collection specified by the index.

Syntax

```
ItemByIndex (Index as Long) as ISCBCdrField
```

Parameter	Description

Index	Index of the item to get from the collection. **Possible values** 1 to Count

Sample Code

```
strClassName = theProject.AllClasses.ItemByIndex(intClass).Name
```

ItemByName

This read-only property returns the field from the collection by the specified field name.

Syntax

```
ItemByName (Name as String) as ISCBCdrField
```

Parameter	Description
Name	Name of the item to get from the collection.

Sample Code

```
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
```

See also

- AllowInteractiveExtraction

ItemIndex

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

Syntax

```
ItemIndex (Name as String) as Long
```

Parameter	Description
Name	Name specifying an item in the collection.

ItemName

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

Syntax

```
ItemName (Index as Long) as String
```

Parameter	Description
Index	Index specifying an item in the collection. **Possible values** 1 to Count

Tag

This property sets or returns a variant for each item of the collection.

Syntax

```
Tag (Index as Long) as Variant
```

Parameter	Description
Index	Specifies the item index. **Possible values** 1 to Count

SCBCdrFolder

A folder can represent an array of workdocs within a batch. A folder can contain one or more workdocs. During classification and extraction, you can access all workdocs of the same folder from within a script.

SCBCdrFolder Methods

The SCBCdrDocPage object provides the following methods.

AddDocument

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

Syntax

```
AddDocument (pWorkdoc as ISCBCdrWorkdoc, pNewIndex as Long)
```

Parameter	Description

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.

Syntax

```
Clear ()
```

InsertDocument

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

Syntax

```
InsertDocument (Index as Long, pWorkdoc as ISCBCdrWorkdoc)
```

Parameter	Description
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.

Syntax

```
MoveDocument (FromIndex as Long, ToIndex as Long)
```

Parameter	Description
FromIndex	Zero-based Index from where workdoc is moved
ToIndex	Zero-based index where workdoc is placed

RemoveDocument

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

Syntax

```
RemoveDocument (index as Long)
```

Parameter	Description

Index	Zero-based index in a Folder from where workdoc is removed

SCBCdrFolder Properties

The SCBCdrDocPage object provides the following properties.

Document

This read-only property returns a workdoc from the specified index of the document array of the Folder.

Syntax

```
Document (Index as Long) as ISCBCdrWorkdoc
```

Parameter	Description
Index	The index of the workdoc within the Folder. **Possible values** 0 to DocumentCount-1

DocumentCount

This read-only property returns the number of workdocs within the Folder.

Syntax

```
DocumentCount as Long
```

FolderData

This property sets or returns a variable number of strings using any string as an index key.

Syntax

```
FolderData (Index as String) as String
```

Parameter	Description
Index	Any non-empty string used as an index key.

Sample Code

```
'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
```

SCBCdrTable

The Cedar table object represents a logical table in a document assigned to a Cedar field of a workdoc.

SCBCdrTable Methods

The SCBCdrField object provides the following methods and properties.

AddColumn

This method adds a new column to a table and returns the zero-based index of the new column.

Syntax

```
AddColumn (ColumnName as String) as Long
```

Parameter	Description
ColumnName	Name of column

Return value

Zero-based index of the new column.

AddRow

This method adds a new row to a table and returns the zero-based index of the new row.

Syntax

```
AddRow() as Long
```

Return value

The zero-based index of the new row.

AddUMColumn

This method adds a new unmapped column to a table and returns the index of the new unmapped column.

Syntax

```
AddUMColumn (pUMColumnIndex as Long)
```

Parameter	Description
pUMColumnIndex	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.

Syntax

```
AppendRows (top as Long, height as Long, PageNumber as Long)
```

Parameter	Description
Top	Top of region used for creation or new rows
Height	Height of region used for creation or new rows
PageNumber	Page number of region

Clear

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

Syntax

```
Clear ()
```

ClearColumn

This method clears the content of an existing column.

Syntax

```
ClearColumn (Column as Variant)
```

Parameter	Description
Column	Zero-based index or name of column

ClearRow

This method clears the content of an existing row.

Syntax

```
ClearRow (RowIndex as Long)
```

Parameter	Description
RowIndex	Zero-based index of row.

ClearUMColumn

This method clears the content of an unmapped column.

Syntax

```
ClearUMColumn (UMColumnIndex as Long)
```

Parameter	Description
UMColumnIndex	Zero-based index of the unmapped column to clear.

DeleteColumn

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

Syntax

```
DeleteColumn (Column as Variant)
```

Parameter	Description
Column	Zero-based index or name of the column.

DeleteRow

This method deletes a row specified by an index.

Syntax

```
DeleteRow (RowIndex as Long)
```

Parameter	Description
RowIndex	Zero-based index of the row.

DeleteUMColumn

This method deletes an unmapped column specified by index.

Syntax

```
DeleteUMColumn (UMColumnIndex as Long)
```

Parameter	Description
UMColumnIndex	Zero-based index of the unmapped column to delete.

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.

Syntax

```
FillColumn (left as Long, top as Long, width as Long, height as Long,
  PageNumber as Long, Column as Variant)
```

Parameter	Description
Left	Left position of area in pixel
Top	Top of area in pixel
Width	Width of area in pixel
Height	Height of area in pixel
PageNumber	DocPage number of the area
Column	Zero-based index or name of the destination column

InsertColumn

This method inserts a new column after specified ColumnIndex.

Syntax

```
InsertColumn (ColumnIndex as Long, ColumnName as String)
```

Parameter	Description
ColumnIndex	Zero-based index of existing column, after which the new column is inserted.
ColumnName	Name of new column

InsertRow

This method inserts a new row after the specified RowIndex.

Syntax

```
InsertRow (RowIndex as Long)
```

Parameter	Description
RowIndex	Zero-based index of existing row, after which the new row inserts.

InsertUMColumn

This method inserts a new, unmapped column.

Syntax

```
InsertUMColumn (UMColumnIndex as Long)
```

Parameter	Description
UMColumnIndex	Zero-based index of new column.

MapColumn

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

Syntax

```
MapColumn (UMColumnIndex as Long, Column as Variant)
```

Parameter	Description
UMColumnIndex	Zero-based index of unmapped source column
Column	Zero-based index or name of destination column

MergeRows

This method merges two rows specified by two indices.

Syntax

```
MergeRows (RowIndex1 as Long, RowIndex2 as Long)
```

Parameter	Description
RowIndex1	Zero-based index of row 1
RowIndex2	Zero-based index of row 2

RemoveAllColumns

This method removes all mapped table columns.

Syntax

```
RemoveAllColumns ()
```

RemoveAllRows

This method removes all table rows.

Syntax

```
RemoveAllRows ()
```

RemoveAllUMColumns

This method removes all unmapped table columns.

Syntax

```
RemoveAllUMColumns ()
```

SwapColumns

This method swaps two specified columns.

Syntax

```
SwapColumns (ColumnIndex1 as Long, ColumnIndex2 as Long)
```

Parameter	Description
ColumnIndex1	Zero-based index of column 1
ColumnIndex2	Zero-based index of column 2

UnMapColumn

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

Syntax

```
UnMapColumn (Column as Variant) as Long
```

Parameter	Description
Column	Zero-based index or name of source column

SCBCdrTable Properties

The SCBCdrField object provides the following properties.

CellColor

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

Syntax

```
CellColor (IsValid as Boolean) as OLE_COLOR
```

Parameter	Description
IsValid	Flag indicating if color refers to valid or invalid table cells

CellLocation

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

Syntax

```
CellLocation (Column as Variant, RowIndex as Long, Location as
CDRLocation)   As Long
```

Parameter	Description
Column	Zero-based index or name of the column
RowIndex	Zero-based index of the row
Location	Location parameter **Possible values** See CDRLocation

CellText

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

Syntax

```
CellText (Column as Variant, RowIndex as Long) as String
```

Parameter	Description
Column	Zero-based index or name of column
RowIndex	Zero-based index of row

Sample Code

```
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
```

CellValid

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

Syntax

```
CellValid (Column as Variant, RowIndex as Long) as Boolean
```

Parameter	Description
Column	Zero-based index of name of column
RowIndex	Zero-based index of row

Sample Code

```
' Makes table object valid theEmptyTable.CellValid(0,0) = True
theEmptyTable.CellValid(1,0) = True
```

CellValidationErrorDescription

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

Syntax

```
CellValidationErrorDescription (Column as Variant, RowIndex as Long) as
String
```

Parameter	Description
Column	Zero-based index or name of column
RowIndex	Zero-based index of row

Sample Code

```
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
```

CellVisible

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

Syntax

```
CellVisible (Column as Variant, RowIndex as Long) as Boolean
```

Parameter	Description
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.

Syntax

```
CellWorktext (Column as Variant, RowIndex as Long) as ISCBCroWorktext
```

Parameter	Description
Column	Zero-based index or name of column
RowIndex	Zero-based index of row

CellWorktextChanged

This property sets or returns a flag indicating whether the cell's worktext has changed.

Syntax

```
CellWorktextChanged (Column as Variant, RowIndex as Long) as Boolean
```

Parameter	Description
Column	Zero-based index or name of column
RowIndex	Zero-based index of row

ColumnColor

This property sets or returns the color of the column.

Syntax

```
ColumnColor (IsValid as Boolean) as OLE_COLOR
```

Parameter	Description
IsValid	Flag indicating if color refers to valid or invalid columns

ColumnCount

This read-only property returns the number of columns.

Syntax

```
ColumnCount as Long
```

ColumnExportEnable

This property sets or returns whether the column of a table field is included in the exported XML file.

Note that the `ColumnCount` attribute of the XML `Columns` element contains the total number of columns in the table, not the number of exported columns.

The default value is `True`.

Syntax

```
ColumnExportEnable (Column as Variant) as Boolean
```

Parameter	Description
Column	Zero-based index or name of the column.

Sample Code

The following sample code disables the XML export for the table field column names that start with "tmp".

```
Dim i As Long Dim currentTable As ISCBCdrTable Set currentTable =
pWorkdoc.Fields.ItemByName("Table").Table(0) For i = 0 To
currentTable.ColumnCount-1 If Left$(currentTable.ColumnName(i),3) = "tmp"
Then currentTable.ColumnExportEnable(i) = False End If Next i
```

See also

- ExportDocumentToXml
- "Export to XML" in the *Oracle WebCenter Forms Recognition Runtime Server User's Guide*.

ColumnIndex

This read-only property returns the column index for the name of a column.

Syntax

```
ColumnIndex (ColumnName as String) as Long
```

Parameter	Description
ColumnName	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.

Syntax

```
ColumnLabelLocation (Column as Variant, Location as CDRLocation) as Long
```

Parameter	Description
Column	Zero-based index or name of column
Location	Location parameter **Possible values** See CDRLocation

ColumnLabelText

This property sets or returns the column label.

Syntax

```
ColumnLabelText (Column as Variant) as String
```

Parameter	Description
Column	Zero-based index or name of column

ColumnLocation

This property sets or returns the location of the column.

Syntax

```
ColumnLocation (Column as Variant, PageNr as Long, Location as
CDRLocation)As Long
```

Parameter	Description
Column	Zero-based index or name of the column
PageNr	DocPage number
Location	Location parameter **Possible values** See CDRLocation

ColumnMapped

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

Syntax

```
ColumnMapped (Column as Variant) as Boolean
```

Parameter	Description
Column	Zero-based index or name of column

ColumnName

This read-only property returns the name of a column.

Syntax

```
ColumnName (ColumnIndex as Long) as String
```

Parameter	Description
ColumnIndex	Zero-based index of the 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.

Syntax

```
ColumnValid (Column as Variant) as Boolean
```

Parameter	Description
Column	Zero-based index or name of the column.

ColumnVisible

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

Syntax

```
ColumnVisible (Column as Variant) as Boolean
```

Parameter	Description
Column	Zero-based index or name of the column.

Sample Code

```
theTableSettings.ColumnVisible(2) = True 'Set the Column visible to True
to show, False to hide.
```

FieldName

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

Syntax

```
FieldName as String
```

FooterLocation

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

Syntax

```
FooterLocation (Location as CDRLocation) as Long
```

Parameter	Description
Location	Location parameter **Possible values** See CDRLocation

FootPageNr

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

Syntax

```
FooterPageNr as Long
```

FooterText

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

Syntax

```
FooterText as String
```

HeaderLocation

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

Syntax

```
HeaderLocation (Location as CDRLocation) as Long
```

Parameter	Description
Location	Location parameter **Possible values** See CDRLocation

HeaderPageNr

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

Syntax

```
HeaderPageNr as Long
```

HeaderText

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

Syntax

```
HeaderText as String
```

HighlightColumnIndex

This property sets or returns the index of the column to highlight.

Syntax

```
HighlightColumnIndex as Long
```

HighlightRowIndex

This property sets or returns the index of the row to highlight.

Syntax

```
HighlightRowIndex as Long
```

HighlightMode

This property sets or returns the highlighting mode of the table.

Syntax

```
HighlightMode as CDRTableHighlightMode
```

See also

CDRTableHighlightMode

HighlightUMColumnIndex

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

Syntax

```
HighlightUMColumnIndex as Long
```

LocationExplicit

This property sets or returns the LocationExplicit flag.

Syntax

```
LocationExplicit as Boolean
```

RowColor

This property sets or returns the color of the row.

Syntax

```
RowColor (IsValid as Boolean) as OLE_COLOR
```

Parameter	Description
IsValid	Flag indicating if color refers to valid or invalid rows

RowCount

This read-only property returns the number of the rows.

Syntax

```
RowCount as Long
```

RowLocation

This property sets or returns the location of the row.

Syntax

```
RowLocation (RowIndex as Long, Location as CDRLocation) as Long
```

Parameter	Description
RowIndex	Zero-based index of the row.
Location	Location parameter **Possible values** See CDRLocation

RowNumber

This property sets or returns the actual number of row.

Syntax

```
RowNumber (RowIndex as Long) as Long
```

Parameter	Description
RowIndex	Zero-based index of row

Sample Code

```
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
```

RowPageNr

This property sets or returns the DocPage number of a row.

Syntax

```
RowPageNr (RowIndex as Long) as Long
```

Parameter	Description
RowIndex	Zero-based index of row

Significance

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

Syntax

```
Significance (EvalPropIndex as Long) as Double
```

Parameter	Description
EvalPropIndex	Index of evaluation property. **Possible values** - 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)

TableColor

This property sets or returns the color of the table.

Syntax

```
TableColor (IsValid as Boolean) as OLE_COLOR
```

Parameter	Description
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).

Syntax

```
TableFirstPage as Long
```

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).

Syntax

```
TableLastPage as Long
```

TableLocation

This property sets or returns the location of a table.

Syntax

```
TableLocation (PageNr as Long, Location as CDRLocation) as Long
```

Parameter	Description
PageNr	DocPage number
Location	Location parameter **Possible values** See CDRLocation

TableValid

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

Syntax

```
TableValid as Boolean
```

TableValidationErrorDescription

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

Syntax

```
TableValidationErrorDescription as String
```

Sample Code

```
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 pTabler.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
```

Tag

This property sets or returns a tag associated with the table.

Syntax

```
Tag as String
```

TotalSignificance

This property sets or returns the total significance of the table.

Syntax

```
TotalSignificance as Double
```

UMCellColor

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

Syntax

```
UMCellColor as OLE_COLOR
```

UMCellLocation

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

Syntax

```
UMCellLocation (UMColumnIndex as Long, RowIndex as Long, Location as
CDRLocation) as Long
```

Parameter	Description
UMColumnIndex	Zero-based index of unmapped column
RowIndex	Zero-based index of unmapped row
Location	Location parameter **Possible values** See CDRLocation

UMCellText

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

Syntax

```
UMCellText (UMColumnIndex as Long, RowIndex as Long) as String
```

Parameter	Description
UMColumnIndex	Zero-based index of unmapped column
RowIndex	Zero-based index of row

UMCellVisible

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

Syntax

```
UMCellVisible (UMColumnIndex as Long, RowIndex as Long) as Boolean
```

Parameter	Description
UMColumnIndex	Zero-based index of unmapped column
RowIndex	Zero-based index of row

UMCellWorktext

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

Syntax

```
UMCellWorktext (UMColumnIndex as Long, RowIndex as Long) as
ISCBCroWorktext
```

Parameter	Description
UMColumnIndex	Zero-based index of unmapped column
RowIndex	Zero-based index of row

UMColumnColor

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

Syntax

```
UMColumnColor as OLE_COLOR
```

UMColumnCount

This read-only property returns the number of unmapped columns.

Syntax

```
UMColumnCount as Long
```

UMColumnLabelLocation

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

Syntax

```
UMColumnLabelLocation (UMColumnIndex as Long, Location as CDRLocation) as
Long
```

Parameter	Description
UMColumnIndex	Zero-based index of unmapped column
Location	Location parameter **Possible values** See CDRLocation

UMColumnLabelText

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

Syntax

```
UMColumnLabelText (UMColumnIndex as Long) as String
```

Parameter	Description
UMColumnIndex	Zero-based index of unmapped column

UMColumnLocation

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

Syntax

```
UMColumnLocation (UMColumnIndex as Long, PageNr as Long, Location as
CDRLocation) as Long
```

Parameter	Description
UMColumnIndex	Zero-based index of unmapped column
PageNr	DocPage number
Location	Location parameter **Possible values** See CDRLocation

UMColumnVisible

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

Syntax

```
UMColumnVisible (UMColumnIndex as Long) as Boolean
```

Parameter	Description
UMColumnIndex	Zero-based index of unmapped column

WeightingFactor

This property sets or returns a weighting factor for a corresponding evaluation property.

Syntax

```
WeightingFactor (EvalPropIndex as Long) as Double
```

Parameter	Description
EvalPropIndex	Index of evaluation property. **Possible values** • 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)

RowValidationErrorDescription

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

Syntax

```
RowValidationErrorDescription (RowIndex as Long) as String
```

Parameter	Description
RowIndex	Zero-based index of row

Sample Code

```
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
```

RowValid

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

Syntax

```
RowValid (RowIndex as Long) as Boolean
```

Parameter	Description
RowIndex	Zero-based index of row

LabellinePageNr

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

Syntax

```
LabellinePageNr as Long
```

SCBCdrTextblock

This object represents a TextBlock on a document. A TextBlock can contain one or more lines.

SCBCdrTextblock Properties

The SCBCdrTextblock object provides the following properties.

Color

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

Syntax

```
Color as OLE_COLOR
```

Height

This read-only property returns the height of the TextBlock in pixels.

Syntax

```
Height as Long
```

Left

This read-only property returns the left border of the TextBlock in pixels.

Syntax

```
Left as Long
```

PageNr

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

Syntax

```
PageNr as Long
```

Text

This read-only property returns the whole text of the TextBlock.

Syntax

```
Text as String
```

Top

This read-only property returns the top border of the TextBlock in pixels.

Syntax

```
Top as Long
```

Visible

This property sets or returns whether the highlighted rectangle of the TextBlock is visible if the TextBlock highlighting is enabled.

Syntax

```
Visible as Boolean
```

Weight

This read-only property returns the block weight.

Syntax

```
Weight as Double
```

Width

This read-only property returns the width of the TextBlock in pixels.

Syntax

```
Width as Long
```

WordCount

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

Syntax

```
WordCount as Long
```

WordID

Use this read-only property as an index for the Word array of the workdoc.

Syntax

```
WordID (index as Long) as Long
```

Parameter	Description
Index	Index of the word inside the TextBlock. **Possible values** 0 to WordCount -1

SCBCdrWord

This object represents a textual word of a document.

SCBCdrWord Properties

The SCBCdrWord object provides the following properties.

Color

This property sets or returns the color used for highlighting checked words.

Syntax

```
Color as OLE_COLOR
```

Height

This read-only property returns the height of the word in pixels.

Syntax

```
Height as Long
```

Left

This read-only property returns the left border of the word in pixels.

Syntax

```
Left as Long
```

PageNr

This read-only property returns the number of the DocPage where the word is located.

Syntax

```
PageNr as Long
```

StartPos

This read-only property returns the index of the first character of the word inside the worktext attached to the workdoc.

Syntax

```
StartPos as Long
```

Text

This read-only property returns the text of the word.

Syntax

```
Text as String
```

TextLen

This read-only property returns the number of characters of the word.

Syntax

```
TextLen as Long
```

Tooltip

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

Syntax

```
Tooltip as String
```

Top

This read-only property returns the top border of the word in pixels.

Syntax

```
Top as Long
```

Visible

If the word highlighting for checked words is enabled, this property sets or returns if the highlighted rectangle of the word is visible.

Syntax

```
Visible as Boolean
```

Width

This read-only property returns the width of the Word in pixels.

Syntax

```
Width as Long
```

Worktext

This read-only property returns the Worktext object of the Word.

Syntax

```
Worktext as ISCBCroWorktext
```

SCBCdrWorkdoc

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

SCBCdrWorkdoc Methods

The SCBCdrWorkdoc object provides the following methods.

AddDocFile

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

Syntax

```
AddDocFile (Path as String, FileType as CDRDocFileType, Assignment as
CDRPageAssignment)
```

Parameter	Description
FilePath	Path to the file to add.
FileType	File type of the specified file, such as a CIDoc or image. **Possible values** See CDRDocFileType
Assignment	Specifies how DocPages are assigned to the workdoc. **Possible values** See CDRPageAssignment

Sample Code

The following sample code shows how to add a CI-PDF file to the workdoc.

```
pWorkdoc.AddDocFile
("C:\coversheet.pdf",CDRDocFileTypeCroCIDoc,CDRPageAssignNewPage)
```

AddField

This method adds a field to the workdoc.

Syntax

```
AddField (Name as String)
```

Parameter	Description

Name	Contains the name for the new field

Sample Code

The following sample code adds the field "AdditionalField" to the workdoc

```
pWorkdoc.AddField("AdditionalField")
```

AddHighlightRectangle

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

Syntax

```
AddHighlightRectangle (Left as Long, Top as Long, Width as Long,   Height
as Long, PageNr as Long, Color as OLE_COLOR)
```

Parameter	Description
Left	Left of highlight rectangle
Top	Top of highlight rectangle
Width	Width of highlight rectangle
Height	Height of highlight rectangle
PageNr	Document page number of highlight rectangle
Color	Color of highlight rectangle

Sample Code

```
pWorkdoc.AddHighlightRectangle(10,10,100,100,1,vbCyan)
```

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 typically results in one block per line.

Syntax

```
AnalyzeAlignedBlocks (edgeSide as CDREdgeSide, leftAlignTolerance as Long,
XDist as Double, YDist as Double, Join as Boolean, minDistance as Double)
```

Parameter	Description

edgeSide	Determines whether left or right aligned blocks are searched. **Possible values** See CDREdgeSide
leftAlignTolerance	The distance in millimeters that aligned lines might differ. Useful if the document was scanned slightly tilted.
XDist	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, a new block is created.
YDist	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.
Join	Specifies whether overlapping blocks are joined. Set to TRUE if you want to join them.
minDistance	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.

Increasing the distance results in bigger text blocks. Minimizing the distance increases the number of smaller text blocks.

Note: Search strings comprising multiple words 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.

Syntax

```
AnalyzeBlocks (XDist as Double, YDist as Double)
```

Parameter	Description
XDist	Minimum X distance between two TextBlocks
YDist	Minimum Y distance between two TextBlocks

Sample Code

```
pWorkdoc.AnalyzeBlocks(4,4)
```

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.

Syntax

```
AnalyzeEdges (edgeSide as CDREdgeSide, AlignTolerance as Double, YDist as
Double, MinNoOfWords as Long, minDistance as Double, [pageNr as Long =
TRUE])
```

Parameter	Definition
edgeSide	Specifies if edges that contain left or right aligned words are requested. **Possible values** See CDREdgeSide
AlignTolerance	Specifies in millimeters how far the left (right) values of words bounding rectangle can differ in order for it to still be considered aligned.
YDist	Specifies in millimeters how far two words can be apart vertically and still belong to the same edge.
MinNoOfWords	Specifies how many words have to belong to a valid edge. Edges that contain less than MinNoOfWords after analyzing the document are deleted.
minDistance	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.
pageNr	Specifies the page to analyze for edges. Set the value to "-1" to analyze all pages. Optional parameter. The default value is "-1".

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.

Syntax

```
AnalyzeEdges2 (edgeSide as CDREdgeSide, AlignTolerance as Double, YDist as
Double, MinNoOfWords as Long, minDistance as Double, pageNr as Long,
vbCheckedOnly as Boolean)
```

Parameter	Description
edgeSide	Specifies if edges that contain left or right aligned words are requested. **Possible values** See CDREdgeSide
AlignTolerance	This value in millimeters specifies how far the left (right) values of words bounding rectangle can differ in order for it to still be considered aligned.

YDist	Specifies in millimeters how far two words can be apart vertically and still belong to the same edge.
MinNoOfWords	Specifies how many words have to belong to a valid edge. Edges that contain less than MinNoOfWords after analyzing the document are deleted.
minDistance	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.
pageNr	Specifies the page to analyze for edges. Set the value to "-1" to analyze all pages.
vbCheckedOnly	If set to TRUE, the method applies processing for visible text lines only, otherwise this function works exactly like AnalyzeEdges.

AnalyzeParagraphs

Use this method to determine all the paragraphs present in workdoc.

Syntax

```
AnalyzeParagraphs ()
```

AppendWorkdoc

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

Syntax

```
AppendWorkdoc (pWorkdoc as ISCBCdrWorkdoc)
```

Parameter	Description
pWorkdoc	Workdoc to append.

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.

Syntax

```
AssignDocToPage (DocIndex as Long, DocPage as Long, WorkdocPage as Long)
```

Parameter	Description
DocIndex	Zero-based CIDoc or image index
DocPage	Zero-based DocPage inside the image or CIDoc
WorkdocPage	Zero-based DocPage inside the workdoc

Clear

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

Syntax

```
Clear ()
```

ClearHighlightRectangles

This method removes all highlighted rectangles.

Syntax

```
ClearHighlightRectangles ()
```

CreateFromWorktext

This method creates a workdoc from the OCRed text of an image.

Syntax

```
CreateFromWorktext (pWorktext as ISCBCroWorktext)
```

Parameter	Description
pWorktext	The object pointer of the worktext.

DeleteFile

This method deletes all WDC files and the corresponding TIF files of the workdoc.

Syntax

```
DeleteFile (DeleteDocFiles as Boolean)
```

Parameter	Description
DeleteDocFiles	Flag to inform whether to delete files or not

ExportDocumentToXml

This method exports the structure and the field data of the current workdoc to an XML file or MSXML object in a predefined format.

Use the named properties XML_ExportCandidates, XML_ExportWords, and XML_ExportWordChars to configure the export to optionally capture the field candidates, OCR word data and the associated character data.

By default, the method exports all fields and table field columns. Use the XmlExportEnabled and ColumnExportEnable properties to exclude specific fields or table field columns from the XML export.

Whenever possible, the XML element and attribute names correspond to the SCBCdrWorkdoc property names.

An ErrorDescription attribute is only added to an XML element if the corresponding `Valid` attribute is set to false.

Note: The *Components\Tools* directory contains the XML schema file *Workdoc.xsd*, which you can use to validate the exported XML file.

Syntax

```
ExportDocumentToXml(ByVal vTarget As Variant)
```

Parameter	Description
vTarget	**Possible values** • A string which specifies the filename, including path. Any existing file will be overwritten. **Note:** You can only specify existing directories, the method does not create them. • An MSXML 3.0 or MSXML 6.0 object. It is the equivalent of saving the XML file and reparsing it using this object.

Sample Code

The following sample code saves the OCR data, candidates, fields and workdoc structure to an XML file.

```
pWorkdoc.NamedProperty("XML_ExportWords") = True pWorkdoc.NamedProperty
("XML_ExportWordChars") = True pWorkdoc.NamedProperty("XML_
ExportCandidates") = True pWorkdoc.ExportDocumentToXml
("C:\ExistingFolder\" & pWorkdoc.Filename & ".xml")
```

Sample Code

The following sample code saves the XML data to an MSXML2.DOMDocument60 object instead of a file.

```
' Note: Add reference to Microsoft XML, version 6.0 in the script page Dim
xmlDoc60 As MSXML2.DOMDocument60 Set xmlDoc60 = New MSXML2.DOMDocument60
pWorkdoc.ExportDocumentToXml(xmlDoc60) ' Change xmlDoc60 here
xmlDoc60.documentElement.appendChild(xmlDoc60.createElement("NewNode")) '
... xmlDoc60.Save("xmlDoc60.xml") Set xmlDoc60 = Nothing
```

XML element definitions

Section	Description
DocClass	Contains document class information, such as class name, parent class and classification results.
DocFiles	Contains the document file structure, such as name and type (CI or Image document.)
DocPages	Contains the document page information, such as size and applied rotation.

Words	Contains information about the single words in the document, such as word text, page number, and position of the word in pixels.
Characters	For a word, contains information about the single characters that compose it, such as character code and position in pixels. **Note:** For CI documents, the reported position and confidence values are those for the word.
Fields	Contains the workdoc field information, such as name, extracted text, text position and validity.
Candidates	For a field, contains all candidate information, such as text, weight and position.

Sample XML

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <Workdoc XML_
version="2.0" FileName="01English_US01_STP.wdc"> <DocClass
DocClassName="Invoices"> <ParentDocClass DocClassName="Invoices"/>
<ClsDocClass ID="1" ClsDocClassName="Invoices" Res="4" Confidence="0"/>
<ClsDocClass ID="2" ClsDocClassName="Generic" Res="1" Confidence="1"/> ...
</DocClass> <DocFiles DocFileCount="1"> <DocFile ID="0"
DocFileName="C:\...\01English_US01_STP.tif"
DocFileType="CDRDocFileTypeCroImage"/> </DocFiles> <DocPages
DocPageCount="3"> <DocPage PageNr="0" DocIndex="0" DocPageIndex="0"
Width="2464" Height="3508" XRes="300" YRes="300" Rotation="0"
ImportedFileName="00000478.tif" ImportedFilePageIndex="0"/> <DocPage
PageNr="1" DocIndex="0" DocPageIndex="1" Width="2464" Height="3508"
XRes="300" YRes="300" Rotation="0" ImportedFileName="00000478.tif"
ImportedFilePageIndex="1"/> <DocPage PageNr="2" DocIndex="0"
DocPageIndex="2" Width="2464" Height="3508" XRes="300" YRes="300"
Rotation="0" ImportedFileName="00000562.tif" ImportedFilePageIndex="0"/>
</DocPages> <Lines LineCount="54"/> <Words WordCount="359"> ... <Word
ID="3" Page="0" Line="2" Left="1496" Top="149" Width="67" Height="22">
<Text>PAGE</Text> <Characters CharCount="4"> <Char ID="0" Code="P"
Confidence="100" Left="1496" Top="150" Width="14" Height="19"/> <Char
ID="1" Code="A" Confidence="100" Left="1511" Top="150" Width="16"
Height="19"/> ... </Characters> </Word> ... </Words> <Fields
FieldCount="88"> ... <Field ID="2" Name="InvoiceNumber" Valid="false"
Page="0" Left="1649" Top="219" Width="139" Height="31"
ErrorDescription="Invalid invoice number"> <Text>7A6F2</Text> <Candidates
CandidateCount="61"> <Candidate ID="0" Weight="1.3563312626" Page="0"
Left="1649" Top="219" Width="139" Height="31"> <Text>7A6F2</Text>
</Candidate> <Candidate ID="1" Weight="0.53850805759" Page="0" Left="2337"
Top="219" Width="139" Height="30"> <Text>23013</Text> </Candidate> ...
</Candidates> </Field> ... <Field ID="15" Name="LineItems" Valid="true"
Page="-1" Left="0" Top="0" Width="0" Height="0"> <Table Valid="true">
<Columns ColumnCount="14"> ... <Column ID="4" Name="Description"/> <Column
ID="5" Name="Quantity"/> ... </Columns> <Rows RowCount="5"> <Row ID="0"
Page="0" Valid="true"> ... <Cell Column="4" Left="482" Top="1420"
Right="1146" Bottom="1552" Valid="true"> <Text>CDROM EDITION</Text>
```

```
</Cell> <Cell Column="5" Left="1221" Top="1420" Right="1813" Bottom="1552"
Valid="true"> <Text>1</Text> </Cell> ... </Row> ... </Rows> </Table>
</Field> </Fields> </Workdoc>
```

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.

Syntax

```
ExportToXml (ByVal DocumentLanguage as String, ByVal DocumentType as
String, ByVal Customer as String, ByVal eExportType as CDRExportType,
ByVal XMLFilePath as String)
```

Parameter	Description
DocumentLanguage	Customize this parameters to use it for filtering purposes. The values have no correlation with the workdoc. Note If the string is empty, the value defaults to Default. Null is not allowed.
DocumentType	Customize this parameters to use it for filtering purposes. The values have no correlation with the workdoc. Note If the string is empty, the value defaults to Default. Null is not allowed.
Customer	Customize this parameters to use it for filtering purposes. The values have no correlation with the workdoc. Note If the string is empty, the value defaults to Default. Null is not allowed.
eExportType	This parameter defines the type of information from the current document for the export to the XML files. **Possible values** CDRExportType
XMLFilePath	Defines the path for the XML file. Leave it as an empty string to save the XML files in the current program start directory. It is recommended to define the file path in script. You can specify an existing directory 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 directories, the method does not create them. If the method fails to create the file, an error message notifies you about the failed export.

Sample Code

The following sample code exports the OCR data for the current pWorkdoc into an XML file located in the *C:\Temp* directory.

```
pWorkdoc.ExportToXml("", "", "", CDRExportTypeOCRData, "C:\Temp\")
```

XML file format

Section	Description
Document	Contains the general document information, such as page count, line count, and word count.
Words	Contains information about the single words in the document, such as word text, word length, page number, and position of the word in pixels.
Characters	Contains information about the single characters of the word, such as character text and character position in pixels. For CI documents, this method exports only the word positions, but no individual character positions.

Sample XML

Example of the XML format with parameter eExportType set to CDRExportTypeOCRData

```
<?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>
```

GetEdge

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

Syntax

```
GetEdge (edgeSide as CDREdgeSide, edgeIndex as Long, pLeft as Long, pTop
as Long, pBottom as Long, pPageNr as Long)
```

Parameter	Description
edgeSide	Specifies if edges that contain left or right aligned words are requested. Possible values See CDREdgeSide
edgeIndex	Index of the edge to return. **Possible values** 0 to the result of EdgeCount – 1.
pLeft	Contains left coordinate of the edge.
pTop	Contains top coordinate of the edge.
pBottom	Contains bottom coordinate of the edge.
pPageNr	Contains page number of the edge.

GetFileSizeKB

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

Syntax

```
GetFileSizeKB(pWorkdoc as SCBCdrWorkdoc) as Integer
```

Sample Code

```
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
```

GetWorktextForPageArea

This method 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. These are the same coordinates that you would enter for a reading zone. For more information, see "Modify the general zone properties" in the *Oracle WebCenter Forms Recognition Designer User's Guide*.

Note: Use a zone to test the page area coordinates.

Syntax

```
GetWorktextForPageArea(Page as Long, Left as Long, Top as Long, Width as
Long, Height as Long, IncludePartial as Boolean)
```

Parameter	Description
Page	Page number of the image. 0 represents the first page of a multi-page document.
Left	Left coordinate of the page area in pixels.
Top	Top most coordinate of the page area in pixels.
Width	Width (length) of the area in pixels.
Height	Height of the area in pixels.
includePartial	True: Complete words that appear partially in the specified area with outside information. False: Restrict reading of worktext to specified area. Example The word "Invoice" exists on the page, but our page area only captures "Inv". Setting includePartial to False returns only "Inv", setting includePartial to True returns the entire word "Invoice".

Sample Code

The following sample code takes the OCR results of the top left page area and places the result into the first row table cell.

```
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
```

Load

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

Syntax

```
Load (Filename as String, ImageRootPath as String)
```

Parameter	Description
Filename	Name of the file.
ImageRootPath	Relative path of the file.

PDFExport

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

Syntax

```
PDFExport (FileName as String)
```

Parameter	Description
FileName	Name of the exported PDF file.

PDFGetInfoType

This method returns the PDF export type of a given document page.

Syntax

```
PDFGetInfoType (PageIdx as Long, pExportStyle as CDRPDFExportStyle)
```

Parameter	Description
PageIdx	Zero-based page number in the document.
pExportStyle	Export type. **Possible values** CDRPDFExportStyle

PDFSetInfoType

This method sets the type of a document page for the export into a PDF file.

Syntax

```
PDFSetInfoType (PageIdx as Long, ExportStyle as CDRPDFExportStyle)
```

Parameter	Description
PageIdx	Zero-based page number in the document.
ExportStyle	Export type. **Possible values** CDRPDFExportStyle

ReadZone

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

Syntax

```
ReadZone (PageIndex as Long, [left as Double = FALSE],  [top as Double =
FALSE], [right as Double = 1],  [bottom as Double = 1])
```

Parameter	Description
PageIndex	Specifies the DocPage where the OCR or text conversion is executed. **Possible values** • 0 to PageCount - 1 for working on single pages • -1 for executing OCR on all DocPages
Right	Specifies the right border of the OCR region in percent. Optional parameter. The default value is 1. To read until the right border, set the value to 100
Left	Specifies a left offset for the OCR region in percent. Optional parameter. The default value is 0. To read from the left border, set the value to 0
Top	Specifies the top offset for the OCR region in percent. Optional parameter. The default value is 0. To read from the top border, set the value to 0.
Bottom	Specifies the bottom line of the OCR region in percent. Optional parameter. The default value is 1. To read until the bottom border, set the value to 100.

RebuildBasicObjects

This method rebuilds the basic workdoc object words, text lines, and blocks without applying OCR or auto-rotation.

Execute this method after any modification to `Workdoc.Worktext`.

Syntax

```
RebuildBasicObjects()
```

Refresh

This method refreshes the workdoc’s DocPage currently shown in the viewer.

Syntax

```
Refresh ()
```

RenameDocFile

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

Syntax

```
RenameDocFile (DocIndex as Long, NewName as String)
```

Parameter	Description
DocIndex	Specifies the zero-based CIDoc or Image Index.
NewName	New name given to the document at DocIndex.

ReplaceFirstImage

This method replaces the first image in workdoc.

Syntax

```
ReplaceFirstImage (Path as String)
```

Parameter	Description
Path	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.

Syntax

```
Save (Filename as String, ImageRootPath as String)
```

Parameter	Description
Filename	Filename of the workdoc.
ImageRootPath	Relative path where all corresponding DocFiles are saved, empty if files are saved in the same directory as the workdoc.

SetDocPageIndex

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

Sample Code

The following sample code shows how to append one document to another.

```
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) Next j
```

UnloadDocs

This method releases all the images and CIDocs that belong to this workdoc.

Syntax

```
UnloadDocs ()
```

SCBCdrWorkdoc Properties

The SCBCdrWorkdoc object provides the following properties.

AttractorColor

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

Syntax

```
AttractorColor as OLE_COLOR
```

Sample Code

The following sample code sets the AttractorColor to green.

```
pWorkdoc.AttractorColor = vbGreen
```

BlockColor

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

Syntax

```
BlockColor as OLE_COLOR
```

Sample Code

The following sample code sets the color for block highlighting to cyan.

```
pWorkdoc.BlockColor = vbCyan
```

BlockCount

This read-only 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.

Syntax

```
BlockCount as Long
```

Sample Code

The following sample code 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
```

CandidateColor

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

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

Syntax

```
CandidateColor as OLE_COLOR
```

Sample Code

The following sample code sets the candidate color to magenta

```
pWorkdoc.CandidateColor = vbMagenta
```

ClsEngineConfidence

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

Syntax

```
ClsEngineConfidence (lMethodIndex as Long) as Long
```

Parameter	Description
lMethodIndex	Zero-based engine index in collection of classification engines.

Sample Code

The following sample code displays 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
```

ClsEngineDistance

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

Syntax

```
ClsEngineDistance (lMethodIndex as Long) as Long
```

Parameter	Description
lMethodIndex	Zero-based engine index in collection of classification engines.

Sample Code

The following sample code displays 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
```

ClsEngineResult

This property sets or returns 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.

Syntax

```
ClsEngineResult (MethodIndex as Long, DocClassIndex as Long) as
CDRClassifyResult
```

Parameter	Description
MethodIndex	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 WebCenter Forms Recognition 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.
DocClassIndex	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.

Sample Code

The following sample code sets the classification result of the Brainware Classify Engine to YES for a document in doclass "VOID". If Brainware Classify is the only engine or all other classes would be CDRClassifyNo, the document gets classified as VOID.

```
pWorkdoc.ClsEngineResult(Project.ClassifySettings.ItemIndex("Brainware
Classify Engine"), Project.AllClasses.ItemIndex("VOID"))= CDRClassifyYes
```

See also

CDRClassifyResult

ClsEngineWeight

This property sets or returns the classification weights within the Classification Result Matrix.

Syntax

```
ClsEngineWeight (MethodIndex as Long, DocClassIndex as Long) as Double
```

Parameter	Description
MethodIndex	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 WebCenter Forms Recognition 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.
DocClassIndex	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.

CurrentBatchState

This read-only property returns the temporary document batch state, a numeric value between 0 and 999. To set this value, use the methods LoadWorkdoc and UpdateDocument of the Cedar Batch component.

Syntax

```
pWorkdoc.CurrentBatchState
```

DisplayPage

This property sets or returns the displayed DocPage specified by the zero-based index of the workdoc in the viewer.

Use this property within the VerifierFormLoad event to set the initial page that displays when the form is loaded.

You can also set this property in class level events, such as OnAction or the Validate event for fields, to navigate to a specified document page.

Note: When the Verifier option "Keep Focus on Field" is active, this option takes precedence over any DisplayPage value set in script. If the focus is currently on a field which is mapped to document text, any attempt to set DisplayPage results in navigation to the mapped document page.

Syntax

```
DisplayPage as Long
```

Sample Code

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.

Note: Index 0 represents page 1.

The following sample code displays page 3 if the document has 4 pages or more.

```
Private Sub ScriptModule_VerifierFormLoad(pWorkdoc as
SCBCdrPROJLib.SCBCdrWorkdoc, FormClassName as String, FormName as String)
```

```
If pWorkdoc.PageCount >=4 Then pWorkdoc.DisplayPage = 2 End if End Sub
```

See also

- OnAction
- Validate
- VerifierFormLoad

DocClassName

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

Syntax

```
DocClassName as String
```

Sample Code

```
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
```

DocFileCount

This read-only property returns the number of documents from which the workdoc is built.

Syntax

```
DocFileCount as Long
```

DocFileDatabaseID – Unique ID

The read-only property pWorkdoc.DocFileDatabaseID returns the database ID of document files attached to a WebCenter Forms Recognition 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 scripts as a unique identifier of document files that were processed by WebCenter Forms Recognition.

Syntax

```
DocFileDatabaseID (ByVal Index as long) as Long
```

Parameter	Description
Index	The index parameter. **Possible values** 0 to DocFileCount -1

Sample Code

The following sample code returns the unique ID of the last document file attached to a workdoc.

```
Dim lUniqueID as Long lUniqueID = pWorkdoc.DocFileDatabaseID
(pWorkdoc.DocFileCount - 1)
```

DocFileName

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

Syntax

```
DocFileName (index as Long) as String
```

Parameter	Description
Index	The index parameter. **Possible values** 0 to DocFileCount-1

Sample Code

If a workdoc was created from a single document, such as a multi-TIFF file, you can get the name of the document file by accessing the 0 index.

```
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
```

DocFileType

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

Syntax

```
DocFileType (index as Long) as CDRDocFileType
```

Parameter	Description
Index	The index parameter. **Possible values** 0 to DocFileCount-1

See also

CDRDocFileType

DocState

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

Syntax

```
DocState as CDRDocState
```

See also

CDRDocState

EdgeCount

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

Syntax

```
EdgeCount (edgeSide as CDREdgeSide) as Long
```

Parameter	Description
edgeSide	Flag to distinguish between left and right edges. **Possible values** CDREdgeSide

ErrorDescription

This property sets or returns an error description.

Syntax

```
ErrorDescription as String
```

Sample Code

```
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
```

FieldColor

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

Syntax

```
FieldColor (FieldValid as Boolean) as OLE_COLOR
```

Parameter	Description
FieldValid	If set to TRUE it specifies the color for valid fields or it specifies the color for invalid fields if FALSE.

Fields

This read-only property provides access to all fields of a document.

Syntax

```
Fields as ISCBCdrFields
```

Sample Code

The following sample code reads the text content of a simple field.

```
Dim FieldContent as string FieldContent = pWorkdoc.Fields.Item
("MyField").Text
```

FileName

This read-only property contains the database ID of the workdoc and returns the database workdoc ID and Name.

Note: To get the file name of the image from which the workdoc was created, use the DocFileName property.

Syntax

```
Filename as String
```

Folder

This read-only property accesses the folder to which the workdoc belongs.

Syntax

```
Folder as ISCBCdrFolder
```

FolderIndex

This read-only property provides the index of the Folder to which a workdoc belongs.

Syntax

```
FolderIndex as Long
```

ForceClassificationReview

This property provides the ability to force a manual classification review even if the classification succeeded. Set or return this property in the PostClassify event.

Syntax

```
ForceClassificationReview as Boolean
```

Sample Code

The following sample code shows how to force the manual classification process from the script event "PostClassify".

```
Private Sub ScriptModule_PostClassify(pWorkdoc as
SCBCdrPROJLib.SCBCdrWorkdoc) If pWorkdoc.DocClassName =
"VeryImportantClass" Then pWorkdoc.ForceClassificationReview = True End If
 End Sub
```

HighlightCandidate

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

Syntax

```
HighlightCandidate as Long
```

HighlightField

This property sets or returns the field index value of the highlighted field.

Syntax

```
HighlightField as Long
```

HighlightMode

This property sets or returns the highlighting mode.

Syntax

```
HighlightMode as CDRHighlightMode
```

See also

CDRHighlightMode

Image

This read-only property returns an image object for the specified DocPage of the workdoc.

Syntax

```
Image (index as Long) as ISCBCroImage
```

Parameter	Description
Index	Index of the DocPage. **Possible values** 0 to PageCount - 1.

IsPlainText

This property sets or returns if worktext is plain text.

Syntax

```
IsPlainText as Boolean
```

Language

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

Syntax

```
Language as String
```

LineColor

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

Syntax

```
LineColor as OLE_COLOR
```

PageCount

This read-only property returns the number of displayable DocPages of the workdoc.

Syntax

```
PageCount as Long
```

Sample Code

```
intImageCount=pWorkdoc.PageCount 'Get the number of pages in workdoc
```

Pages

This read-only property returns a single DocPage of the workdoc.

Syntax

```
Pages (PageIndex as Long) as ISCBCdrDocPage
```

Parameter	Description
PageIndex	Index of the DocPage to access, which is valid from 0 to PageCount-1.

Paragraph

This read-only property returns the paragraph array of the workdoc.

Syntax

```
Paragraph (index as Long) as ISCBCdrTextBlock
```

Parameter	Description
Index	Specifies the index of the paragraph. **Possible values** 0 to ParagraphCount – 1.

ParagraphCount

This read-only property returns the number of paragraphs.

Syntax

```
ParagraphCount as Long
```

ShowTooltips

This property sets or returns if tool tips display when moving the mouse pointer over a displayed workdoc.

Syntax

```
ShowTooltips as Boolean
```

SkipTrainingWithEngine

Use this property to set or return whether the specified trainable engine has to skip this document in the training process.

Syntax

```
SkipTrainingWithEngine (bstrEngineName as String) as Boolean
```

Parameter	Description
bstrEngineName	Name of classification engine.

Table

This read-only property returns a table for a given index of the workdoc.

Syntax

```
Table (index as Long) as ISCBCdrTable
```

Parameter	Description
Index	Specifies the index of the table. **Possible values** 0 to TableCount-1

TableCount

This read-only property returns the number of table objects stored within the workdoc.

Syntax

```
TableCount as Long
```

TextBlock

This read-only property returns the TextBlock by an index of the workdoc.

Syntax

```
TextBlock (index as Long) as ISCBCdrTextBlock
```

Parameter	Description
Index	[in] Specifies the index of the TextBlock. Valid indices are from 0 to BlockCount-1.

TextlineCount

This read-only property returns the number of text lines present in a workdoc.

Syntax

```
TextlineCount as Long
```

TrainedWithEngine

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

Syntax

```
TrainedWithEngine (bstrEngineName as String) as Boolean
```

Parameter	Description
bstrEngineName	Name of engine.

UniqueID

This read-only property returns the unique id of the workdoc as a 60-character string with the identifier generation date, time, and a dynamically created UUID.

WebCenter Forms Recognition generates an identifier when creating a new workdoc object and when adding new images or document files to the workdoc.

You can use the `UniqueID` property to identify the document.

Syntax

```
UniqueId as String
```

Sample Code

```
Private Sub ScriptModule_PreClassify(pWorkdoc As
SCBCdrPROJLib.SCBCdrWorkdoc) Dim strWorkdocID As String strWorkdocID =
pWorkdoc.UniqueID End Sub
```

UniqueID Example

```
2019-11-05-18-01-32-484-CA4B5BA7-6488-4685-89D0-55C410DF1172
```

Word

This read-only property provides access to the Word array of the workdoc.

Syntax

```
Word (index as Long) as ISCBCdrWord
```

Parameter	Description
Index	Index of the requested Word. **Possible values** 0 to WordCount-1

WordColor

This property sets or returns the color used for word highlighting.

Syntax

```
WordColor as OLE_COLOR
```

WordCount

This read-only property returns the number of words of the workdoc.

Syntax

```
WordCount as Long
```

Sample Code

```
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
```

WordSegmentationChars

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

Syntax

```
WordSegmentationChars as String
```

Textline

This read-only property returns text line by an index of the workdoc.

Syntax

```
Textline (index as Long) as ISCBCdrTextBlock
```

Parameter	Description
Index	Zero-based index.

Worktext

This property provides access to the raw OCR results represented by the SCBCroWorktext object.

Note: After any modification to Workdoc.Worktext, it is highly recommended to rebuild the Workdoc objects using Workdoc.RebuildBasicObjectsto avoid corrupted words in the Workdoc.

Syntax

```
Worktext as ISCBCroWorktext
```

Sample Code

The following sample code shows how to rebuild the Workdoc after removing lines.

```
For lngLine = (lngLineCount - 1) To (lngHitLine) Step -1
pWorkdoc.Worktext.RemoveLine(lngLine) Next lngLine
pWorkdoc.RebuildBasicObjects 'added to rebuild the workdoc's objects from
the modified workdocs's main worktext
```

SCBCdrWorkdoc Named Properties

The SCBCdrWorkdoc object provides the following named properties.

BatchID

This read-only workdoc property allows you to get the Batch ID in which the current workdoc resides.

Syntax

```
strBatchID as String
```

Sample Code

The following sample code shows how to return the Batch ID.

```
Dim strBatchID as String strBatchID = pWorkdoc.NamedProperty("BatchID")
```

CreationDate

This read-only workdoc property returns 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.

Syntax

```
CreationDate as Date
```

Sample Code

The following sample code shows how to get the creation date.

```
Dim dtCreationDate as Date dtCreationDate = pWorkdoc.NamedProperty
("CreationDate")
```

CreationDateAsFileTimeUTC

This read-only workdoc property returns the UTC 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.

Syntax

```
CreationDateAsFileTimeUTC as Long
```

Sample Code

The following sample code shows how to return the creation date.

```
Dim dtCreationDateUTC as Long dtCreationDateUTC = pWorkdoc.NamedProperty
("CreationDateAsFileTimeUtc")
```

ExportPdfVersion

Use this named property in the PostImport or PostClassify event to specify the PDF format used for export.

Note: The parameter is taken into account only if the option "Generate PDF file on export" in the Runtime Server configuration is active.

For more information, see "Generate PDF files on export" in the *Oracle WebCenter Forms Recognition Runtime Server User's Guide*.

Syntax

```
pWorkdoc.NamedProperty("ExportPdfVersion") as String
```

Parameter	Description
A2B	Default value. Export the file in PDF/A-2B format.
1.7	Export the file in PDF 1.7 format.

Sample Code

```
Private Sub ScriptModule_PostClassify(pWorkdoc as
SCBCdrPROJLib.SCBCdrWorkdoc) pWorkdoc.NamedProperty("ExportPdfVersion") =
"1.7" End Sub
```

IgnoreAnalysisFailures

This is an optional capability to ignore any errors during WebCenter Forms Recognition'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 trigger 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 do 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.

Syntax

```
pWorkdoc.NamedProperty("IgnoreAnalysisFailures")
```

Sample Code

```
' 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
```

PCAppType

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

Syntax

```
pWorkdoc.NamedProperty ("PCAppType")
```

Parameter	Description
"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 US dollars. 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.

Sample Code

```
Private Sub Document_PreExtract(pWorkdoc as SCBCdrPROJLib.ISCBCdrWorkdoc)
pWorkdoc.NamedProperty("PCAppType") = "HVOL" 'High Value Amount Range End
Sub
```

PCCheckType

Use this named property to configure the check type recognition.

Syntax

```
pWorkdoc.NamedProperty ("PCCheckType")
```

Parameter	Description
"ALL" Personal checks Business checks Cash tickets Deposit slips Money orders	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" Personal checks	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 percent personal checks, this setting can improve processing speed while not significantly affecting accuracy.
"PB" – default setting Personal checks Business checks	This parameter sets the Check Analysis Engine to expect checks and cash tickets. Use this setting if your documents consist mainly of checks.
"PBD" Personal checks Business checks Cash tickets Deposit slips	This parameter sets the Check Analysis Engine to expect checks, cash tickets, and deposit slips.

Sample Code

```
Private Sub Document_PreExtract(pWorkdoc as SCBCdrPROJLib.ISCBCdrWorkdoc)
```

```
pWorkdoc.NamedProperty("PCCheckType") = "P" 'Personal Checks only End Sub
```

PCDateHint

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

Syntax

```
pWorkdoc.NamedProperty ("PCDateHint")
```

Parameter	Description
Format	Use the following format for the reference date. YYYY.MM.DD Example 2015.06.18 **Note:** To set the reference date to the system date, add an empty string.

Sample Code

```
Private Sub Document_PreExtract(pWorkdoc as SCBCdrPROJLib.ISCBCdrWorkdoc)
pWorkdoc.NamedProperty("PCDateHint") = "2013.04.12" 'April 12th 2012 End
Sub
```

The following example sets the reference date to the system date:

```
Private Sub Document_PreExtract(pWorkdoc as SCBCdrPROJLib.ISCBCdrWorkdoc)
pWorkdoc.NamedProperty("PCDateHint") = " " 'System date End Sub
```

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.

Syntax

```
pWorkdoc.NamedProperty ("PCReReadAlways")
```

Parameter	Description
True	The Check Analysis Engine repeats the analysis, if - Executed in the Designer's Definition Mode for extraction. - The AnalyzeField method is used in the script. If AnalyzeField is used on a Check Analysis Engine field, all fields with Check Analysis Engine assigned are re-analyzed. The "<Field>_PostAnalysis" event only triggers for the field for which the analyzed field was triggered. - AnalyzeDocument is used in script.

False	Default value. The Check Analysis Engine does not execute a secondary analysis on a document when WebCenter Forms Recognition did not unload the document from the memory.

Sample Code

```
Private Sub Document_PreExtract(pWorkdoc as SCBCdrPROJLib.ISCBCdrWorkdoc)
pWorkdoc.NamedProperty("PCReReadAlways") = True 'Switch on re-Analysis End
Sub
```

SkipDocumentReprocessingAfterMerging

By default, when workdocs are combined in the AppendWorkdoc event, the new combined document is classified and extracted after merging. Set this named property in the AppendWorkdoc event to `True` to skip reprocessing. In this case, the new combined document has the classification and extraction results of `pLastWorkdoc` after merging.

Sample Code

The following sample code shows how to skip reprocessing.

```
Private Sub ScriptModule_AppendWorkdoc(pLastWorkdoc As
SCBCdrPROJLib.ISCBCdrWorkdoc, pCurrentWorkdoc As
SCBCdrPROJLib.ISCBCdrWorkdoc, pAppendType As SCBCdrPROJLib.CdrMPType) '
Merge pLastWorkdoc and pCurrentWorkdoc if they are of same class "CLASS_A"
 If pLastWorkdoc.DocClassName = "CLASS_A" And pCurrentWorkdoc.DocClassName
= "CLASS_A" pAppendType = CdrSubseqPage ' The new combined document
usually does not need to be classified/extracted
pLastWorkdoc.NamedProperty("SkipDocumentReprocessingAfterMerging") = True
End If End Sub
```

SkipTableCellMassValidation

This method allows you to optionally activate special "skip table cell mass validation" mode for validation of table cells. By default, WebCenter Forms Recognition uses "mass validation of invalid cells". This means that when the user presses Enter within an invalid cell, all other invalid cells are automatically re-validated by the system. This behavior can lead to performance problems in WebCenter Forms Recognition 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 triggered 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".

Sample Code

The following sample code shows how to switch the validation mode individually for each processed document.

```
pWorkdoc.NamedProperty("SkipTableCellMassValidation") = True
```

XML_ExportCandidates

This named property controls whether the list of candidates for each field in the workdoc is exported to the XML file. The text, weight and position of each candidate is exported.

Set the property to `True` in the ExportDocument event to include the candidate list for each field in the exported file.

The default value is `False`.

Syntax

```
pWorkdoc.NamedProperty("XML_ExportCandidates") as Boolean
```

See also

- ExportDocumentToXml
- "Export to XML" in the *Oracle WebCenter Forms Recognition Runtime Server User's Guide*.

XML_ExportWordChars

This named property controls whether the characters composing OCR words in the workdoc are exported to the XML file. The character code, confidence and position of each character are exported.

Note: For CI documents, position and confidence values for individual characters are not available. The Top, Left, Height, Width and Confidence values for each letter are those for the word in which the character is found.

Set the property to `True` in the ExportDocument event to include the character information in the exported file. This property has no impact if the named property XML_ExportWords is set to `False`.

The default value is `False`.

Syntax

```
pWorkdoc.NamedProperty("XML_ExportWordChars") as Boolean
```

See also

- ExportDocumentToXml
- "Export to XML" in the *Oracle WebCenter Forms Recognition Runtime Server User's Guide*.

XML_ExportWords

This named property controls whether the OCR words in the workdoc are exported to the XML file.

Set the property to `True` in the ExportDocument event to include the list of words in the exported file.

The default value is `False`.

Syntax

```
pWorkdoc.NamedProperty("XML_ExportWords") as Boolean
```

See also

- ExportDocumentToXml

- "Export to XML" in the *Oracle WebCenter Forms Recognition Runtime Server User's Guide*.

SCBCdrPROJLib

The Cedar project object SCBCdrPROJLib represents a complete project definition including all document classes, field definitions, and used classification and extraction methods.

SCBCdrPROJLib Type Definitions

The SCBCdrPROJLib object provides the following type definitions.

CDRApplicationName

This type defines the application type.

Type	**Description**
TANDesigner	Designer
TANLearnSetManager	Learn Set Manager
TANLocalVerifier	Verifier used as local project for SLW
TANRuntimeServer	Runtime Service Instance
TANVerifier	Verifier
TANWebVerifier	Web Verifier
TANUnknown	Unknown application

CDRBatchReleaseAction

This types defines the automatic action when releasing a batch in Verifier.

Type	**Description**
CdrBatchReleaseActionCancel	Return to current batch and last document verified.
CdrBatchReleaseActionReturnToList	Return to batch list.
CDRBatchReleaseActionUndefined	Undefined
CdrBatchReleaseActionUserDefined	Default value Let the user decide what to do next selecting the required option in a dialog.
CdrBatchReleaseActionVerifyNextInvalidBatch	Open next batch to verify.

CdrBatchReleaseActionVerifyNextInvalidState	Verify this batch with next invalid state if batch is still invalid, otherwise get next invalid batch.

CDRClassifyMode

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

Type	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 WebCenter Forms Recognition workflow settings for batches.

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

Type	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
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 in Designer without changing its workflow status
CDRModifiedByVerifierApplication	The document was saved in Verifier 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

CDRScanningFailed	Images scanning failed
CDRScanningSucceeded	Images scanning succeeded

CdrDocumentBinarizationMode

This type defines the binarization mode.

Type	Description
CdrDocumentBinarizationSkipped	Deactivates forced binarization of CIDoc images. Activating this setting may improve OCR results for grayscale and colored images. **This setting is compatible with the following preprocessing methods.** - Binarisation - Despeckle [IG] - Invert **This setting is compatible with the following recognition engines.** - FineReader 10 - FineReader 11 - Cairo OMR - QualitySoft Barcode **This setting is not compatible with the following preprocessing methods.** - Box & Comb Removal - Clean Border [IG] - Lines Manager **This setting is not compatible with the following recognition engines.** - Cleqs Barcode - Transcripts You can use non-compatible methods and/or engines with this setting active provided that preprocessing includes Binarisation. In case of incompatible preprocessing methods, Binarisation must appear prior to the method. **Note**: This binarization is a simple threshold based method and does not produce results identical to the dynamic binarization which is performed when `CdrDocumentBinarizationSkipped` is not active. **Sample Code**

	The following sample code shows how to activate the setting. `Private Sub ScriptModule_Initialize(ByVal ModuleName As String) Settings.DocumentBinarizationMode = CdrDocumentBinarizationSkipped End Sub`
CdrDocumentBinarizationAdvanced	Default setting. Executes advanced document binarization.

CDRFieldType

This type defines the type of a FieldDef

Type	Description
CDRFieldTypeTable	The field type is table.
CDRFieldTypeText	The field type is text, which may be single or multi-line text.

CdrFocusChangeReason

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

Type	Description
CdrBeforeFormClosed	Focus changed by closing the form
CdrBeforeFormLoaded	Focus is not set yet as the form has just been loaded
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
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

CdrForceValidationMode

This table defines the options for Force Validation.

Type	Description
CdrForceValDefault	Inherit the Force Validation Mode.
CdrForceValForbidden	Force validation by pressing Enter three times is not allowed.
CdrForceValPermitted	Force validation by pressing Enter three times is allowed.

CdrLicenseCounter

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

Type	Description
TLCFineReaderRemainingUnits	Remaining page units available to be processed by the FineReader licensing scheme.
TLCPeriodDocumentsClassified	Documents classified within the licensing period.
TLCPeriodDocumentsExported	Documents exported within the licensing period.
TLCPeriodDocumentsExtracted	Documents extracted within the licensing period.
TLCPeriodDocumentsOCRed	Documents OCRed within the licensing period.
TLCPeriodDocumentsProcessed	Documents processed within the licensing period.
TLCPeriodDocumentsValidatedVerifier	Documents validated in Verifier within the licensing period.
TLCPeriodPagesImported	Pages imported within the licensing period.
TLCPeriodPagesOCRed	Pages OCRed within the licensing period.
TLCPeriodPagesProcessed	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	Total Overall 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, it means that the feature is licensed and available for usage.

Type	**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.
CDRLfnDesignerDesignLicense	The Designer program 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.

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.
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.
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 program module License Feature.
CDRLfnVerifierCount	The Verifier program count License Feature.
CDRLfnZoneAnalysisEngine	The Zone Analysis engine License Feature.

CdrLocalTrainingReason

This type defines the possible reasons for local training.

Type	**Description**
CdrAddedByUser	A Verifier user manually initiated the training process.
CdrAddedSinceDocumentWasPoorlyExtracted	The document was considered as poorly extracted and therefore needs to be trained.
CdrRejectedByUser	A Verifier user manually rejected the training process.

CdrRejectedSinceDocumentWasWellExtracted	The document was well extracted before the manual verification process and therefore is not needed to be trained.

CdrMessageSeverity

This type defines the different message severities.

Type	Description
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 administrators through the System Monitoring service of Runtime Server. This option is applicable when the call is executed from within the Runtime Server only.
CDRSeverityLogFileOnly	Store the message to the program 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 only.

CdrMessageType

This type defines the different message types.

Type	Description
CDRTypeInfo	An informational message.
CDRTypeWarning	A warning message.
CDRTypeError	An error message.

CdrMPType

This type defines the possible results of the multi-page classification.

Type	Description
CdrAttachmentPage	The classified page is an attachment.
CdrFirstPage	The classified page is the first page of a new document and does not belong to the previous page.

CdrLastPage	The classified page is the last page of the current document. The next page starts a new document.
CdrPageUndefined	The page could not be classified.
CdrSinglePage	The classified page belongs to a single page document.
CdrSubseqPage	The classified page belongs to the previous page. More pages can be appended to the current document.
CdrUserCorrectedPage	The page was corrected manually by the Verifier user.
CdrUserRejectedPage	The page was rejected manually by the Verifier user.

CDRsiModule

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

Type	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

CdrSLWDifferentResultsAction

Specified how the program continues the processing when the Template and Associative Search engines determine different results during classification.

Type	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 Designer.
CdrUseDocumentClassName	Automatically assign current document class name to the supplier field content.
CdrUseSupplierField	Automatically assign supplier field content to the document class name.

CdrTableFocusChangeReason

This type defines the possible causes for a cell focus change in a verification table.

Type	Description
CdrTfcrBrainwareExtractionApplied	The focus changed due to applied interactive Brainware table extraction.
CdrTfcrCellBitmapClicked	The focus changed because a cell bitmap was clicked.
CdrTfcrCellDoubleClicked	The focus changed because a cell was double clicked.
CdrTfcrCellLocationClicked	The focus changed because the location of this cell was clicked in the active Cairo Viewer.
CdrTfcrColumnMapped	The focus changed because a column was mapped.
CdrTfcrColumnsSwapped	The focus changed because columns were swapped.
CdrTfcrColumnUnmapped	The focus changed because a column was unmapped.
CdrTfcrEnterPressed	The focus changed because the validation of a cell was invoked.
CdrTfcrFirstInvalidFieldSelected	The focus changed because the first invalid field was selected.
CdrTfcrFocusRefreshed	The focus changed Selected the table was refreshed.
CdrTfcrFormLoaded	The focus was initialized because a form was loaded.
CdrTfcrMouseClicked	The focus changed because a cell was selected by a mouse click.
CdrTfcrRowsMerged	The focus changed because rows were merged.
CdrTfcrRowsRemoved	The focus changed because one or more table rows were removed.
CdrTfcrSelectionCopied	The focus changed because a user-selected area was copied from the active Cairo Viewer.
CdrTfcrTableCandidateChanged	The focus changed because a new table candidate was selected by the user.
CdrTfcrTabPressed	The focus changed because the user pressed the Tab key or any arrow key.
CdrTfcrUnknownReason	The focus changed because of an unknown cause.
CdrTfcrWordCopied	The focus changed because a word was copied from the active Cairo Viewer.

CdrTableHeaderClickType

This type defines the possible events which can occur when the user clicks on a table header button.

Type	Description
CdrColumnHeaderClicked	The user clicked a column header button.
CdrColumnHeaderDoubleClicked	The user double-clicked a column header button.
CdrColumnHeaderRightButtonClicked	The user right-clicked a column header button.
CdrRowHeaderClicked	The user clicked a row header button.
CdrRowHeaderDoubleClicked	The user double-clicked a row header button.
CdrRowHeaderRightButtonClicked	The user right-clicked a row header button.
CdrTableHeaderClicked	The user clicked the table header button.
CdrTableHeaderDoubleClicked	The user double-clicked the table header button.
CdrTableHeaderRightButtonClicked	The user right-clicked the table header button.

CdrValFieldType

This enumeration contains different validation types for fields.

Type	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.

CdrVerifierClassifyReason

This type defines the reasons for document classification.

Type	Description
CdrChangedReason	The user selected a new class without leaving the classification view.

Type	Description
CdrInitReason	Manual classification view has just been displayed.
CdrValidatedReason	The document class has been changed.

CDRVerifierExceptionReason

This type defines the reasons for an exception event in Verifier.

Type	Description
CDRExceptionUserActivated	Exception induced by a user clicking on the Exception Handling button in Verifier.
CDRExceptionFieldValidated	Exception induced indirectly after field validation.
CDRExceptionDocumentValidated	Exception induced indirectly after document validation.
CDRExceptionUserAction	Exception induced indirectly by a user when executing an action in Verifier.
CDRExceptionManualClassification	Exception induced indirectly after manual classification.
CDRExceptionUnknown	Exception induced with unknown reason.
CDRExceptionUserActivatedForBatch	Exception induced by a user clicking on the Exception Handling for batches button in Verifier.

SCBCdrProject Methods

The Cedar Project object provides the following methods.

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.

Syntax

```
ExtractClassificationField (pWorkdoc as ISCBCdrWorkdoc)
```

Parameter	Description
pWorkdoc	Current workdoc

GetHostProperties

This method allows the user to get information about the current machine, program, and WebCenter Forms Recognition user.

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
appType	Type of the calling application. The parameter can be read from script. For more information, see CDRApplicationName.
appSubType	For internal use only
appInstance	Runtime Service instance name, if ApplicationName is TANRuntimeServer. Not used for other applications.
appUsername	Login name of the current WebCenter Forms Recognition user. • WebCenter Forms Recognition user for Designer, Verifier, LSM, and Web Verifier • Windows user for Runtime Server
appIP	IP address of the computer.
appMachineName	Machine name running the script.
appLicensee	Customer name of the used license file.

Sample Code

The following sample code calls the GetHostProperties in the initialize event. The method returns information into variables as to where the script is executed, who is executing it, and which program module is executing it.

```
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
```

GetVerifierProject

This method returns the Verifier Project.

Syntax

```
GetVerifierProject (ppVal as Object)
```

Parameter	Description
ppVal	[out] Verifier Project Object

Lock

This method locks the Project for updating.

Syntax

```
Lock ()
```

LogScriptMessage

This method is obsolete.

Use LogScriptMessageEx 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.

Syntax

```
LogScriptMessageEx(ByVal Type as CDRMessageType, ByVal Severity as
CDRMessageSeverity, ByVal Message as String)
```

Parameter	Description
Type	Type of the message, such as information, warning, or error. **Possible values** CDRMessageType
Severity	Severity code of the message. This option depicts where the message appears (Log, System Monitoring, or as an email). **Possible values** CdrMessageSeverity

Message	The message text to display or send.

Sample Code

You can place the following sample code in any event. When the event triggers, a message is written to the core product log file (H_, D_, V_ or U_ log).

```
Project.LogScriptMessageEx(CDRTypeInfo, CDRSeverityLogFileOnly, "My
message")
```

```
[Info] |30| 01:59:33.312 | 3108 | 668184k/1428344k | 514004k/3520792k |
57176k/67252k | 238 | 38/43 | My message
```

MoveDocClass

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

Syntax

```
MoveDocClass (Name as String, NewParentName as String)
```

Parameter	Description
Name	Name of moved DocClass
NewParentName	Name of new ParentDocClass

PerformScriptCommandRTS

This method restarts or stops the Runtime Server through a custom script.

You can use this method to perform a stop on Runtime Server. This method stops the currently running

Runtime Server instance executing the script to either stop or restart.

Syntax

```
PerformScriptCommandRTS (CommandID as Long, MessageType as Long, UserCode
as Long, MessageDescription as String)
```

Parameter	Description
CommandID	Identifier of the command to execute on the RTS instance. **Possible values** • 0: Force the RTS instance to stop document processing. • 1: Restart the RTS instance.

MessageType	The type of message to log when the command executes. **Possible values** • 0: Information • 1: Warning • 2: Error **Note:** Error messages are additionally forwarded to the 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 on the common Runtime Server log file and in the case of error messages on the MMC administration console.

Sample Code

The following code example demonstrate how to stop and restart the RTS instance.

```
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 stop from
Custom Script" ' specified message as warning with error code "999"
Project.PerformScriptCommandRTS 1, 1, 999, "RTS is going to be restarted
from Custom Script"
```

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 need to reload each time a batch processes.

Syntax

```
Project.ReleaseAllAdsPools()
```

Sample Code

The following sample code shows the implementation for RTS processing. It is placed in the Initialize event.

```
Private Sub ScriptModule_Initialize(ByVal ModuleName as String)
Project.ReleaseAllAdsPools() End Sub
```

The following sample code shows the implementation for Verifier and Web Verifier process. It is placed in the BatchOpen event.

```
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
```

The scripts above provide an entry in the log similar to the following.

```
[Info] |20| 14:05:26.812 | 7488 | 4820400k/3448008k | 5829788k/6631068k |
195812k/200160k | 543 | 73/57 | Disconnecting ADS Pool for class:
Invoices, field: VendorName
```

ReportLicensingStatus

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

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

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

For further details on licensing counters present or available in the license file, see the *Oracle WebCenter Forms Recognition Licensing Guide*.

Syntax

```
ReportLicensingStatus (ReportActiveLicensingOnly as Boolean, Severity as
SCBCdrPROJLib.CDRMessageSeverity)
```

Parameter	Description
ReportActiveLicensingOnly	**Possible values** • True: Return only the license counters which are active in the license file. • False: Return all license counters.
Severity	The location of the utilization output to be sent to. **Possible values** See CdrMessageSeverity

Sample Code

The following code example demonstrates how to get licensing utilization information for all licensing counters.

```
Project.ReportLicensingStatus(False,
SCBCdrPROJLib.CDRMessageSeverity.CDRSeverityLogFileOnly)
```

Log file output example

```
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.
```

```
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.
```

See also

- GetLicenseValueByName
- GetLicenseValueByID

SecurityUpdateAddUserGroup and SecurityUpdateAddUserGroupPwd

This method updates or adds the database security credentials. This script call creates or updates the WebCenter Forms Recognition users, roles, and groups.

When updating the security policy of WebCenter Forms Recognition through a custom script, only the database tables update. You cannot modify the project security 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 is recovered and marked as "deleted = false" as soon as the user is present in SecurityUpdate.

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

Syntax

```
SecurityUpdateAddUserGroup (UserName as String, ExternalGroupID as Long,
UserRole as String, UserDomain String) SecurityUpdateAddUserGroupPwd
(UserName as String, UserPassword as String, ExternalGroupID as Long,
UserRole as String,UserDomain String)
```

Parameter	**Description**
UserName	The user name to create or update within the database. These are the user credentials to enter to log on to the system. If Domain is populated, the user must enter MyDomain\UserName for logging in to the verification application.
UserPassword	This password applies only when creating or recovering a user. For those auto-imported users that were previously imported into WebCenter Forms Recognition, the password remains unchanged. Use case rules • Auto-imported users with empty passwords are required to change their password upon first login. • Auto-imported users with NON-empty passwords are NOT required to change their password upon first login.

	Auto-imported users who already changed their password upon first login are not required to change their password again.
ExternalGroupID	The external group ID is a security number. A batch and a user are assigned a group ID that enables the user to verify only batches that fall under the same group ID assigned to that user.

UserRole	The user role assigned to the Verifier user.
	The role can be one or a logical combination of the following text strings.
	• ADM: Administrator
	• AEB: Authorization for External Batches
	• SET: Can access settings
	• VER: Verifier user
	• SLV: Verifier supervisor (learnset nomination)
	• SLM: Learnset Manager (global learnset manager)
	• FLT: Filtering
	The following combinations of roles are possible.
	Create a user with Verifier and Filter roles, but with no SET role:
	`Project.SecurityUpdateAddUserGroup "User2", 999, "VER|FLT", "BDomain"`
	Create a user with Verifier, Settings and Filter roles:
	`Project.SecurityUpdateAddUserGroup "User2", 999, "VER|SET|FLT", "BDomain"`
	Create a user with Verifier role only, with no SET and FLT role:
	`Project.SecurityUpdateAddUserGroup "User2", 999, "VER", "BDomain"`
	Note: There is no need to combine SET/FLT roles with ADM, SLV, or SLM as these already contain FLT and SET roles by default.

Sample Code

The example below updates the database user security on a regular basis. The script can be modified to lookup users/roles and update the WebCenter Forms Recognition user table.

```
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
"User2", 222, "AEB", "BDomain" Project.SecurityUpdateAddUserGroup
```

```
"User10", 777, "ADM", "BDomain" Project.SecurityUpdateAddUserGroupPwd
("User2", "pass", 777, "VER|FLT", "") Project.SecurityUpdateCommit End Sub
```

See also

- SecurityUpdateStart
- SecurityUpdateCommit
- SecurityUpdateUserParameter
- UpdateSystemSecurity
- PostImportBatch

SecurityUpdateCommit

This method completes the security update process. This script call is required to complete updating the WebCenter Forms Recognition users, roles, and groups.

When updating the security policy of WebCenter Forms Recognition through a custom script, only the database tables update. The project security is not modified after a script update.

Syntax

```
Project.SecurityUpdateCommit
```

Sample Code

The following sample code updates the database user security on a regular basis. The script can be updated to lookup users/roles and update the WebCenter Forms Recognition user table.

```
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,
"VER|SET", "BDomain" Project.SecurityUpdateAddUserGroup "User6", 777,
"VER|FLT", "BDomain" Project.SecurityUpdateAddUserGroup "User7", 333,
"AEB", "BDomain" Project.SecurityUpdateAddUserGroup "User10", 777, "ADM",
"BDomain" Project.SecurityUpdateCommit End Sub
```

See also

- SecurityUpdateStart
- SecurityUpdateAddUserGroup
- SecurityUpdateUserParameter
- UpdateSystemSecurity
- PostImportBatch

SecurityUpdateStart

This method instantiates the security update process. This script call is required to begin updating the WebCenter Forms Recognition users, roles, and groups.

When updating the security policy of WebCenter Forms Recognition through a custom script, only the database tables update. The project security is not modified after a script update.

Syntax

```
Project.SecurityUpdateStart
```

Sample Code

The following sample code updates the database user security on a regular basis. The script can be updated to lookup users/roles and update the WebCenter Forms Recognition user table.

```
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, "VER|SET",
"BDomain" Project.SecurityUpdateAddUserGroup "User6", 777, "VER|FLT",
"BDomain" Project.SecurityUpdateAddUserGroup "User7", 333, "AEB",
"BDomain" Project.SecurityUpdateAddUserGroup "User10", 777, "ADM",
"BDomain" Project.SecurityUpdateCommit End Sub
```

See also

- SecurityUpdateCommit
- UpdateSystemSecurity
- SecurityUpdateUserParameter
- 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.

Call this method between SecurityUpdateStart and SecurityUpdateCommit.

An administrator needs to configure the group settings in the Web Verifier settings.

Syntax

```
SecurityUpdateUserParameter (BSTR UserName, BSTR UserDomain, BSTR
ParameterName, VARIANT Param1, VARIANT Param2)
```

Parameter	Description
ParameterName	PrimaryGroupID This parameter can have two variants: • Param1: "GroupName" with Param2 as String that represents the group name displayed in the Web Verifier administrator group settings. "ExternalGroupID" with Param2 as Integer that represents the ExternalGroupID that was added in SecurityUpdateAddUserGroup or SecurityUpdateAddUserGroupPwd.

Sample Code

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.

```
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
```

ShowValidationTemplates

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

Syntax

```
ShowValidationTemplates (pContainer as ISCBCdrPPGContainer)
```

Parameter	Description
pContainer	Container used to save the validation templates and their settings.

Unlock

This method unlocks the project after updating.

Syntax

```
Unlock ()
```

UpdateAddressPool

This method updates the address analysis pool.

Syntax

```
UpdateAddressPool ()
```

SCBCdrProject Properties

The Cedar Project object provides the following properties.

AllClasses

This read-only property returns a collection of all defined DocClasses of this Project.

Syntax

```
AllClasses as ISCBCdrDocClasses
```

See also

- SCBCdrDocClasses
- SCBCdrDocClass

BaseClasses

This read-only property returns a collection that contains all defined BaseDocClasses.

Syntax

```
BaseClasses as ISCBCdrDocClasses
```

See also

- SCBCdrDocClasses
- SCBCdrDocClass

ClassificationMode

This property sets returns the used classification mode.

Syntax

```
ClassificationMode as CDRClassifyMode
```

See also

CDRClassifyMode

CurrentClient

This property sets or returns the "Client" attribute of the batch.

Syntax

```
CurrentClient as String
```

DefaultClassifyResult

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

Syntax

```
DefaultClassifyResult as String
```

DefaultLanguage

This read-only property returns the language used as default.

Syntax

```
DefaultLanguage as String
```

Sample Code

```
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("Table", 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
```

Filename

This read-only property returns the project's path and file name.

Syntax

```
Filename as String
```

ForceValidation

This property sets or returns the Force Validation mode.

If ForceValidation is permitted, the user can overrule the validation by pressing Enter three times. If it is forbidden, the user cannot change the content of the field disregarding the validation rules.

Syntax

```
ForceValidation as CdrForceValidationMode
```

See also

CdrForceValidationMode

LastAddressPoolUpdate

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

Syntax

```
LastAddressPoolUpdate as Date
```

MinClassificationDistance

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

Syntax

```
MinClassificationDistance as Double
```

MinClassificationWeight

This property sets or returns the minimal classification weight.

Syntax

```
MinClassificationWeight as Double
```

MinParentClsDistance

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

Syntax

```
MinParentClsDistance as Double
```

MinParentClsWeight

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

Syntax

```
MinParentClsWeight as Double
```

NoUI

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

Syntax

```
NoUI as Boolean
```

Page

This read-only property returns the Cairo Page object of the current Project.

Syntax

```
Page as ISCBCroPage
```

ParentWindow

This write-only property sets the parent window of the login dialog box.

Syntax

```
ParentWindow as Long
```

SLWDifferentResultsAction

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

Syntax

```
SLWDifferentResultsAction as CdrSLWDifferentResultsAction
```

See also

CdrSLWDifferentResultsAction

SLWSupplierInvalidDifferentClsResults

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

Syntax

```
SLWSupplierInvalidIfDifferentClsResults as Boolean
```

ValidationSettingsColl

This read-only property returns a collection of all activated validation engines.

Syntax

```
ValidationSettingsColl as ISCBCroCollection
```

ValidationTemplates

This read-only property returns a collection of all available validation templates.

Syntax

```
ValidationTemplates as ISCBCroCollection
```

VersionCount

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

Syntax

```
VersionCount (Filename as String) as Long
```

Parameter	Description
Filename	Name of the file.

WordSegmentationChars

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

Syntax

```
WordSegmentationChars as String
```

SCBCdrDocClass

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

SCBCdrDocClass Methods

The SCBCdrDocClass object provides the following methods and properties.

GetFieldAnalysisSettings

This method returns the analysis settings for the document class.

Syntax

```
GetFieldAnalysisSettings (FieldName as String, Language as String,
ppAnalysisSettings as ISCBCdrAnalysisSettings)
```

Parameter	Description
FieldName	Name of the field for which the analysis settings are retrieved.
ppAnalysisSettings	Name of the analysis settings object used in the code to assign the settings to.

Sample Code

The following sample code shows how to get the analysis settings.

```
'To assign them for example to be used for the Associative Search Engine
Dim theDocClass as SCBCdrDocClass Dim theAnalysisSettings as
ISCBCdrAnalysisSettings 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
```

InitField

This method reinitializes a required field in workdoc.

Syntax

```
InitField (pWorkdoc as ISCBCdrWorkdoc, pField as ISCBCdrField)
```

Parameter	Description
pWorkdoc	Current workdoc.
pField	Field to clear.

ShowClassValidationDlg

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

Syntax

```
ShowClassValidationDlg (pContainer as ISCBCdrPPGContainer)
```

Parameter	Description
pContainer	Container in which the property page displays.

ShowFieldValidationDlg

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

Syntax

```
ShowFieldValidationDlg (FieldName as String, pContainer as
ISCBCdrPPGContainer)
```

Parameter	Description
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.

Syntax

```
ShowGeneralFieldPPG (FieldName as String, TabIndexActive as Long,
pContainer as ISCBCdrPPGContainer)
```

Parameter	Description
FieldName	Field for which the dialog box is shown.
TabIndexActive	Zero-based Index for the tab that displays.
pContainer	Container in which the property page displays.

SCBCdrDocClass Properties

The SCBCdrDocClass object provides the following methods and properties.

ClassificationField

Use this property to set or return the name of the field used for the classification.

Syntax

```
ClassificationField as String
```

ClassificationRedirection

This property to set or return the name of the target DocClass.

Syntax

```
ClassificationRedirection as String
```

ClassifySettings

This read-only property returns a collection of chosen classification engines and their settings for this DocClass.

Syntax

```
ClassifySettings as ISCBCroCollection
```

DerivedDocClasses

This read-only property returns a collection of all DocClasses derived directly from this DocClass.

Syntax

```
DerivedDocClasses as ISCBCdrDocClasses
```

DisplayName

This property sets or returns the display name of the DocClass currently not used. If the property is empty, the DocClass name is used.

Syntax

```
DisplayName as String
```

Fields

This read-only property provides access to FieldDefs of a DocClass.

Syntax

```
Fields as ISCBCdrFieldDefs
```

ForceSubtreeClassification

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

Syntax

```
ForceSubtreeClassification as Boolean
```

ForceValidation

This property sets or returns the Force Validation mode.

If ForceValidation is permitted, the user can overrule the validation by pressing Enter three times. If it is forbidden, the user cannot change the content of the field disregarding the validation rules.

Syntax

```
ForceValidation as CdrForceValidationMode
```

See also

CdrForceValidationMode

Hidden

This property sets or returns whether the DocClass is visible in the Project designer.

Syntax

```
Hidden as Boolean
```

ManualTableTrainingMode

This property sets or returns the option for manual table extraction training mode.

Syntax

```
ManualTableTrainingMode as Boolean
```

Name

This property sets or returns the name of the Document Class

Syntax

```
Name as String
```

Page

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

Syntax

```
Page as ISCBCroPage
```

Parent

This read-only property returns the parent DocClass of the actual DocClass.

Syntax

```
Parent as ISCBCdrDocClass
```

SubtreeClsMinDist

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

Syntax

```
SubtreeClsMinDist as Double
```

SubtreeClsMinWeight

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

Syntax

```
SubtreeClsMinWeight as Double
```

UseDerivedValidation

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

Syntax

```
UseDerivedValidation as Boolean
```

ValidationSettingsColl

This read-only property returns a collection of all activated validation engines

Syntax

```
ValidationSettingsColl as ISCBCroCollection
```

ValidationTemplateName

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

Syntax

```
ValidationTemplateName as String
```

ValidClassificationResult

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

Syntax

```
ValidClassificationResult as Boolean
```

VisibleInCorrection

This property sets or returns 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.

Syntax

```
VisibleInCorrection as Boolean
```

Sample Code

The following sample code shows how to dynamically modify the property of classes prior to showing the classification view.

The example below hides Invoices, BOLZ and UNICOM classes from verification availability.

```
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
```

SCBCdrDocClasses

This collection contains all defined DocClass objects of the Cedar project.

SCBCdrDocClasses Properties

The SCBCdrDocClasses object provides the following properties.

Collection

This read-only property returns the collection used internally to store the DocClasses.

Syntax

```
Collection as ISCBCroCollection
```

Count

This read-only property returns the number of items within the collection.

Syntax

```
Count as Long
```

Item

This read-only property returns a specified item from the collection.

Syntax

```
Item (Index as Variant) as ISCBCdrDocClass
```

Parameter	Description
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 read-only property returns an item from the collection specified by the index.

Syntax

```
ItemByIndex (Index as Long) as ISCBCdrDocClass
```

Parameter	Description
Index	Index of the item to get from the collection. **Possible values** 1 to Count

ItemByName

This read-only property returns an item from the collection specified by name.

Syntax

```
ItemByName (Name as String) as ISCBCdrDocClass
```

Parameter	Description
Name	[in] Name of the item to get from the collection.

ItemExists

This property returns TRUE if an item with the specified name exists in the collection.

Syntax

```
ItemExists (Name as String) as Boolean
```

Parameter	Description
Name	Name of item to search for.

ItemIndex

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

Syntax

```
ItemIndex (Name as String) as Long
```

Parameter	Description
Name	[in] Name specifying an item in the collection.

ItemName

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

Syntax

```
ItemName (Index as Long) as String
```

Parameter	Description
Index	Index specifying an item in the collection. **Possible values** 1 to Count

Tag

Use this property to set or return a variant for each item of the collection.

Syntax

```
Tag (Index as Long) as Variant
```

Parameter	Description
Index	Specifies the item index. **Possible values** 1 to Count

SCBCdrFieldDef

A Cedar FieldDef object represents the definition of a single FieldDef inside a Cedar DocClass

SCBCdrFieldDef Methods

The SCBCdrFieldDef object provides the following methods.

AppendListItem

This method adds a new list item and returns a new item index for it.

Syntax

```
AppendListItem (bstrItem as String) as Long
```

Parameter	Description
bstrItem	String inserted into the list.

Return value

Index of the new item.

GetAssignedAnalysisEngineName

This method returns the name of the assigned analysis engine or NULL if no engine is assigned.

Syntax

```
GetAssignedAnalysisEngineName(Language as String, EngineName as String)
```

Parameter	Description
Language	Language name **Note:** The default language of the project is "German".
EngineName	Name of the analysis engine

Sample Code

The following code example reads the assigned analysis engine name and displays it in a message box.

```
Private Sub Document_OnAction(pWorkdoc As SCBCdrPROJLib.ISCBCdrWorkdoc,
ByVal ActionName As String) Dim strEngineName As String If ActionName =
"ShowAnalysisEngineName" Then Project.AllClasses.ItemByName
("Invoices").Fields.ItemByName("Date").GetAssignedAnalysisEngineName
(Project.DefaultLanguage, strEngineName) MsgBox strEngineName End If End
Sub
```

RemoveListItem

This method removes a list item by its index.

Syntax

```
RemoveListItem (lIndex as Long)
```

Parameter	Description

lIndex	Index of entry to remove from the list.

Sample Code

```
Project.AllClasses.ItemByName("Invoice").Fields("Currency").RemoveListItem
(lngItem)
```

SCBCdrFieldDef Properties

The SCBCdrFieldDef object provides the following properties.

AllowDelayedValidation

This property sets or returns if "Allow delayed validation" is enabled.

Syntax

```
AllowDelayedValidation as Boolean
```

AllowInteractiveExtraction

This property sets or returns if table correction is enabled.

Syntax

```
AllowInteractiveExtraction as Boolean
```

Possible values

- True: Table correction in Verifier is enabled, but the table is not impacted by learning.
- False: Table correction in Verifier is disabled.

Sample Code

The following sample code looks at a field and determines if the LineItems (Brainware Table Extraction Field) are configured for automatic extraction.

```
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
```

AlwaysValid

This property sets or returns if the content of this FieldDef is always valid.

Syntax

```
AlwaysValid as Boolean
```

AnalysisTemplate

This read-only property returns the name of the analysis template if used.

Syntax

```
AnalysisTemplate (Language as String) as String
```

Parameter	Description
Language	Language parameter

BoostDigitsOnly

This property sets or returns if ’Boost digits only’ is enabled.

Syntax

```
BoostDigitsOnly as Boolean
```

BoostField

This property sets or returns if ’Boost field’ is enabled.

Syntax

```
BoostField as Boolean
```

ColumnCount

This read-only property returns the number of table columns if FieldType is table.

Syntax

```
ColumnCount as Long
```

ColumnName

This read-only property returns the name of the table column specified by an index if FieldType is table.

Syntax

```
ColumnName (ColumnIndex as Long) as String
```

Parameter	Description
ColumnIndex	Zero-based index of the table column

DefaultValidationSettings

This read-only property returns the validation settings with the default language.

Syntax

```
DefaultValidationSettings as ISCBCdrValidationSettings
```

Derived

This read-only property returns TRUE if the FieldDef properties are derived from an upper DocClass.

Syntax

```
Derived as Boolean
```

DisplayName

This property sets or returns the 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. A program can use the DisplayName instead of the FieldDef name to show a more readable name of the FieldDef.

Syntax

```
DisplayName as String
```

EvalSetting

This property sets or returns the activated evaluation engine and its settings.

Syntax

```
EvalSetting (Language as String) as Object
```

Parameter	Description
Language	Language parameter

EvalTemplate

This read-only property returns the name of the evaluation template if used.

Syntax

```
EvalTemplate (Language as String) as String
```

Parameter	Description
Language	Language of Project

FieldID

This read-only property returns the internally used FieldID.

Syntax

```
FieldID as Long
```

FieldType

This property sets or returns the type of the FieldDef.

Syntax

```
FieldType as CDRFieldType
```

See also

CDRFieldType

ForceValidation

This property sets or returns the Force Validation mode.

Syntax

```
ForceValidation as CdrForceValidationMode
```

See also

CdrForceValidationMode

ListItem

This property sets or returns a list item string for a given index.

Syntax

```
ListItem (lIndex as Long) as String
```

Parameter	Description
lIndex	Zero-based index.

ListItemCount

This read-only property returns the number of strings in the ListItem list.

Syntax

```
ListItemCount as Long
```

Sample Code

```
Dim lngItem as Long For lngItem = Project.AllClasses.ItemByName
("Invoice").Fields("Currency").ListItemCount - 1 To 0 Step -1
```

MaxLength

This property sets or returns the maximum number of characters permitted for this FieldDef.

Syntax

```
MaxLength as Long
```

MinLength

This property sets or returns the minimal number of characters for this FieldDef.

Syntax

```
MinLength as Long
```

Name

This property sets or returns the name of the FieldDef.

Syntax

```
Name as String
```

NoRejects

This property sets or returns if rejects are permitted.

Syntax

```
NoRejects as Boolean
```

OCRConfidence

This property sets or returns the confidence level for OCR.

Possible values

0 to 100

Syntax

```
OCRConfidence as Long
```

SmartIndex

This property sets or returns all definitions about smart indexing.

Syntax

```
SmartIndex as ISCBCdrSmartIndex
```

UseDerivedOCRSettings

This property sets or returns if the OCR settings of the parent DocClass are used.

Syntax

```
UseDerivedOCRSettings as Boolean
```

UseDerivedValidation

This property sets or returns if the derived validation rules are used for validation of this FieldDef.

Syntax

```
UseDerivedValidation as Boolean
```

UseMaxLen

This property sets or returns if the maximum number of characters is limited to the value given by MaxLength.

Syntax

```
UseMaxLen as Boolean
```

UseMinLen

This property sets or returns if the usage of the minimal number of characters given by the property MinLength is activated.

Syntax

```
UseMinLen as Boolean
```

ValidationSettings

This property sets or returns the chosen validation engine and its settings.

Syntax

```
ValidationSettings (Language as String) as ISCBCdrValidationSettings
```

Parameter	Description
Language	Defines the language for classification, extraction and validation.

ValidationTemplate

This read-only property returns the name of the validation template.

Syntax

```
ValidationTemplate (Language as String) as String
```

Parameter	Description
Language	Defines the language for classification, extraction and validation.

ValidationType

This read-only property returns the type of validation.

Syntax

```
ValidationType as CdrValFieldType
```

See also

CdrValFieldType

VerifierColumnWidth

This property sets or returns the width of the specified column of the table.

Syntax

```
VerifierColumnWidth (ColumnIndex as Long) as Long
```

Parameter	Description
ColumnIndex	Zero-based Index of the table column

SCBCdrFieldDefs

This collection contains all defined FieldDef objects of a single DocClass.

SCBCdrFieldDefs Properties

The SCBCdrFieldDefs object provides the following properties.

Collection

This read-only property returns the collection used internally to store the FieldDefs.

Syntax

```
Collection as ISCBCroCollection
```

Count

This read-only property returns the number of items within the FieldDef collection.

Syntax

```
Count as Long
```

Item

This read-only property returns a specified item from the collection.

Syntax

```
Item (Index as Variant) as ISCBCdrFieldDef
```

Parameter	Description
Index	[in] The index can either be a Long value specifying the index within the collection or a string specifying the item by name. **Possible values** • 1 to Count • Item name

ItemByIndex

This read-only property returns an item from the collection specified by index.

Syntax

```
ItemByIndex (Index as Long) as ISCBCdrFieldDef
```

Parameter	Description
Index	[in] Index of the item to get from the collection.

ItemByName

This read-only property returns an item from the collection specified by name.

Syntax

```
ItemByName (Name as String) as ISCBCdrFieldDef
```

Parameter	Description
Name	[in] Name of the item to get from the collection.

ItemExists

This property returns TRUE if an item with the specified name exists in the collection.

Syntax

```
ItemExists (Name as String) as Boolean
```

Parameter	Description
Name	[in] Name of item to search for.

ItemIndex

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

Syntax

```
ItemIndex (Name as String) as Long
```

Parameter	Description
Name	[in] Name specifying an item in the collection.

ItemName

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

Syntax

```
ItemName (Index as Long) as String
```

Parameter	Description
Index	Index specifying an item in the collection **Possible values** 1 to Count

Tag

Use this property to set or return a variant for each item of the collection.

Syntax

```
Tag (Index as Long) as Variant
```

Parameter	Description
Index	Specifies the item index. **Possible values** 1 to Count

SCBCdrLicenseInfoAccess

The Licensing Information Access object allows direct retrieval to the active licensing object. You can directly query any licensing component in a custom script.

SCBCdrLicenseInfoAccess Methods

The SCBCdrLicenseInfoAccess provides the following 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.

Syntax

```
GetLicenseCounterByID(CounterID as SCBCdrPROJLib.CDRLicenseCounter, Count
as Long, Active as Boolean)
```

Parameter	Description

CounterID	Counter ID 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 is active, or specified in the license file.

Sample Code

The following sample code returns the OCRed number of documents.

```
Dim theLicensingInterface2 as SCBCdrPROJLib.SCBCdrLicenseInfoAccessDim
theObject2 as Object Dim vValue2 as Long Dim vValue3 as Variant Dim
LicInfoMsg2 as String vValue2=0 vValue3=0 Set theObject2 = Project Set
theLicensingInterface2 = theObject2 '
theLicensingInterface2.GetLicenseCounterByID(TLCPeriodPagesOCRed, vValue2,
False) ' theLicensingInterface2.GetLicenseCounterByID(TLCTotalPagesOCRed,
vValue3, False) ' theLicensingInterface2.GetLicenseCounterByID
(TLCFineReaderRemainingUnits, vValue2, True)
theLicensingInterface2.GetLicenseCounterByName ("Overall OCRed Pages",
vValue2, True) LicInfoMsg2 = "OCRed count - " & CStr(vValue2) MsgBox
(LicInfoMsg2, vbOkOnly,"Get License Count By ID")
```

See also

- CdrLicenseCounter
- CdrLicenseFeatureName
- GetLicenseCounterByName
- GetLicenseValueByID
- GetLicenseValueByName

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.

Syntax

```
GetLicenseCounterByName(CounterName as String, Count as Long, Active as
Boolean)
```

Parameter	Description
CounterName	Counter name from which the value is retrieved. The name must match the one in the license file.

Count	The returned utilization value from the licensing mechanism. This stores the value of usage.
Active	Identifies if the license counter is active, or specified in the license file.

Sample Code

The following sample code returns the OCRed count of documents in script.

```
Dim theLicensingInterface as SCBCdrPROJLib.SCBCdrLicenseInfoAccess Dim
theObject as Object Dim vValue1 as Variant Dim LicInfoMsg as String Set
theObject = Project Set theLicensingInterface = theObject
theLicensingInterface.GetLicenseCounterByName("OCRed Pages per Day",
vValue1, True) LicInfoMsg = "OCRed count - " & CStr(vValue1)
```

See also

- CdrLicenseCounter
- CdrLicenseFeatureName
- GetLicenseCounterByID
- GetLicenseValueByID
- GetLiceneseValueByName

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.

Syntax

```
GetLicenseValueByID(PropertyID as SCBCdrPROJLib.CDRLicenseFeatureName,
Value as Variant)
```

Parameter	Description
PropertyID	Depicts the item for which to get 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.

Sample Code

The following sample code returns the Email Importing flag in the license file.

```
Dim theLicensingInterface as SCBCdrPROJLib.SCBCdrLicenseInfoAccess Dim
theObject as Object Dim vValue1 as Variant Dim LicInfoMsg as String Set
theObject = Project Set theLicensingInterface = theObject
theLicensingInterface.GetLicenseValueByID(CDRLfnEMailsImporting, vValue1)
LicInfoMsg = "Email Importing - " & CStr(vValue1) MsgBox(LicInfoMsg,
vbOkOnly,"Get License Value By ID")
```

See also

- CdrLicenseCounter
- CdrLicenseFeatureName
- GetLicenseCounterByID
- GetLicenseValueByID
- GetLiceneseValueByName

GetLicenseValueByName

This method returns the license counter information for any given item in the license file.

Syntax

```
GetLicenseValueByName(PropertyName as String, Value as Variant)
```

Parameter	Description
PropertyName	Property name on which to retrieve values. Various options can be found in the license file. The property name must match the property name in the license file.
Value	The returned value of the property. The data type varies depending on the item being returned.`

Sample Code

The following sample code returns the Email Importing flag in the license file.

```
Dim theLicensingInterface as SCBCdrPROJLib.SCBCdrLicenseInfoAccess Dim
theObject as Object Dim vValue1 as Variant Dim LicInfoMsg as String Set
theObject = Project Set theLicensingInterface = theObject
theLicensingInterface.GetLicenseValueByName("Serial", vValue1) LicInfoMsg
= "Primary Dongle Serial Number - " & CStr(vValue1) MsgBox(LicInfoMsg,
vbOkOnly,"Get License Value By Name")
```

See also

- CdrLicenseCounter
- CdrLicenseFeatureName
- GetLicenseCounterByID
- GetLicenseValueByID
- GetLiceneseValueByName

SCBCdrSettings

The Cedar Settings object stores arbitrary strings for usage in script.

SCBCdrSettings Methods

The SCBCdrSettings object provides the following methods.

AddClient

This method adds a new client with the specified name to the current Settings object.

Syntax

```
AddClient (newVal as String)
```

Parameter	Description
newVal	Name of new client

AddKey

This method adds a new key specified by its name and its parent.

Syntax

```
AddKey (newVal as String, Parent as String)
```

Parameter	Description
newVal	New key name
Parent	Name of the parent key. In case of a new base key, use an empty string for the Parent.

See also

- "Client keys" in the *Oracle WebCenter Forms Recognition Designer User's Guide*.

Clear

This method clears all clients and keys from the Settings object.

Syntax

```
Clear ()
```

MoveKey

This method moves a key specified by its name to the NewParent specified by its name.

Syntax

```
MoveKey (Key as String, NewParent as String)
```

Parameter	Description
Key	Name of key to move.
NewParent	Name of new parent, empty string in case of moving it as a base key.

RemoveClient

This method removes a client specified by its name.

Syntax

```
RemoveClient (ClientName as String)
```

Parameter	Description
ClientName	Name of client to remove

RemoveKey

This method removes a key specified by its name.

Syntax

```
RemoveKey (KeyName as String)
```

Parameter	Description
KeyName	Name of key to remove.

SCBCdrSettings Properties

The SCBCdrSettings object provides the following properties.

ActiveClient

This property sets or returns the name of the currently active client.

Syntax

```
ActiveClient as String
```

Client

This read-only property returns the name of the specified client.

Syntax

```
Client (Index as Long) as String
```

Parameter	Description
Index	Zero-based client index.

ClientCount

This read-only property returns the number of clients.

Syntax

```
ClientCount as Long
```

DocumentBinarizationMode

This property sets or returns the binarization mode for document conversion in the recognition engine.

Syntax

```
DocumentBinarizationMode as CdrDocumentBinarizationMode
```

See also

CdrDocumentBinarizationMode

GlobalLearnsetPath

This property sets or returns the global Learnset path.

Syntax

```
GlobalLearnsetPath as String
```

Key

This read-only property returns the key name specified by index.

Syntax

```
Key (Index as Long) as String
```

Parameter	Description
Index	Zero-based index of the key.

KeyCount

This read-only property returns the number of keys.

Syntax

```
KeyCount AS Long
```

KeyIcon

This property sets or returns the value for the specified key.

Syntax

```
KeyIcon (Key as String) as String
```

Parameter	Description
Key	Name of the key.

KeyParent

This read-only property returns the parent name of the specified key index.

Syntax

```
KeyParent (Index as Long) as String
```

Parameter	Description
Index	Zero-based key index.

ProjectFileName

This property sets or returns the file name of the project.

Syntax

```
ProjectFileName as String
```

SupervisedLearningDisabled

This property sets or returns the state of supervised learning in Designer and local Verifier workstations.

Syntax

```
SupervisedLearningDisabled as Boolean
```

TopDownEventSequence

This property sets or returns the value of a top-down event sequence.

Syntax

```
TopDownEventSequence as Boolean
```

Value

This property sets or returns the value of the specified key.

Syntax

```
Value (Key as String, Parent as String, Client as String) as String
```

Parameter	Description
Key	Key name, which is assigned to the value.
Parent	Parent name of the key.
Client	Name of the client. If this parameter is an empty string, the active client is used.

Sample Code

```
MyDBPath = Settings.Value(" DatabaseName ", " ", " ") 'now we can open the
database DB.Open(MyDBPath, …)
```

SCBCdrScriptModule

This is a global object at the project level. All script module events occurred at project level belong to this object.

SCBCdrScriptModule Methods

The SCBCdrScriptModule object provides the following methods.

ReadZone

Use this method to read a zone on a CroImage object, the settings of which were previously saved in the ScanJObs' definition.

Syntax

```
ReadZone (Image as ISCBCroImage, ZoneName as String) as String
```

Description	Definition
Image	[in] SCBCroImage object
ZoneName	[in] Name of Zone to read

ReadZoneEx

Use this method to read a zone on a CroImage object, the settings of which were previously saved in the ScanJobs' definition.

Syntax

```
ReadZoneEx (Image as ISCBCroImage, ZoneName as String, Result as
ISCBCroWorktext)
```

Description	Definition
Image	[in] SCBCroImage object
ZoneName	[in] Name of read zone
Result	[out] Result of reading returned as SCBCroWorktext object

SCBCdrScriptModule Properties

The SCBCdrScriptModule object provides the following property.
ModuleName

This read-only property returns the name of the module that initialized the 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

Syntax

```
ModuleName as String
```

Sample Code

The following sample code 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
```

The following function 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
```

SCBCdrScriptAccess

WebCenter Forms Recognition 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 WebCenter Forms Recognition custom script. Using this interface it is possible to get, modify and dump project and class level scripts.

SCBCdrScriptAccess Methods

The SCBCdrScriptAccess object provides the following methods.

DumpAllPages

This method dumps all script pages available in the project as a Unicode text file.

Syntax

```
DumpAllPages(FileName as String)
```

Parameter	Description
FileName	[in] name of the dump file.

Sample Code

The following sample code exports all script pages to a file (project and classes).

```
theScriptAccess.DumpAllPages("Script Export.txt")
```

ExportAllPages

CURRENTLY NOT SUPPORTED.

This method exports all available script pages in a reimportable format to the specified folder.

Syntax

```
ExportAllPages(FolderName as String)
```

Parameter	Description
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.

Syntax

```
ExportClassPage(FolderName as String, ClassName as String)
```

Parameter	Description
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 returns the project or specified class level script code.

Syntax

```
GetPageCode(ClassName as String, ScriptCode as String)
```

Parameter	Description

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.

Syntax

```
ImportAllPages(FolderName as String)
```

Parameter	Description
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.

Syntax

```
ImportClassPage(FolderName as String, ClassName as String)
```

Parameter	Description
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.

Syntax

```
SetPageCode(ClassName as String, ScriptCode as String)
```

Parameter	Description
ClassName	[in] name of the class.
ScriptCode	[out] class script code.

Sample Code

The following sample code sets the script code to blank.

```
theScriptAccess.SetPageCode(strClassName, "")
```

Working with the Associative Search Engine (CDRADSLib)

This library provides functions to work with the Associative Search engine.

SCBCdrSupExSettings Type Definitions

The SCBCdrSupExSettings object provides the following type definition.

CdrAutoUpdateType

This type definition specifies the automatic import property.

Type	Description
CdrAUTFile	Automatic import from file for the associative search field.
CdrAUTNone	No automatic import for the associative search field.
CdrAUTODBC	Automatic import from ODBC source for the associative search field.

SCBCdrSupExSettings Methods

The SCBCdrSupExSettings object provides the following methods.

AddColumn

This method adds a new column field to the pool.

Syntax

```
AddColumn (ColumnName as String, IsSearchField as Boolean, NewColumnIndex
as Long)
```

Parameter	Description
ColumnName	Name of the column field.
IsSearchField	**Possible values** • TRUE: The inserted column field is a search field. • FALSE: The inserted column field is not a search field.
NewColumnIndex	[out] Index of the newly created entry in the pool.

AddFilterAttribute

This method adds new filters for chosen attribute of the multi-column attribute search. Select attributes from the data source of the Associative Search Engine.

Note: Multiple entries for the same attribute are combined as logical OR, additional entries for different attributes are combined with logical AND.

Syntax

```
AddFilterAttribute(Key as String, Value as String)
```

Parameter	Description
Key	Name of the attribute to filter.
Value	Value of the attribute to search for in the data source.

Sample Code

The following sample code configures the multi-column attribute filtering to be used with the Vendor Search button in Verifier. The Vendor Search button in Verifier is related to the object: General, Process: DialogFunc.

```
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
```

AddPhrase

This method appends a new phrase to the list of phrases to use for the address.

Syntax

```
AddPhrase (Phrase as String, IsIncludePhrase as Boolean)
```

Parameter	Description
Phrase	[in] This string variable contains the phrase to add 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.

Syntax

```
ChangeEntry (ColumnName as String, EntryData as String)
```

Parameter	Description
ColumnName	[in] Name of the column to change.
EntryData	[in] The content of the specified column is updated with this data.

ClearFilterAttributes

This method clears all existing filters of the multi-column attribute filtering.

Syntax

```
ClearFilterAttributes()
```

Sample Code

```
Dim theSupplierSettings as Object Set theSupplierSettings =
FieldAnalysissettings Dim theAdsSettings as CDRADSLib.SCBCdrSupExSettings
Set theAdsSettings = theSupplierSettings
theAdsSettings.ClearFilterAttributes
```

CommitAddEntry

This method takes effect after execution of StartAddEntry and ChangeEntry.

Use this method only in context with the StartUpdate, StartAddEntry, ChangeEntry, and CommitUpdate methods.

Syntax

```
CommitAddEntry (NewIndex as Long)
```

Parameter	Description
NewIndex	[out] Index of new entry.

CommitUpdate

This method closes and save the currently opened pool.

Syntax

```
CommitUpdate ()
```

GeneratePool

This method imports the pool from the specified source by the property AutomaticImportMethod.

Syntax

```
GeneratePool ()
```

GeneratePoolFromCsvFile

This method removes the previous pool and generates a new one using CSV file designed in the new format.

Syntax

```
GeneratePoolFromCsvFile ()
```

Sample Code

```
Sub GeneratePoolData (pworkdoc As SCBCdrPROJLib.SCBCdrWorkdoc) ; Get a
handle for the field and settings Dim oCustomerASSA As SCBCdrField Dim
FieldAnalysissettings As ISCBCdrAnalysisSettings Dim PoolAnalysisSettings
As SCBCdrSupExSettings Dim SupplierExtractionEngine As New
SCBCdrSupplierExtractionEngine Set oCustomerASSA = pworkdoc.Fields
("VendorName") Project.AllClasses("Invoices").GetFieldAnalysisSettings
(oCustomerASSA.Name, "German", FieldAnalysissettings) Set
PoolAnalysisSettings = Project.AllClasses("Invoices").Fields
(oCustomerASSA.Name).AnalysisSetting("German") ; Using
PoolAnalysisSettings you can alter the attributes for the pool, such as
ImportFileName. PoolAnalysisSettings.GeneratePoolFromCSVFile ; Perform the
search SupplierExtractionEngine.SearchField("BOLZ Company, Gutman Company,
Inc, Worldnet", pworkdoc, FieldAnalysissettings, oCustomerASSA.Name) ;
Check if there is at least 1 result If oCustomerASSA.CandidateCount > 0
Then ; do something End If End Sub
```

GeneratePoolFromODBC

This method removes the previous pool and generates a new one using the ODBC source with the parameters set on the property page.

The method filters the full pool's records, prior to performing the ASE pool search.

Use this functionality with caution. If a Runtime Server or Verifier performs this operation, the possibility of the same pool folder being constantly filtered and altered will impact the performance of the project.

The Runtime Server performing this operation should be single threaded.

Syntax

```
GeneratePoolFromODBC ()
```

Sample Code

```
Sub GeneratePoolData (pworkdoc As SCBCdrPROJLib.SCBCdrWorkdoc) ; Get a
handle for the field and settings ; The field in the project file should
already be configured for ODBC Dim oCustomerASSA As SCBCdrField Dim
FieldAnalysissettings As ISCBCdrAnalysisSettings Dim PoolAnalysisSettings
As SCBCdrSupExSettings Dim SupplierExtractionEngine As New
SCBCdrSupplierExtractionEngine Set oCustomerASSA = pworkdoc.Fields
("VendorName") Project.AllClasses("Invoices").GetFieldAnalysisSettings
(oCustomerASSA.Name, "German", FieldAnalysissettings) Set
PoolAnalysisSettings = Project.AllClasses("Invoices").Fields
(oCustomerASSA.Name).AnalysisSetting("German") ; Modify the SQL query to
set your lookup criteria PoolAnalysisSettings.SQLQuery = "SELECT * FROM
VendorData WHERE Col002 in ('BOLZ', 'Gutman', 'WorldNet')" ; Perform the
search PoolAnalysisSettings.GeneratePoolFromODBC
SupplierExtractionEngine.SearchField("BOLZ Company, Gutman Company, Inc,
Worldnet", pworkdoc, FieldAnalysissettings, oCustomerASSA.Name) ; Check if
there is at least 1 result If oCustomerASSA.CandidateCount > 0 Then ;do
something End If End Sub
```

Note: The files in the pool folder will be reduced as they contain a smaller, filtered ADS pool.

GetClassNameByID

This method returns the formatted document class name for the pool entry specified by its unique ID.

Syntax

```
GetClassNameByID (IDHigh as Long, IDLow as Long, ClassName as String)
```

Parameter	Description
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 specified by its index and the column name.

Syntax

```
GetEntry (Index as Long, FieldName as String) as String
```

Parameter	Description
Index	[in] Index of the entry to get.
FieldName	[in] Name of the column to get.

GetFormattedValueByID

This method returns the formatted entry representation for the pool entry specified by its unique ID

Syntax

```
GetFormattedValueByID (IDHigh as Long, IDLow as Long, FormattedValue as
String)
```

Parameter	Description
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.

Syntax

```
GetIDByIndex (Index as Long, IDHigh as Long, IDLow as Long)
```

Parameter	Description
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.

Syntax

```
GetIndexByID (IDHigh as Long, IDLow as Long, Index as Long)
```

Parameter	**Description**
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.

Syntax

```
GetSearchArea (SearchAreaIndex as Long, Left as Long, Top as Long, Width
as Long, Height as Long)
```

Parameter	**Description**
SearchAreaIndex	Index of the area. **Possible values** 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 percent from left border of document.
Top	Distance in percent from top of document.
Width	Search area width in percent.
Height	Search area height in percent.

RemovePhrase

This method removes a phrase from a list of phrases for address analysis specified by its index.

Syntax

```
RemovePhrase (PhraseIndex as Long)
```

Parameter	Description
PhraseIndex	Zero-based index of the phrase to delete.

SetSearchArea

This method sets the area on the document in which to search.

Syntax

```
SetSearchArea (SearchAreaIndex as Long, Left as Long, Top as Long,  Width
as Long, Height as Long)
```

Parameter	Description
SearchAreaIndex	Index of the area. **Possible values** - 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 percent from left border of document.
Top	Distance in percent from top of document.
Width	Search area width in percent.
Height	Search area height in percent.

StartAddEntry

This method prepares the insertion of a new entry to the associative search pool.

Syntax

```
StartAddEntry ()
```

StartUpdate

This method generates and opens a new empty pool, or opens an existing pool for the update.

Syntax

```
StartUpdate (RemoveExistingPool as Boolean)
```

Parameter	Description
RemoveExistingPool	When this Boolean variable is set to true, 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. Moreover, in case this parameter is true, and the "AddColumn" method is invoked, the "AddColumn" method reports an error because it must be prohibited to modify the existing column.

SCBCdrSupExSettings Properties

The SCBCdrSupExSettings object provides the following property.

ClassNameFormat

This property sets or returns the format definition for a document class name.

Syntax

```
ClassNameFormat as String
```

ColumnCount

This read-only property returns the number of columns of a currently opened pool.

Syntax

```
ColumnCount as Long
```

ColumnName

This property sets or returns or sets the name of the column by its index.

Syntax

```
ColumnName (ColumnIndex as Long) as String
```

Parameter	Description
ColumnIndex	Zero-based index of the column.

EnableCandidateEvaluation

This property sets or returns if a candidate evaluation (so called Second Pass) is permitted.

For `EnableCandidateEvaluation`, 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.

Syntax

```
EnableCandidateEvaluation as Boolean
```

EntryCount

This read-only property returns the number of entries of the pool.

Syntax

```
EntryCount as Long
```

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.

Syntax

```
EvalFirstPageOnly as Boolean
```

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.

Syntax

```
EvalOverSearchAreas as Boolean
```

FieldContentsFormat

This property sets or returns the format definition for the representation of the engine.

Syntax

```
FieldContentsFormat as String
```

FindLocation

This property sets or returns if address analysis is enabled. If TRUE, the position of the address is found.

Syntax

```
FindLocation as Boolean
```

IdentityColumn

This property sets or returns the unique ID of a column name.

Syntax

```
IdentityColumn as String
```

ImportFieldNames

This property sets or returns if the column names are taken from the first line of a CSV file.

Syntax

```
ImportFieldNames as Boolean
```

ImportFileName

This property sets or returns if the column names are taken from the first line of a CSV file.

Syntax

```
ImportFileName as String
```

ImportFileNameRelative

This property sets or returns if the name of a CSV file is stored relative to the path of the project file.

Syntax

```
ImportFileNameRelative as Boolean
```

IsPhraseIncluded

This property sets or returns if a phrase to find the address is sufficient.

Syntax

```
IsPhraseIncluded (PhraseIndex as Long) as Boolean
```

Parameter	Description
PhraseIndex	[in] Index of phrase [zero-based].

IsSearchField

This property sets or returns if a field is used for an associative search.

Syntax

```
IsSearchField (ColumnIndex as Long) as Boolean
```

Parameter	Description
ColumnIndex	[in] Index of column [zero-based]

LastImportTimeStamp

This read-only property returns the time stamp for the last import.

Syntax

```
LastImportTimeStamp as Date
```

MaxCandidates

This property sets or returns the maximum number of results of the associative search engine.

Syntax

```
MaxCandidates as Long
```

MinDistance

This property sets or returns the required minimum distance to the next best candidate for a valid result.

Syntax

```
MinDistance as Double
```

MinRelevance

This property sets or returns the minimum relevance for search results, default value is 0.0.

Syntax

```
MinRelevance as Double
```

MinThreshold

This property sets or returns the required minimum value for a valid engine result.

Syntax

```
MinThreshold as Double
```

NumberKeepLocalCopies

Use this property to allow additional local copies of the ASE pool for later reuse.

The default value is 1. If the value is not modified, the system deletes any additional local pool copies when the application that created the copy is closed.

Note: The property is saved within the project once the script is executed. To change the property, modify and rerun the script.

Syntax

```
NumberKeepLocalCopies as Long
```

Sample Code

The following sample code sets the NumberKeepLocalCopies property to 10.

```
Dim theDocClass as SCBCdrDocClass Dim theSupplierSettings as
CDRADSLib.SCBCdrSupExSettings Dim theAnalysisSettings as
ISCBCdrAnalysisSettings Dim theObject as Object Set
theDocClass=Project.AllClasses.ItemByName("Invoices")
theDocClass.GetFieldAnalysisSettings
"VendorASSA",Project.DefaultLanguage,theAnalysisSettings Set theObject =
theAnalysisSettings Set theSupplierSettings = theObject
theSupplierSettings.NumberKeepLocalCopies = 10
```

ODBCName

This property sets or returns the name of the ODBC source.

Syntax

```
ODBCName as String
```

Passwords

This property sets or returns the password of the ODBC source.

Syntax

```
Password as String
```

Phrase

This property sets or returns the phrase by its index.

Syntax

```
Phrase (PhraseIndex as Long) as String
```

PhrasesCount

This read-only property returns the number of phrases used for address analysis.

Syntax

```
PhrasesCount as Long
```

PoolName

This property sets or returns the name of the associative search pool.

Syntax

```
PoolName as String
```

PoolPath

This property sets or returns the name of the path of the associative search pool.

Syntax

```
PoolPath as String
```

PoolPathRelative

This property sets or returns if the pool is saved relative to the path of the project.

Syntax

```
PoolPathRelative as Boolean
```

PoolVersion

This property sets or returns the pool version.

Syntax

```
PoolVersion as Long
```

Possible values

- 2: Brainware V2
- 3: Brainware V3
- 4: RMS

ProjectPath

This read-only property returns the path of the project file.

Syntax

```
ProjectPath as String
```

SavePoolInternal

This property sets or returns if a pool is saved within the project file or as separate files.

Syntax

```
SavePoolInternal as Boolean
```

SearchAreaActive

This property sets or returns if the corresponding search area is active or not.

Syntax

```
SearchAreaActive(SearchAreaIndex as Long) as Boolean
```

Parameter	Description
SearchAreaIndex	Index of the area. **Possible values** • 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.

Syntax

```
Separator as String
```

SQLQuery

This property sets or returns an SQL statement used to import ODBC source.

Syntax

```
SQLQuery as String
```

UserName

This property sets or returns the user name required to login in to the ODBC source.

Syntax

```
Username as String
```

VendorTypeColumn

This property sets or returns the column that defines the vendor type.

Possible values

- 0: No class is created for this vendor through SLW.
- 1: Allows one document for that vendor to be trained.
- 2: Allows unlimited training.

Syntax

```
VendorTypeColumn as String
```

SCBCdrSupplierExtractionEngine Methods

The SCBCdrSupplierExtractionEngine object provides the following method.

SearchField

This method searches for a text in a supplier pool and returns the result as field candidates.

Syntax

```
SearchField(ByVal QueryText As String, pWorkdoc As ISCBCdrWorkdoc,
pAnalysisSettings As ISCBCdrAnalysisSettings, ByValFieldName As String)
```

Parameter	Description
QueryText	The search text.
pWorkdoc	The current workdoc.
pAnalysisSettings	Provides access to the Supplier Extraction Engine settings needed to perform a search.
ByValFieldName	Field name that must exist in the workdoc, incorporates the search results as candidates.

Sample Code

```
Sub GeneratePoolData (pworkdoc As SCBCdrPROJLib.SCBCdrWorkdoc) ; Get a
handle for the field and settings ; The field in the project file should
already be configured for ODBC Dim oCustomerASSA As SCBCdrField Dim
FieldAnalysissettings As ISCBCdrAnalysisSettings Dim PoolAnalysisSettings
As SCBCdrSupExSettings Dim SupplierExtractionEngine As New
```

```
SCBCdrSupplierExtractionEngine Set oCustomerASSA = pworkdoc.Fields
("VendorName") Project.AllClasses("Invoices").GetFieldAnalysisSettings
(oCustomerASSA.Name, "German", FieldAnalysissettings) Set
PoolAnalysisSettings = Project.AllClasses("Invoices").Fields
(oCustomerASSA.Name).AnalysisSetting("German") ; Modify the SQL query to
set your lookup criteria PoolAnalysisSettings.SQLQuery = "SELECT * FROM
VendorData WHERE Col002 in ('BOLZ', 'Gutman', 'WorldNet')" ; Perform the
search PoolAnalysisSettings.GeneratePoolFromODBC
SupplierExtractionEngine.SearchField("BOLZ Company, Gutman Company, Inc,
Worldnet", pworkdoc, FieldAnalysissettings, oCustomerASSA.Name) ; Check if
there is at least 1 result If oCustomerASSA.CandidateCount > 0 Then ;do
something End If End Sub
```

Working with the Check Analysis Engine (SCBCdrParaCheckLib)

This class contains the functions for the Check Analysis engine.

SCBCdrParaCheckLib Type Definitions

The SCBCdrParaCheckLib object provides the following type definition.

- CROParaCheckPayeeRecognitionMode: This type definitions specifies the recognition mode.

Type	Description
CROParaCheckPayeeRecognitionModeAcc	This is the default mode. In this mode, the names in the vocabulary entries reflect the entire payee name information.
CROParaCheckPayeeRecognitionModeKeyWordSearch	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.

SCBCdrParaCheckAnalysisSettings Methods

The SCBCdrParaCheckAnalysisSettings provides the following methods.

AddItemToVocabulary

This method adds a vocabulary entry to the list of possible items.

Syntax

```
AddItemToVocabulary (Name as string, Weight as long, caption as
CroParaCheckVocabularyEntriesAddOptions, IsVisible as Boolean)
```

Parameter	Description
Name	**Possible values** - Payee Name: Provide the name of the possible payee candidates. - 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. Typically, most words have a weight of zero. A weight value of 15 corresponds to a vocabulary entry that appears in approximately 50 percent of the images.
caOption	**Possible values** - CroParaCheckVocabularyEntriesAddStatic: Add entry to the static vocabulary. - CroParaCheckVocabularyEntriesAddDynamic: Add entry to the dynamic vocabulary.
IsVisible	**Possible values** - True: The entry is visible in the GUI table in Designer. - False: The entry is not visible in the GUI table in Designer.

Sample Code

```
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
```

ClearVocabulary

Use this method to remove vocabulary entries.

Syntax

```
ClearVocabulary (clOption as CroParaCheckVocabularyEntriesClearOptions)
```

Parameter	Description

clOption	- CroParaCheckVocabularyEntriesClearStaticOnly: Remove all static entries - CroParaCheckVocabularyEntriesClearDynamicOnly: Remove all dynamic entries - CroParaCheckVocabularyClearEntriesAll: Remove all dynamic and static entries

Sample Code

```
Dim theFieldCheckAnalysisSettings as Object Set
theFieldCheckAnalysisSettings = FieldAnalysissettings Dim
ParaCheckSettings as SCBCdrParaCheckAnalysisSettings Set ParaCheckSettings
= theFieldCheckAnalysisSettings ParaCheckSettings.ClearVocabulary
CroParaCheckVocabularyEntriesClearDynamicOnly
```

SCBCdrParaCheckAnalysisSettings Properties

The SCBCdrParaCheckAnalysisSettings provides the following properties.

FieldType

This property sets or returns the field type.

Syntax

```
FieldType as Long
```

MinDistance

This property sets or returns the minimum distance.

Syntax

```
MinDistance as double
```

MinWeight

This property sets or returns the minimum weight.

Syntax

```
MinWeight as double
```

PayeeLineRecMode

This property sets or returns the engine recognition mode.

Syntax

```
PayeeLineRecMode as CROParaCheckPayeeRecognitionMode
```

See also

CROParaCheckPayeeRecognitionMode

PayeeVocCoverage

This property sets or returns the vocabulary coverage parameter for the payee line. This value expresses the occurrence likelihood of the predefined names on the checks.

Define the vocabulary coverage as a LONG value in the range between 1 and 100. The default value is 35.

Syntax

```
PayeeVocCoverage as Long
```

Sample Code

```
Private Sub Document_PreExtract(pWorkdoc as SCBCdrPROJLib.ISCBCdrWorkdoc)
Set ParaCheckSettings = Project.AllClasses(pWorkdoc.DocClassName).Fields
("Payee").AnalysisSetting(Project.DefaultLanguage)
ParaCheckSettings.PayeeVocCoverage = 85 End Sub
```

PayeeVocEntries

This read-only property returns the collection of vocabulary entries objects.

Syntax

```
PayeeVocEntries as ISCBCroPayeeVocEntries
```

Vocabulary Entry Objects (PayeeVocEntries)

This is a collection of all vocabulary entry objects contained in the current ParaCheckSettings object.

PayeeVocEntries Methods

The PayeeVocEntries object provides the following methods.

ItemExists

This method returns TRUE if an item with the specified name exists inside the collection.

Syntax

```
ItemExists (Name as String) as Boolean
```

Parameter	Description
Name	Name of item to search for.

MoveItem

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

Syntax

```
MoveItem (OldIndex as Long, NewIndex as Long)
```

Parameter	Description
OldIndex	Index of the item to move. **Possible values** 1 to Count
NewIndex	New index of the item after moving it. **Possible values** 1 to Count

Remove

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

Syntax

```
Remove (ItemName as String)
```

Parameter	Description
ItemName	Name of item to remove.

PayeeVocEntries Properties

The PayeeVocEntries object provides the following properties.

Count

This read-only property returns the number of items within the vocabulary entry collection.

Syntax

```
Count as Long
```

Item

This read-only property returns a specified item from the collection.

Syntax

```
Item (Index as Variant) as ISCBCroPayeeVocEntry
```

Parameter	Description

Index	The index can either be a long value specifying the index within the collection, or a string specifying the item by name. **Possible values** - 1 to Count - Item name

ItemByIndex

This read-only property returns an item from the collection specified by the index.

Syntax

```
ItemByIndex (Index as Long) as ISCBCroPayeeVocEntry
```

Parameter	Description
Index	Index of the item to get from the collection **Possible values** 1 to Count

ItemByName

This read-only property returns the specified item from the collection.

Syntax

```
ItemByName (Name as String) as ISCBCroPayeeVocEntry
```

Parameter	Description
Name	Name of the item to get from the collection.

ItemIndex

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

Syntax

```
ItemIndex (Name as String) as Long
```

Parameter	Description
Name	Name specifying an item in the collection.

ItemName

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

Syntax

```
ItemName (Index as Long) as String
```

Parameter	Description
Index	Index specifying an item in the collection **Possible values** 1 to Count

Tag

This property sets or returns a variant for each item of the collection.

Syntax

```
Tag (Index as Long) as Variant
```

Parameter	Description
Index	Specifies the item index. **Possible values** 1 to Count

Working with the Format Analysis Engine (SCBCdrFormatEngine)

This class provides methods and properties for the Format Analysis engine.

Sample code

The following sample code searches for a 1 to 10 digits long number in the workdoc by using the Simple Expression algorithm. If the word is found, the variable *pValid* is set to true.

```
Private Sub PoNo_Validate(pField as SCBCdrPROJLib.ISCBCdrField, pWorkdoc
as SCBCdrPROJLib.ISCBCdrWorkdoc, pValid as Boolean) Dim FormatEngine as
SCBCdrFormatEngine Dim oPOTemplate as SCBCdrFormatSettings If FormatEngine
Is Nothing Then Set FormatEngine = New SCBCdrFormatEngine If oPOTemplate
Is Nothing Then Set oPOTemplate = New SCBCdrFormatSettings
oPOTemplate.DeleteAll oPOTemplate.AddFormat("#[1-10]")
oPOTemplate.MaxDistance = 0.1 oPOTemplate.MaxWordCount = 5
oPOTemplate.MaxWordGap = 4 oPOTemplate.MaxWordLen = 100
oPOTemplate.KeepSpaces = False oPOTemplate.CaseSensitive = False
oPOTemplate.IgnoreCharacters(0) = ",.-/()[]" oPOTemplate.CompareType(0) =
CdrTypeSimpleExpression Dim strFoundString as String If
FormatEngine.FindStringFirst(pWorkdoc, oPOTemplate, , strFoundString) Then
 strFoundString = "Found" pField.Text = strFoundString pValid = True Else
pValid = False End If End Sub
```

Sample Code

The following sample code 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.

```
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 '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
```

Sample Code

The following sample code 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.

```
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 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
```

SCBCdrFormatEngine Methods

The SCBCdrFormatEngine provides the following methods.

FindStringFirst

This method finds the first word in the workdoc according to the pSettings parameter. The search starts from the first word in the workdoc and the first FormatSetting search pattern.

Note: The method does not take the region settings into account.

Syntax

```
FindStringFirst(pWorkdoc as ISCBCdrWorkdoc, pSettings as
SCBCdrFormatSettings, pFormatIndex as Long, pFoundString as String,
pWordID as Long) as Boolean
```

Parameter	Description
pWorkdoc	The current workdoc.
pSettings	The settings of the specific format string.
pFormatIndex	Output parameter that contains the index of the matched format setting if `pMatched` is true. contains the index of the format string that produced a successful match Optional parameter. The default value is 0.
pFoundString	Output parameter that contains the found string. Optional parameter. The default value is "0".
pWordID	Output parameter that contains the WordID of the best matching word in the workdoc. Optional parameter. The default value is 0.

Return value

- True: The string was found.
- False: The string was not found.

FindStringNext

This method finds the next word in the workdoc according to the pSettings parameter.

The method continues to search from the last word or format setting index found by the previous `FindStringFirst` or `FindStringNext` call.

Note: The method does not take the region settings into account.

Syntax

```
FindStringNext(pWorkdoc as ISCBCdrWorkdoc, pSettings as
SCBCdrFormatSettings, pFormatIndex as Long, pFoundString as String,
pWordID as Long) as Boolean
```

Parameter	Description
pWorkdoc	The current workdoc.
pSettings	The settings of the specific format string.
pFormatIndex	Output parameter that contains the index of the matched format setting if `pMatched` is true. Optional parameter. The default value is 0.
pFoundString	Output parameter that contains the found string. Optional parameter. The default value is "0".
pWordID	Output parameter that contains the index of the best matching word in the workdoc. Optional parameter. The default value is 0.

Return value

- True: The string was found.
- False: The string was not found.

See also

- FindStringFirst

GetBlockID

This method retrieves the workdoc block index for the current matched word. This method is called typically after executing `FindStringFirst` or `FindStringNext` for search operations where the entire block is needed, such as in an address block search algorithm.

Syntax

```
GetBlockID(plBlockID as Long)
```

Parameter	Description
plBlockID	Output parameter. • Contains a valuable result if the `AnalysisMethod` property is set to `CdrAnalysisString`. • Contains 0 (zero) if the `AnalysisMethod` property is set to `CdrAnalysisDesignator`.

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: (`StrLength-MatchedSubstrLength)/StrLength`.

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.

2.0

Syntax

```
TestString(bstrText as String, nFormatIndex as Long, pFormatSettings as
SCBCdrFormatSettings) as Single
```

Description	Definition
bstrText	The string to test against the Format String.
nFormatIndex	Zero-based index of the Format String list.
pFormatSettings	The settings of the specific Format String. See SCBCdrFormatSettings for more information.

SCBCdrFormatEngine Properties

The SCBCdrFormatEngine provides the following property.

SrchFlag

This bit flag property sets or returns 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.

Bit 3: Contains the trigram method. If set to `CdrTypeTrigram`, the new trigram method added in 5.8 is used, otherwise the trigram method of the version 5.7 and before is used.

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.

Syntax

```
SrchFlag (index as Long) as Long
```

Parameter	Description
Index	Zero-based index of the format string list. **Possible values** 0 to FormatCount-1

See also

- "Rules for string construction from words" in the *Oracle WebCenter Forms Recognition Designer User's Guide*.

Working with Format Settings (SCBCdrFormatSettings)

This class provides methods and properties to specify the format settings.

Sample code

The following sample code searches for the word *Invoice* or a very similar/misspelled word in the workdoc by using the Levenshtein algorithm. If the word is found, the variable *InvoiceFound* is set to true.

```
Dim FormatSettings As SCBCdrFormatSettings Dim FormatEngine as
SCBCdrFormatEngine Dim strStringFound as string Dim lngWordID as long Dim
InvoiceFound as Boolean Set FormatSettings = New SCBCdrFormatSettings
FormatSettings.AddFormat("Invoice") FormatSettings.CompareType(0) =
CdrTypeLevenShtein FormatSettings.AnalysisMethod(0) =
SCBCdrFormLib.CdrAnalysisString FormatSettings.MaxWordCount = 4
FormatSettings.MaxWordGap= 5.00 FormatSettings.MaxDistance = 0.35
FormatSettings.CaseSensitive = False FormatSettings.MaxWordLen = 150 If
FormatEngine.FindStringFirst(pWorkdoc,
FormatSettings,,strStringFound,lngWordID) Then InvoiceFound = true // The
word "Invoice" was found on the document else InvoiceFound = false // The
word "Invoice" was not found on the document end if
```

SCBCdrFormatSettings Type Definitions

The SCBCdrFormatSettings object provides the following type definitions.

CdrAnalysisMethod

This type definition specifies the analysis method.

Property	Description
CdrAnalysisString	The result contains the matched string.

CdrAnalysisDesignator	The result contains the information positioned to the matched string, where the position is defined by DesignatorType.

CdrDesignatorType

This type definition specifies the designator type.

Property	Description
CdrDesignatorEndOfLine	The compare operation returns all words between the search string and the end of the line.
CdrDesignatorNextBlock	The compare operation returns the block below the search string.
CdrDesignatorNextLine	The compare operation returns the line below the search string.
CdrDesignatorNextParagraph	The compare operation returns the paragraph below the search string.
CdrDesignatorNextWord	The compare operation returns the word right to the search string.
CdrDesignatorPrevWord	The compare operation returns the word left to the search string.
CdrDesignatorThisBlock	The compare operation returns the block that contains the search string.
CdrDesignatorWordAbove	The compare operation returns the word above the search string.
CdrDesignatorWordBelow	The compare operation returns the word below the search string.

SCBCdrFormatSettings Methods and Properties to Modify the List of Format Strings

Use the following methods and properties to modify the list of format strings.

AddFormat

This method adds the format string to the list of format strings.

Syntax

```
AddFormat(newVal as String)
```

Parameter	Description
newVal	Format string to add to the list of format strings.

DeleteAll

This method removes all format strings from the list of format strings.

Syntax

```
DeleteAll()
```

DeleteFormat

This method deletes the specified format string from the list of format strings.

Syntax

```
DeleteFormat(Index as Long)
```

Parameter	Description
Index	Index of the format string to delete.

FormatCount

This read-only property returns the number of format strings in the list of format strings.

Syntax

```
FormatCount as Long
```

MoveFormat

This method moves a format string to a new position in the list of format strings.

Syntax

```
MoveFormat(OldIndex as Long, NewIndex as Long)
```

Parameter	Description
OldIndex	Current index of the format string.
NewIndex	New index of the format string.

SCBCdrFormatSettings Methods and Properties to Specify the Settings for each Format String

Use the following properties to specify the settings for each format string.

AllowMultilineCandidates

This property is reserved for future use and should not be used in the current version.

Syntax

```
AllowMultilineCandidates as Boolean
```

AnalysisMethod

This property sets or returns the analysis method for the specified format string from the list of format strings.

Syntax

```
AnalysisMethod(Index as Long) as CdrAnalysisMethod
```

Parameter	Description
Index	Index of the format string.

Sample Code

```
FormatSettings.AnalysisMethod(0) = SCBCdrFormLib.CdrAnalysisString
```

See also

CdrAnalysisMethod

CompareType

This property sets or returns the string compare algorithm which is used for the specified format string from the list of format strings.

Syntax

```
CompareType(Index as Long) as CdrCompareType
```

Parameter	Description
Index	Index of the format string.

Return value

The string compare algorithm.

See also

CdrCompareType

DesignatorType

This property sets or returns the DesignatorType for the specified format string from the list of format strings.

Syntax

```
DesignatorType(Index as Long) as CdrDesignatorType
```

Parameter	Description
Index	Index of the format string.

See also

CdrDesignatorType

FormatString

This property sets or returns the specified format string in/from the list of format strings.

Syntax

```
FormatString (index as Long) as String
```

Parameter	Description
Index	Index of the format string.

FormatValid

This read-only property returns the result of the validation check for the specified format string from the list of format strings.

Syntax

```
FormatValid(Index as Long)as Boolean
```

Parameter	Description
Index	Index of the format string.

IgnoreCharacters

This property sets or returns the ignore character string for the specified format string from the list of format strings. The ignore character string contains the characters or symbols which are ignored in the search algorithm.

Syntax

```
IgnoreCharacters(Index as Long) as String
```

Parameter	Description
Index	Index of the format string.

Prefix

This property sets or returns the prefix string for the specified format string from the list of format strings. The prefix string contains the characters or symbols which are ignored when found at the beginning of the word.

Syntax

```
Prefix(Index as Long) as String
```

Parameter	Description
Index	Index of the format string.

Suffix

This property sets or returns the suffix string for the specified format string from the list of format strings. The suffix string contains the characters or symbols which are ignored when found at the end of the word.

Syntax

```
Suffix(Index as Long) as String
```

Parameter	Description
Index	Index of the format string.

SCBCdrFormatSettings Methods and Properties to Specify the General Search Settings

Use the following methods and properties to specify the general search settings.

CaseSensitive

This property sets or returns if the search operation works case-sensitive.

Syntax

```
CaseSensitive as Boolean
```

KeepSpaces

This property sets or returns if the engine keeps the spaces between the connected words.

Syntax

```
KeepSpaces as Boolean
```

MaxDistance

This property sets or returns the maximal compare distance. A match requires that the actual compare distance is less or equal to the maximum compare distance.

Syntax

```
MaxDistance as Double
```

MaxWordCount

This property sets or returns the maximum number of words combined as input for the search operation. Words which are combined must be in the same line.

Syntax

```
MaxWordCount as Long
```

MaxWordGap

This property sets or returns the maximum distance in millimeters that permits word concatenation during the search. The requirements strongly depend on the font size.

Syntax

```
MaxWordGap as Double
```

MaxWordLen

This property sets or returns the maximum overall word length in millimeters on the document of the combined input string for the search engine.

Syntax

```
MaxWordLen as Double
```

ResetTranslationLanguage

This method sets the transliteration approach to `CDRTranslationLanguageDefault`, which means that no transliteration executes for the format string, ignore char string, prefix and suffix string. This is the default setting and should be used in case the workdoc was not transliterated. This method has no impact to the methods `FindStringFirst` and `FindStringNext`.

Syntax

```
ResetTranslationLanguage()
```

SettingsCheckSum

This property sets or returns the checksum string created during extraction for the format settings object for the field. The checksum is not stored in the storage object. Checksum is used in WebCenter Forms Recognition internally. This property has no impact to the methods `FindStringFirst` and

`FindStringNext`.

Syntax

```
SettingsCheckSum as String
```

SetTranslationLanguage

This method specifies the transliteration approach for the format analysis engine. In general, use the same approach as used for the workdoc's OCR text. This method has no impact to the methods `FindStringFirst` and `FindStringNext`.

Syntax

```
SetTranslationLanguage(Language as CDRTranslationLanguage)
```

Parameter	**Description**
Language	Transliteration approach **Possible values** See CDRTranslationLanguage

SCBCdrFormatSettings Methods and Properties to Specify the Settings for the Search Regions

Use the following properties to specify the settings for the search regions.

BottomFirst

This property sets or returns the search ending position from the bottom of the document's first page as a percentage value between 0 and 100.

Note: This property has no impact to the methods `FindStringFirst` and `FindStringNext`.

Syntax

```
BottomFirst as Long
```

BottomLast

This property sets or returns the search ending position from the bottom of the document's last page as a percentage value between 0 and 100. This property has no impact to the methods `FindStringFirst` and `FindStringNext`.

Syntax

```
BottomLast as Long
```

BottomSubseq

This property sets or returns the search ending position from the bottom of the document's subsequent pages as a percentage value between 0 and 100.

Note: This property has no impact to the methods `FindStringFirst` and `FindStringNext`.

Syntax

```
BottomSubseq as Long
```

LeftFirst

This property sets or returns the search starting position from the left of the document's first page as a percentage value between 0 and 100. This property has no impact to the methods `FindStringFirst` and `FindStringNext`.

Syntax

```
LeftFirst as Long
```

LeftLast

This property sets or returns the search starting position from the left of the document's last page as a percentage value between 0 and 100. This property has no impact to the methods `FindStringFirst` and `FindStringNext`.

Syntax

```
LeftLast as Long
```

LeftSubseq

This property sets or returns the search starting position from the left of the document's subsequent pages as a percentage value between 0 and 100. This property has no impact to the methods `FindStringFirst` and `FindStringNext`.

Syntax

```
LeftSubseq as Long
```

RightFirst

This property sets or returns the search ending position from the right of the document's first page as a percentage value between 0 and 100. This property has no impact to the methods `FindStringFirst` and `FindStringNext`.

Syntax

```
RightFirst as Long
```

RightLast

This property sets or returns the search ending position from the right of the document's last page as a percentage value between 0 and 100. This property has no impact to the methods `FindStringFirst` and `FindStringNext`.

Syntax

```
RightLast as Long
```

RightSubseq

This property sets or returns the search ending position from the right of the document's subsequent pages as a percentage value between 0 and 100. This property has no impact to the methods `FindStringFirst` and `FindStringNext`.

Syntax

```
RightSubseq as Long
```

UseFirstPage

This property sets or returns if the engine searches on first page of the document. This property has no impact to the methods `FindStringFirst` and `FindStringNext`.

Syntax

```
UseFirstPage as Boolean
```

UseLastPage

This property sets or returns if the engine searches on the last page of the document. This property has no impact to the methods `FindStringFirst` and `FindStringNext`.

Syntax

```
UseLastPage as Boolean
```

UseRegions

This property sets or returns if the engine searches on specific regions of the document only. This property has no impact to the methods `FindStringFirst` and `FindStringNext`.

Syntax

```
UseRegions as Boolean
```

UseSubseqPage

This property sets or returns if the engine searches on subsequent pages of the document. This property has no impact to the methods `FindStringFirst` and `FindStringNext`.

Syntax

```
UseSubseqPage as Boolean
```

TopFirst

This property sets or returns the search starting position from the top of the document’s first page as a percentage value between 0 and 100. This property has no impact to the methods `FindStringFirst` and

`FindStringNext`.

Syntax

```
TopFirst as Long
```

TopLast

This property sets or returns the search starting position from the top of the document’s last page as a percentage value between 0 and 100. This property has no impact to the methods `FindStringFirst` and `FindStringNext`.

Syntax

```
TopLast as Long
```

TopSubseq

This property sets or returns the search starting position from the top of the document’s subsequent pages as a percentage value between 0 and 100. This property has no impact to the methods `FindStringFirst` and `FindStringNext`.

Syntax

```
TopSubseq as Long
```

Working with Batches (SCBCdrBATCHLib)

This library provides classes and types to work with batches.

SCBCdrBatchRoot Methods

The SCBCdrBatchRoot object provides the following methods for batch administration.

Connect

This method connects to the batch root.

Syntax

```
Connect(BatchRootPath as String, ImageRootPath as String, UserName as
String Password as String, ModuleType as String)
```

Parameters

Parameter	**Description**
BatchRootPath	Batch Root Path If ActivateSupport is True, the parameter can be an empty string.
ImageRootPath	Image Root Path If ActivateSupport is True, the parameter can be an empty string.

UserName	The user that can connect to the database job. In most cases it is applicable to use the special name "LOGIN_AS_CURRENT" in order to reuse the current connection to the database.
Password	Password If UserName = LOGIN_AS_CURRENT, the password can be an empty string.
ModuleType	Type of the module connecting

Sample code

```
Dim pBatchRoot as New SCBCdrBATCHLib.SCBCdrBatchRoot … pBatchRoot.Connect
("Job name", "", "LOGIN_AS_CURRENT", "", "Module type name")
```

Disconnect

This method disconnects the batch from the batch root.

Syntax

```
Disconnect()
```

SetConnectionProperties

Use this method to specify the used job and the instance name.

Syntax

```
SetConnectionProperties(SelectedJobName as String, InstanceName as String,
CreateJobIfNotExist as Boolean)
```

Parameters

Parameter	Description
SelectedJobName	Job name from the JOBS table.
InstanceName	RTS instance used to connect.
CreateJobIfNotExist	True: Create the job if not present. False: Use an existing job only.

Sample code

```
Dim pBatchRoot as New SCBCdrBATCHLib.SCBCdrBatchRoot …
theBatchRoot.SetConnectionProperties "My DB Job", "MyRTS", False
```

SetFilter

This method sets the filter for the batch root. By default, all batches from the job are available at the batch root level. This method allows you to select only batches of a specific state.

Syntax

```
SetFilter(State as Long)
```

Parameter	Description
State	State of the batch you want to access in the batch root level.

SetLoginProperties

This method sets the login credentials for the project database user.

Syntax

```
SetLoginProperties(UserName as String, Password as String, ModuleType as
String, InstanceName as String)
```

Parameters

Parameter	Description
UserName	User name
Password	Password
ModuleType	Type of the module connecting
InstanceName	RTS instance used to connect

Sample code

```
Dim pBatchRoot as New SCBCdrBATCHLib.SCBCdrBatchRoot …
  theBatchRoot.SetLoginProperties "username", "password", "Server", "MyRTS"
```

SCBCdrBatchRoot Properties

The SCBCdrBatchRoot object provides the following properties for batch administration.

ActivateSupport

This property sets or returns if database support for the batch connection is enabled.

Syntax

```
ActivateSupport as Boolean
```

Sample code

```
Dim pBatchRoot as New SCBCdrBATCHLib.SCBCdrBatchRoot
```

```
pBatchRoot.ActivateSupport = True
```

BatchCount

This read-only property returns the number of batches according to the filter conditions applied by the SetFilter method.

Syntax

```
BatchCount as Long
```

See also

SetFilter

BatchId

This read-only property returns the batch id as a string. The index of the batch is influenced by the filter conditions.

Syntax

```
BatchID(BatchIndex as Long)
```

Parameter	Description
BatchIndex	Index of the batch.

See also

SetFilter

Comparing Strings (SCBCdrSTRCOMPLib)

This library provides several implementations of string compare algorithms.

SCBCdrSTRCOMPLib Type Definitions

The SCBCdrStringComp object provides the following type definition.

CdrCompareType

This table contains a list of string compare algorithms.

Type	Description
CdrTypeLevenShtein	Levenshtein algorithm
CdrTypeRegularExpression	Regular expression
CdrTypeSimpleExpression	Simple expression

CdrTypeStringComp	Exact string compare
CdrTypeTrigram	Trigram algorithm

SCBCdrStringComp Methods

The SCBCdrStringComp object provides the following methods.

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.

Syntax

```
Distance (String as String, Distance as Double)
```

Parameter	Description
String	The string to compare with the search expression.
Distance	[out] Distance of the compare operation. **Possible values** Any value between 0.0 and 1.0.

ValidateSearchExpression

This method performs a syntax check for the specified compare method and search expression.

Syntax

```
ValidateSearchExpression (Type as CdrCompareType, SearchExpression as
String) as Boolean
```

Parameter	Description
Type	Compare method to use for validation.
SearchExpression	Search expression to validate. **Possible values** CdrCompareType

SCBCdrStringComp Properties

The SCBCdrStringComp object provides the following properties.

CaseSensitive

This property sets or returns if the compare algorithm works case-sensitive.

Syntax

```
CaseSensitive as Boolean
```

CompTypes

This property sets or returns the compare algorithm used for the next call of Distance.

Syntax

```
CompType as CdrCompareType
```

See also

CdrCompareType

LevDeletions

This read-only property returns the count of deletions calculated by the last Distance function.

Syntax

```
LevDeletions as Single
```

LevInsertions

This read-only property returns the count of insertions calculated by the last Distance function.

Syntax

```
LevInsertions as Single
```

LevRejects

This read-only property returns the count of rejects calculated by the last Distance function.

Syntax

```
LevRejects as Single
```

LevReplacements

This read-only property returns the count of replacements calculated by the last Distance function.

Syntax

```
LevReplacements as Single
```

LevSame

This read-only property returns the count of equal characters calculated by the last Distance function.

Syntax

```
LevSame as Single
```

LevTraceMatrix

This read-only property returns the Levenshtein trace matrix calculated by the last Distance function.

Syntax

```
LevTraceMatrix as String
```

LevTraceResult

This read-only property returns the Levenshtein trace result calculated by the last Distance function.

Syntax

```
LevTraceResult as String
```

MatchEndPosition

This read-only property returns the matching end position calculated by the last Distance function.

Syntax

```
MatchEndPosition as Single
```

MatchStartPosition

This read-only property returns the matching start position calculated by the last Distance function.

Syntax

```
MatchStartPosition as Single
```

SearchExpression

This property sets or returns the search expression used for the next compare operation.

Syntax

```
SearchExpression as String
```

Working with Images (SCBCroIMGLib)

The SCBCroIMGLib provides the following objects to work with images.

- SCBCroImage

- SCBCroViewer

SCBCroIMGLib Type Definitions

The SCBCroImage object provides the following type definitions.

CroDocToView

This type definition specifies how the document coordinates are transferred into the viewer coordinates.

Type	Description
CroViewPixX	Viewer window coordinate x in pixels
CroViewPixY	Viewer window coordinate y in pixels

CROImgIOMode

This type definition specifies the I/O mode.

Type	Description
CROImgIOAsync	Reserved for future enhancements.
CROImgIOSync	Default value. I/O mode Sync The function waits until saving of the image has finished.

CROImgFileTypes

This type definition specifies the image file type.

Type	Description
CROcBMP	Microsoft Windows Bitmap (.bmp) Uncompressed monochrome or 1- to 32-bit color binary bitmap **Multiple Images**: No
CROcBMP_ RLE	Microsoft Windows Bitmap (.bmp) RLE-compressed monochr. or 1- to 32-bit color binary bitmap **Multiple Images**: No

CROcBRK_G3	Brooktrout file (.brk) Group 3 (G3) compressed monochrome (1-bit) file **Image Type**: Bitmap **Data Encoding**: Binary **Multiple Images**: No
CROcBRK_ G3_2D	Brooktrout file (.brk), Group 3 (G3)
CROcCAL	CALS Raster (.CAL, .RAS, .CALS) Monochrome bitmap, to provide a standardized graphics interchange for electronic publishing, such as technical graphics, CAD/CAM, and image processing applications.
CROcCLP	Windows Clipboard (.CLP) RLE or uncompressed up to 24-bit color bitmap **Multiple Images**: No Used to hold any of several data types that can be stored by the Windows Clipboard
CROcDCX	DCX (.DCX), RLE Monochrome Color table (16 or 256 entries) 24-bit RGB Bitmap **Multiple Images**: Yes Allows to store multiple PCX files in one file.
CROcEPS	Encapsulated PostScript (.EPS), (.EPI) 1-bit monochrome MetafileStorage and interchange of single images (up to one page) across a wide range of platforms.
CROcICA_G3	Image Object Content Architecture .(ICA) Monochrome, 4, 8, 16-bit palette gray-scale and color, RGB, YCrCb, & YCbCr Bitmap, G3 compr. **Multiple Images**: No
CROcICA_G4	Image Object Content Architecture .(ICA) Monochrome, 4, 8, 16-bit palette gray-scale and color, RGB, YCrCb, & YCbCr Bitmap, G4 compr. **Multiple Images**: No

CROcICO	ICOn resource file(.ICO), Monochrome, 3, 4, 8, 24-bit palette Bitmap, **Multiple Images**: Yes Storage and interchange of iconic bitmaps (icons), across many Windows applications, regardless of the pixel resolution and color scheme of the display hardware.
CROcIFF	Electronic Art’s Interchange File Format (.iff) RGB palette up to 24-bit, binary Bitmap **Multiple Images**: No RLE or uncompressed
CROcIMT	IMNET graphics,
CROcJPG	JPEG File Interchange Format (.JPG) 8-bit gray scale or 24-bit YCrCb color 2D raster, JPEG compr. **Multiple Images**: No Interchange of .JPG files between applications on different platforms.
CROcLDF	Lura Document format
CROcMOD_G3	Mixed Object Document Content Architecture Monochrome, 4, 8, 16-bit palette gray-scale and color, RGB, YCrCb, & YCbCr Bitmap **Multiple Images**: Yes G3 compression Allows multiple IOCA images to be stored in one file.
CROcMOD_G4	Mixed Object Document Content Architecture Monochrome, 4, 8, 16-bit palette gray-scale and color, RGB, YCrCb, & YCbCr Bitmap **Multiple Images**: Yes G4 compression Allows to store multiple IOCA images in one file.
CROcNCR	NCR Image
CROcNCR_G4	NCR Image, compression G4
CROcPCT	Macintosh PICT (.PCT) Monochrome, 48-bit color palette, up to 24-bit RGB metafile **Multiple Images**: No Used to capture a picture as a set of Macintosh QuickDraw library function calls

CROcPCX	PC Paintbrush File Format (.PCX) Monochrome, 2 - 8-bit color table, 24-bit RGB Bitmap **Multiple Images**: No Storing images for ZSoft Corporation's paint program, such as PC Paintbrush
CROcPNG	Portable Network Graphics (.PNG) Truecolor up to 48 bpp, Grayscale up to 16 bpp, Palette up to 256 colors Bitmap / raster, **Multiple Images**: No Effective lossless compression of truecolor images'.
CROcPSD	Adobe Photoshop (.PSD), all colors including RGB, CMYK, multi-channel Bitmap Multiple Images: NoStoring bitmaps in the Adobe Photoshop graphical editing application.
CROcRAS	Sun Raster Data Format (.RAS), 1-bit monochrome, 8-bit gray scale or colormapped, 24 or 32-bit RGB colormap or Truecolor Bitmap **Multiple Images**: No Bitmap format for Sun UNIX platforms.
CROcSGI	SGI Image File Format (.SGI), RLE or umcompr. monochrome, gray-scale, and color including RGB, and RGB with alpha channel Bitmap **Multiple Images**: No Display of images from the SGI image library
CROcTGA	Truevision Targa (.TGA), RLE (usually uncomp.) 8, 16, 24-bit palette or RGB, Alpha data 1 or 8-bits Bitmap (2D raster)
CROcTIF	Tagged Image File Format (.TIF), Monochrome, 4 to 8-bit gray scale, 24-bit palette or RGB Bitmap **Multiple Images**: Yes Standardized data storage and interchange of images obtained from scanners and incorporated into desktop publishing
CROcTIF_G3	Tagged Image File Format (.TIF), Monochrome, 4 to 8-bit gray scale, 24-bit palette or RGB Bitmap **Multiple Images**: Yes G3 compr. Standardized data storage and interchange of images obtained from scanners and incorporated into desktop publishing

CROcTIF_G3_2D	Tagged Image File Format (.TIF), Monochrome, 4 to 8-bit gray scale, 24-bit palette or RGB Bitmap **Multiple Images**: Yes G3 compr., 2-dim. Standardized data storage and interchange of images obtained from scanners and incorporated into desktop publishing
CROcTIF_G4	Tagged Image File Format (.TIF), Monochrome, 4 to 8-bit gray scale, 24-bit palette or RGB Bitmap **Multiple Images**: Yes G4 compr. Standardized data storage and interchange of images obtained from scanners and incorporated into desktop publishing
CROcTIF_HUF	Tagged Image File Format (.TIF), Monochrome, 4 to 8-bit gray scale, 24-bit palette or RGB Bitmap **Multiple Images**: Yes Huffmann compr. Standardized data storage and interchange of images obtained from scanners and incorporated into desktop publishing
CROcTIF_JPG	Tagged Image File Format (.TIF), Monochrome, 4 to 8-bit gray scale, 24-bit palette or RGB Bitmap **Multiple Images**: Yes JPEG compr. Standardized data storage and interchange of images obtained from scanners and incorporated into desktop
CROcTIF_LZW	Tagged Image File Format (.TIF), Monochrome, 4 to 8-bit gray scale, 24-bit palette or RGB Bitmap **Multiple Images**: Yes LZW compr. Standardized data storage and interchange of images obtained from scanners and incorporated into desktop publishing
CROcTIF_PACKED	Tagged Image File Format (.TIF), Monochrome, 4 to 8-bit gray scale, 24-bit palette or RGB Bitmap **Multiple Images**: Yes Packed standardized data storage and interchange of images obtained from scanners and incorporated into desktop publishing
CROcWMF	Microsoft Windows Metafile (.WMF), Monochrome, Color lookup table, RGB up to 24-bit Metafile (bitmap and vector) **Multiple Images**: No Uncompressed. Convenient storage and interchange for applications under MS Windows.

CROcXBM	X BitMap (.xbm), Monochrome Bitmap **Multiple Images**: Yes Uncompressed. Primarily for storing cursor and icon bitmaps for the X System graphical user interface.
CROcXPM	X PixMap (.XPM), monochrome, 2-bit and up gray scale, all colors Bitmap **Multiple Images**: Yes Uncompressed. To store X Window Pixmap information to disk
CROcXWD	X Window Dump (.xwd), monochrome, 2 - 15-bit palette, 16, 24 or 32 TrueColor Bitmap **Multiple Images**: No Uncompressed. Storing screen dumps from the X Window System

CROImgType

This type definition specifies the image color type.

Type	Description
CROcColor24	24-bit color
CROcColor4	4-bit color
CROcColor8	8-bit color
CROcGray1	1-bit gray
CROcGray4	4-bit gray
CROcGray8	8-bit gray
CROcNone	No image type specified

CROLDFLayer

This type definition specifies the Lura Document Format layer.

Type	Description
CRO_LDFAll	All layers are displayed
CRO_LDFImage	The Image layer is displayed
CRO_LDFNone	None of the layers are displayed
CRO_LDFText	The Text layer is displayed
CRO_LDFTextColor	The Text Color layer is displayed

CROTiffTagType

This type definition specifies the data types of the various Tiff tags.

Type	Description
CROTiffTagASCII	Tag is of type ASCII
CROTiffTagByte	Tag is of type Byte
CROTiffTagDouble	Tag is of type Double

CROTiffTagFloat	Tag is of type Float
CROTiffTagLong	Tag is of type Long
CROTiffTagRational	Tag is of type Rational
CROTiffTagSByte	Tag is of type Signed Byte
CROTiffTagShort	Tag is of type Short
CROTiffTagSLong	Tag is of type Signed Long
CROTiffTagSRational	Tag is of type Signed Rational
CROTiffTagSShort	Tag is of type Signed Short
CROTiffTagUndefined	Tag type is Undefined
CROTiffTagUnicode	Tag is of type Unicode

CROViewAlignmentX

This type definition specifies the alignment in x-direction.

Type	Description
CROcAlignCenterX	Centered in x-direction
CROcAlignLeft	Left aligned in x-direction
CROcAlignRight	Right aligned in x-direction

CROViewAlignmentY

This type definition specifies the alignment in y-direction.

Type	Description
CROcAlignBottom	Bottom aligned in y-direction
CROcAlignCenterY	Centered in y-direction
CROcAlignTop	Top aligned in y-direction

CROViewMode

This type definition specifies the viewing mode of the image.

Type	Description
CROcBestFit	Resizes the image to fit to page.
CROcCustomZoom	Resizes the image depending upon the specified zoom percentage.
CROcFitToHeight	Resizes the image to fit to the height of the viewer.
CROcFitToWidth	Resizes the image to fit to the width of the viewer

CroViewerSelectionFunctionality

This type definition specifies if the selection area can be move or resized.

Type	Description
CroViewerSelectionFunctionalityMove	Move selection area
CroViewerSelectionFunctionalityResize	Resize selection area

CroViewToDoc

This type definition specifies how the viewer coordinates are transfered into the document coordinates.

Type	Description
CroDocPage	Document page
CroDocPixX	Document coordinate x in pixels
CroDocPixY	Document coordinate y in pixels

Image Object SCBCroImage

The component SCBCroImage provides methods and properties to load, modify, and save image files.

SCBCroImage Methods

This object provides the following methods.

AppendToMultiImageFile

This method appends the current image to the specified file. Use this method to create and save multi-Tiff or other multi-image files.

Note: The specified file is created if not yet available.

Syntax

```
AppendToMultiImageFile (Filename As String, FileType As CROImgFileTypes,
Mode As CROImgIOMode)
```

Parameter	Description
Filename	Name of the image file
FileType	The default value is CROcTIF_G4. **Possible values** See CROImgFileTypes **Note:** The file type must support storing of multiple images within one file.
Mode	The default value is CROImgIOSync Specifies the I/O mode for saving the image. **Possible values** See CROImgIOMode **Note:** Use the SaveCompleted method to wait for the load process to complete. As long as the save process is running in the background, the image object is locked against any modification.

ChangePixelDepth

This method changes the number of bits per pixel of the currently loaded image.

The current implementation only supports increasing the pixel depth. You can change a black-and-white image to a gray or a 24-bit color image, but you cannot convert a color image to a 8-bit gray or a black-and-white image.

Syntax

```
ChangePixelDepth (BitsPerPixel As Long)
```

Parameter	Description
BitsPerPixel	Specifies the new value for the bits per pixel value of the image. **Possible values** 4, 8, and 24

Clear

This method frees all resources of the currently allocated images and deinitializes the object.

Syntax

```
Clear ()
```

Copy

This method creates a copy of the currently allocated image within the `TargetImage` object instance and frees all previously allocated resources of `TargetImage`.

If rectangle information are provided, only the specified rectangle is copied.

Syntax

```
Copy (TargetImage As ISCBCroImage, SrcXOffset As Long, SrcYOffset As Long,
Width As Long, Height As Long)
```

Parameter	Description
TargetImage	Target image to receive the copy of the image. The object can either be deinitialized or contain any image data. All previously allocated resources of `TargetImage` are freed.
SrcXOffset	Left X coordinate of the rectangle.
SrcYOffset	Top Y coordinate of the rectangle.
Width	Width of the rectangle to copy. Use 0 to copy the entire image.
Height	Height of the rectangle to copy. Use 0 to copy the entire image.

CopyToClipboard

This method copies the current SCBCroImage object to the clipboard.

Syntax

```
CopyToClipboard ()
```

Create

This method allocates memory for a new image with the specified size and properties.

Syntax

```
Create (Width As Long, Height As Long, ImageType As CROImgType,
XResolution As Long, YResolution As Long)
```

Parameter	Description
Width	Width of the newly created image
Height	Height of the newly created image

ImageType	Specifies colors and pixel per colors **Possible values** See CROImgType
XResolution	Specifies the horizontal resolution of the newly created image in dots per inch.
YResolution	Specifies the vertical resolution of the newly created image in dots per inch.

DetectSkew

This method calculates the skew of the current image.

You can use this value as a parameter for the `Rotate` method to deskew the image.

Syntax

```
DetectSkew () As Double
```

FillRectangle

This method fills the rectangle with the specified color.

Syntax

```
FillRectangle (Color As OLE_COLOR, left As Long, top As Long, width As
Long, height As Long)
```

Parameter	Description
Color	Color
left	Left coordinate of the rectangle
top	Top coordinate of the rectangle
width	Width of the rectangle
height	Height of the rectangle

Insert

This method copies the specified part of the image to `TargetImage`. TargetImage is not resized and all image information outside the specified rectangle is not changed.

Use this method to merge two images.

The `TargetImage` object must have the appropriate size to receive the image data and must have the same image type like the source image.

Syntax

```
Insert (TargetImage As ISCBCroImage, TargetXOffset As Long, TargetYOffset
As Long, SrcXOffset As Long, SrcYOffset As Long, Width As Long, Height As
Long)
```

Parameter	Description
TargetImage	Interface pointer to target image, which receives the copy of the image. No additional memory is allocated for this image. Use the `Create` method to allocate memory for the image.
TargetXOffset	Integer value specifying the `Left X` coordinate of the rectangle in the target image.
TargetYOffset	Integer value specifying the `Top Y` coordinate of the rectangle in the target image.
SrcXOffset	Integer value specifying the `Left X` coordinate of the rectangle within the source image.
SrcYOffset	Integer value specifying the `Top Y` coordinate of the rectangle within the source image.
Width	Width of the rectangle to copy. Use 0 to copy the entire image.
Height	Height of the rectangle to copy. Use 0 to copy the entire image.

LoadCompleted

This method checks if a pending load operation has already completed or waits until a pending load operation has completed.

Syntax

```
LoadCompleted (Wait As Boolean) As Boolean
```

Parameter	Description
Wait	TRUE: Wait until the pending load operation has completed. FALSE Return immediately and report only the current status.

LoadFile

This method loads an image in synchronous mode.

If the specified file is a multi-image file, the method loads the first image of the file.

The method triggers the `ContentChanged` event, which causes a `SCBCroViewer` control to repaint the image.

Note: The asynchronous loading of files with `Mode = CROImgIOAsync` is currently not implemented.

Syntax

```
LoadFile (Filename As String, Mode As CROImgIOMode)
```

Parameter	Description
Filename	Specifies the filename to load as image.
Mode	Specifies the I/O mode for loading the file. With `CROImgIOSync`, the method waits until loading of the image has finished.
	Use the `LoadCompleted` method to wait for the load process to complete. As long as the load process runs in the background, the image object is locked against any read access or modification. The default value is `CROImgIOSync`.

LoadMultiImageFile

This method loads an image from a multi-image file.

The parameter `StartIndex` specifies the index of the image in the file.

The method triggers the `ContentChanged` event, which causes the `SCBCroViewer` control to repaint the image.

Note: The asynchronous loading of files with Mode = CROImgIOAsync is currently not implemented.

Syntax

```
LoadMultiImageFile (Filename As String, StartIndex As Long, Mode As
CROImgIOMode)
```

Parameter	Description
Filename	Specifies the file name to load as image.
StartIndex	Specifies the index of the image within the multiimage file which should be loaded.
Mode	Specifies the I/O mode for loading the file. With `CROImgIOSync`, the method waits until loading of the image has finished. With `CROImgIOAsync`, the method returns immediately and the image loads in the background. Use the `LoadCompleted` method to wait for the load process to complete. As long as the load process runs in the background, the image object is locked against any read access or modification. The default value is `CROImgIOSync`.

Move

This method changes the value of the x and y offset of the image. The internal image format is not changed.

Syntax

```
Move (dx As Long, dy As Long)
```

Parameter	Description
dx	This offset is added to the current XOffset.
dy	This offset is added to the current YOffset.

Rotate

This method rotates the image with the specified angle.

The method triggers the `ContentChanged` event, which causes the `SCBCroViewer` control to repaint the image.

Syntax

```
Rotate (angle As Double, resize As Boolean)
```

Parameter	Description
angle	Specifies the angle to rotate the image.
resize	If set to VARIANT_TRUE the image will be enlarged so that no pixel of the original image will be lost, else the image will keep the same size but some pixel information near the corners will be lost as a result of the rotation. The default value is -1.

SaveCompleted

This method checks if a pending save operation has already completed or waits until a pending save operation has completed.

Note: The asynchronous saving of files with Mode = CROImgIOAsync is currently not implemented.

Syntax

```
SaveCompleted (Wait As Boolean) As Boolean
```

Parameter	Description
Wait	TRUE: Wait until the pending save operation has completed. FALSE: Return immediately and report only the current status.

SaveFile

This method saves an image with the specified name and file type.

If a file with the specified name already exists, the existing file is replaced. To create and save multi-image files, use the method AppendToMultiImageFile.

The asynchronous saving of files with Mode = CROImgIOAsync is currently not implemented.

Syntax

```
SaveFile (Filename As String, FileType As CROImgFileTypes, Mode As
CROImgIOMode)
```

Parameter	Description
Filename	Specifies the file name for saving the image.
FileType	Specify the image file format. **Possible values** See CROImgFileTypes The default value is CROcTIF_G4.
Mode	Specifies the I/O mode for loading the file. With `CROImgIOSync`, the method waits until loading of the image has finished. With `CROImgIOAsync`, the method returns immediately and the image loads in the background. Use the `SaveCompleted` method to wait for the load process to complete. As long as the save process runs in the background, the image object is locked against any read access or modification. The default value is `CROImgIOSync`.

ScaleImage

This method resizes an existing image.

The scale factors define if the image becomes smaller or bigger.

You can specify different scale factors for the x and y direction.

A factor of 1 does not change the image size,

A factor of 2 doubles the number of pixels for the specified direction.

A factor of 0.5 halves the number of pixels.

Syntax

```
ScaleImage (ScaleX As Double, ScaleY As Double)
```

Parameter	Description
ScaleX	Scale factor for the image width.
ScaleY	Scale factor for the image height.

TiffAppendFile

This method appends the specified TIFF file to the one to which the image object currently references.

Syntax

```
TiffAppendFile (AppendPage As String)
```

Parameter	Description
AppendPage	The TIFF file to append.

TiffTagAddASCII

This method adds a new TIFF tag to the image object.

Note: The method does not save the changes to the image file. To save the file, call a save method.

Syntax

```
TiffTagAddASCII (TagId As Long, Value As String)
```

Parameter	Description
TagId	The Id of the new tag must be in the range of valid user tags for TIFF files. **Possible values** 32768 to 65535 Additionally, the printer tag is allowed, which is defined as 337.
Value	String value of the new TIFF tag.

TiffTagAddDouble

This method adds a new TIFF tag of type double to the image object.
Note: The method does not save the changes to the image file. To save the file, call a save method.

Syntax

```
TiffTagAddDouble (TagId As Long, TagType As CROTiffTagType, Value As
Double)
```

Parameter	Description
TagId	The Id of the new tag must be in the range of valid user tags for TIFF files. **Possible values** 32768 to 65535 Additionally, the printer tag is allowed, which is defined as 337.
TagType	Type of the new tag. **Possible values** - CROTiffTagDouble - CROTiffTagFloat
Value	Double value of the new TIFF tag.

TiffTagAddLong

This method adds a new TIFF tag of type long to the image object.
Note: The method does not save the changes to the image file. To save the file, call a save method.

Syntax

```
TiffTagAddLong (TagId As Long, TagType As CROTiffTagType, Value As Long)
```

Parameter	Description
TagId	The Id of the new tag must be in the range of valid user tags for TIFF files. **Possible values** 32768 to 65535 Additionally, the printer tag is allowed, which is defined as 337.
TagType	Type of the new tag. Type of the new tag, must be one of CROTiffTagByte, CROTiffTagShort, CROTiffTagLong, CROTiffTagSByte, CROTiffTagSShort or CROTiffTagSLong. **Possible values** - CROTiffTagByte - CROTiffTagShort - CROTiffTagLong - CROTiffTagSByte - CROTiffTagSShort - CROTiffTagSLong
Value	Long value of the new TIFF tag.

TiffTagAddRational

This method adds a new TIFF tag of type rational to the image object.

Note: The method does not save the changes to the image file. To save the file, call a save method.

Syntax

```
TiffTagAddRational (TagId As Long, TagType As CROTiffTagType, Num As Long,
Denom As Long)
```

Parameter	Description
TagId	The Id of the new tag must be in the range of valid user tags for TIFF files. **Possible values** 32768 to 65535 Additionally, the printer tag is allowed, which is defined as 337.
TagType	Type of the new tag. **Possible values** - CROTiffTagDouble - CROTiffTagFloat
Num	Numerator value of the new rational TIFF tag.
Denom	Denominator value of the new rational TIFF tag. Must not be 0.

TiffTagAddVariant

This method adds a new TIFF tag of type variant to the image object.

Note: The method does not save the changes to the image file. To save the file, call a save method.

Syntax

```
TiffTagAddVariant (TagId As Long, TagType As CROTiffTagType, Value As
Variant)
```

Parameter	Description
TagId	The Id of the new tag must be in the range of valid user tags for TIFF files. **Possible values** 32768 to 65535 Additionally, the printer tag is allowed, which is defined as 337.

TagType	Type of the new tag. **Possible values** - CROTiffTagASCII - CROTiffTagByte - CROTiffTagDouble - CROTiffTagFloat - CROTiffTagLong - CROTiffTagShort - CROTiffTagSByte - CROTiffTagSLong - CROTiffTagSShort
Value	Value of the new TIFF tag. The type of this value must correspond to the specified tag type.

TiffTagClearAll

This method removes all TIFF tags defined for the currently loaded image.

Note: The method does not save the changes to the image file. To save the file, call a save method.

Syntax

```
TiffTagClearAll ()
```

TiffTagDelete

This method removes the TIFF tag from the tag list specified by the index of the currently loaded image.

Note: The method does not save the changes to the image file. To save the file, call a save method.

Syntax

```
TiffTagDelete (index As Long)
```

Parameter	Description
index	Index value **Possible values** 0 to TIFFTagCount-1

SCBCroImage Properties

This object provides the following properties.

BitsPerPixel

This read-only property returns the number of bits per pixel of the current loaded image.

Syntax

```
BitsPerPixel As Long
```

DefaultJPGQuality

This property sets or returns the default quality for saving JPG-images.

The value is shared by all image objects running within the same application and is used as long as the `JPGQuality` parameter was not used.

Possible values

1 - 100, where 1 means a high compression and low image quality and a value of 100 means a low compression and a high image quality.

Syntax

```
DefaultJPGQuality As Long
```

FileName

This property returns the path used by the last `LoadFile` or `LoadMultiImageFile` operation or an empty string when no load operation was executed before.

Setting the property causes the image object to load the specified file. This is equivalent to the usage of `LoadFile`.

Syntax

```
FileName As String
```

Height

This read-only property returns the height of the image in pixels.

Syntax

```
Height As Long
```

ImageCount

This read-only property returns the number of images in the file where the image was loaded from, such as the number of images in a multi-image file like a Multi-TIFF.

Note: Use `LoadMultiImageFile` to load an image from a specified index of the file.

Syntax

```
ImageCount As Long
```

ImageIndex

This read-only property returns the index of the currently loaded image within the multi-image file.

Note: This value corresponds to the StartIndex parameter used in the last `LoadMultiImageFile` operation.

Syntax

```
ImageIndex As Long
```

ImageType

This read-only property returns the type of the currently loaded image.

Syntax

```
ImageType As CROImgType
```

JPGQuality

This property specifies the default value for saving JPG images. This value is shared by all image objects running within the same application and is used as long as the JPGQuality parameter was not used.

Possible value

1 - 100, where 1 means a high compression and low image quality and 100 means a low compression and a high image quality.

Syntax

```
JPGQuality As Long
```

LDFEnabled

This read-only property returns if the Lura document format is available.

Syntax

```
LDFEnabled As Boolean
```

LDFLayer

This property sets or returns the current Lura document layer for the image object.

Syntax

```
LDFLayer As CROLDFLayer
```

LDFSettings

This property sets or returns the settings object for the Lura document object.

Syntax

```
LDFSettings As ISCBCroLDFSettings
```

Modified

This property sets or returns the modified state of the image, which is changed by methods like `Rotate` or `ChangePixelDepth`.

Syntax

```
Modified As Boolean
```

TIFFForceSngStrip

This property controls the saving of TIFF files.

By default, the data section in TIFF files is stored with several strips.

As some older applications do not support multi-striped TIFF files, you can force the image object to save a TIFF file using a single strip.

Syntax

```
TIFFForceSngStrip As Boolean
```

TiffTagCount

This read-only property returns the number of user defined TIFF tags in the current image.

Syntax

```
TiffTagCount As Long
```

TiffTagDouble

This read-only property returns the TIFF tag value of type double of the TIFF tag list specified by the index.

Note: The specified TIFF tag must have a TIFF tag id of `CROTiffTagFloat` or `CROTiffTagDouble`.

Syntax

```
TiffTagDouble (index As Long) As Double
```

Parameter	Description
index	Zero-based index **Possible values** 0 to TIFFTagCount-1

TiffTagID

This read-only property returns the TIFF tag id of the TIFF tag specified by index.

Syntax

```
TiffTagID (index As Long) As Long
```

Parameter	Description
index	Zero-based index **Possible values** 0 to TIFFTagCount-1

TiffTagLong

This read-only property returns the TIFF tag value of the TIFF tag specified by the index.

Note: The specified TIFF tag must have a TIFF tag id of `CROTiffTagByte`, `CROTiffTagShort`, `CROTiffTagLong`, `CROTiffTagSByte`, `CROTiffTagSShort`, or `CROTiffTagSLong`.

Syntax

```
TiffTagLong (index As Long) As Long
```

Parameter	Description
index	Zero-based index **Possible values** 0 to TIFFTagCount-1

TiffTagRatDenom

This read-only property returns the denominator of a rational TIFF tag value of the tag specified by the index.

Note: The specified TIFF tag must have a TIFF tag id of `CROTiffTagRational` or `CROTiffTagSRational`.

Syntax

```
TiffTagDouble (index As Long) As Double
```

Parameter	Description
index	Zero-based index **Possible values** 0 to TIFFTagCount-1

TiffTagRatNum

This read-only property returns the numerator of a rational TIFF tag value of the tag specified by index.

Note: The specified TIFF tag must have a TIFF tag id of `CROTiffTagRational` or `CROTiffTagSRational`.

Syntax

```
TiffTagRatNum (index As Long) As Long
```

Parameter	Description
index	Zero-based index **Possible values** 0 to TIFFTagCount-1

TiffTagString

This read-only property returns the TIFF tag string value of the TIFF tag specified by index.

Note: The specified TIFF tag must have a TIFF tag id of `CROTiffTagByte` or `CROTiffTagShort`, `CROTiffTagLong`, `CROTiffTagSByte`, `CROTiffTagSShort`, or `CROTiffTagSLong`.

Syntax

```
TiffTagString (index As Long) As String
```

Parameter	Description
index	Zero-based index **Possible values** 0 to TIFFTagCount-1

TiffTagType

This read-only property returns the data type of the specified tag in the Tiff file.

Syntax

```
TiffTagType (index As Long) As CROTiffTagType
```

Parameter	Description
index	Zero-based Index of Tiff tag

TiffTagVariant

This read-only property returns the TIFF tag value of the TIFF tag specified by index.

Note: The specified TIFF tag must have a TIFF tag id of `CROTiffTagByte`, `CROTiffTagASCII`, `CROTiffTagShort`, `CROTiffTagLong`, `CROTiffTagSByte`, `CROTiffTagSShort`, `CROTiffTagSLong`, `CROTiffTagFloat` or `CROTiffTagDouble`.

Syntax

```
TiffTagVariant (index As Long) As Variant
```

Parameter	Description
index	Zero-based index **Possible values** 0 to TIFFTagCount-1

Width

This read-only property returns the width of the image in pixels.

Syntax

```
Width As Long
```

XOffset

This read-only property returns the X coordinate of the left top in pixel. By default, the value is 0. To change the value, use the `Move` method.

Syntax

```
XOffset As Long
```

XResolution

This read-only property returns the horizontal resolution of the image in dots per inch.

Syntax

```
XResolution As Long
```

YOffset

This read-only property returns the y coordinate of left top in pixels. By default, the value is 0. To change the value, use the `Move` method.

Syntax

```
YOffset As Long
```

YResolution

This read-only property returns the vertical resolution of the image in dots per inch.

Syntax

```
YResolution As Long
```

Sample Code

The following sample code shows the usage of some SCBCroImage features, such as `LoadFile`, `DetectSkew`, `Rotate`, and `SaveFile`. The results are displayed with SCBCroViewer.

```
Dim img As SCBCroImage ' Button DetectSkew Private Sub btnDeskew_Click()
Dim angle As Double angle = img.DetectSkew img.Rotate angle End Sub '
Button LoadFile Private Sub btnLoadImage_Click() Set img = New SCBCroImage
 img.LoadFile "d:\images\aeg\00000003.tif" Set SCBCroViewer1.Item = img
End Sub ' Button Rotate image Private Sub btnRotateRight_Click()
img.Rotate 90 End Sub ' Button SaveFile Private Sub btnSave_Click()
img.SaveFile "d:\images\aeg\TestSave.tif", CROcTIF_G4 End Sub
```

Viewer Object SCBCroViewer

The component SCBCroViewer provides methods and properties to display various objects, such as images, text documents, annotations, or OCR and scanning results.

In general, there are two different ways to display an object.

- Viewing documents
- Viewing Cairo projects and annotations

SCBCroViewer Methods

This object provides the following methods.

ConvertDocToView

This method converts document coordinates into viewer coordinates.

Syntax

```
ConvertDocToView (mode As CroDocToView, x As Long, y As Long) As Long
```

Parameter	Description
mode	**Possible values** See CroDocToView
X	X-value of the document coordinate
Y	Y-value of the document coordinate

ConvertViewToDoc

This method converts viewer coordinates into document coordinates.

Syntax

```
ConvertViewToDoc (mode As CroViewToDoc, x As Long, y As Long) As Long
```

Parameter	Description
mode	**Possible values** See CroViewToDoc
X	X-value of the viewer coordinate
Y	Y-value of the viewer coordinate

FitToRectangle

This method displays the selected area in the viewer.

Syntax

```
FitToRectangle (Left As Long, Top As Long, Width As Long, Height As Long)
```

Parameter	Description
Left	Left coordinate of the selected rectangle
Top	Top coordinate of the selected rectangle
Width	Width of the selected rectangle
Height	Height of the selected rectangle

MakeRectangleVisible

This method makes a rectangle with a defined distance to the border visible.

Syntax

```
MakeRectangleVisible (Left As Long, Top As Long, Width As Long, Height As
Long, DistanceToBorder As Long)
```

Parameter	Description
Left	Left coordinate of the rectangle
Top	Top coordinate of the rectangle
Width	Width of the rectangle
Height	Height of the rectangle
DistanceToBorder	Distance to border

Refresh

This method refreshes the contents of the viewer.

Syntax

```
Refresh ()
```

RemoveSelection

This method removes the selection object from the viewer.

Syntax

```
RemoveSelection ()
```

ScrollTo

This method scrolls a displayed object to a specified offset.

The result is identical to using the `OffsetX` and `OffsetY` properties, but performs this operation in one step, avoiding unnecessary display movement.

Syntax

```
ScrollTo (XOffset As Long, YOffset As Long)
```

Parameter	Description
XOffset	Offset in x-direction in pixels
YOffset	Offset in y-direction in pixels

SetSelection

This method creates a selection object and inserts it into the viewer.

Syntax

```
SetSelection (Left As Long, Top As Long, Width As Long, Height As Long)
```

Parameter	Description
Left	Left coordinate of the selection rectangle
Top	Top coordinate of the selection rectangle
Width	Width of the selection rectangle
Height	Height of the selection rectangle

StartManualSelection

This method creates a selection object if the left mouse button is pressed and then released.

Syntax

```
StartManualSelection ()
```

StartManualSelection2

This method provides the same functionality as `StartManualSelection`, but does not immediately remove the existing selection.

Syntax

```
StartManualSelection2 ()
```

SCBCroViewer Properties

This object provides the following properties.

AlignmentX

This property sets or returns the alignment the displayed object, if its displayed width is smaller than the viewer window’s width.

Syntax

```
AlignmentX As CROViewAlignmentX
```

Possible values

See CROViewAlignmentX

AlignmentY

This property sets or returns the alignment the displayed object, if its displayed height is smaller than the viewer window’s height.

Syntax

```
AlignmentY As CROViewAlignmentY
```

Possible values

See CROViewAlignmentY

BackColor

This property sets or returns the back color of the viewer.

Syntax

```
BackColor As OLE_COLOR
```

BorderColor

This property sets or returns the color of the border.

Syntax

```
BorderColor As OLE_COLOR
```

BorderStyle

This property sets or returns the style of the border.

Syntax

```
BorderStyle As Long
```

BorderWidth

This property sets or returns the width of the border

Syntax

```
BorderWidth As Long
```

Brightness

This read-only property returns the brightness property that controls the display of colored images.

Note: This property has no effect on black-and-white images.

Syntax

```
Brightness As Double
```

Possible values

–255.0 to 255.0

Default value: 0.0

Darkest: -255.0

Brightest: 255.0

Contrast

This property controls the display of colored images. The valid range for the contrast property is from 0.0 to 100.0, where 1.0 is the default value, 0.0 is the lowest contrast and 100.0 is the highest contrast.

Note: This property has no effect on black-and-white images.

```
Contrast As Double
```

Possible values

0.0 to 100.0

Default value: 1.0

Lowest contrast: 0.0

Highest contrast: 100.0

DisplayPage

This property sets or returns the displayed page of a document.

Syntax

```
DisplayPage As Long
```

EnableTool

This property controls the activation of the user interactions tool of the displayed item.

Syntax

```
EnableTool As Boolean
```

Possible values

True: Default value. Enables the displayed object to process mouse and keyboard events.

False: The tool object is disabled and no mouse and keyboard events are processed by the displayed item.

FreeScrollPosX

This property sets or returns the scrolling behavior of the viewer for the horizontal direction.

Syntax

```
FreeScrollPosX As Boolean
```

Possible values

True: The user can scroll the document without restrictions.

False: Default value. Limits the scrolling offset to the size of the document.

FreeScrollPosY

This property sets or returns the scrolling behavior of the viewer for the vertical direction.

Syntax

```
FreeScrollPosY As Boolean
```

Possible values

True: The user can scroll the document without restrictions.

False: Default value. Limits the scrolling offset to the size of the document.

Item

This property sets or returns a reference of the object to be displayed in the viewer.

Syntax

```
Item As Object
```

The object must be a displayable object for the Cairo viewer.

The following objects are displayable

- The Cairo image object SCBCroImage
- The Cairo project SCBCroProject
- The Cairo annotation object SCBCroAnnotation
- The Cedar workdoc object

OffsetX

This read-only property controls the horizontal scrolling offset of the displayed object.

Note: An offset of 0 means that the left margin of the document meets the left margin of the viewer.

Syntax

```
OffsetX As Long
```

OffsetY

This read-only property controls the vertical scrolling offset of the displayed object.

Note: An offset of 0 means that the upper margin of the document meets the upper margin of the viewer.

Syntax

```
OffsetY As Long
```

PageCount

This property sets or returns the page count of the currently displayed document.

Syntax

```
PageCount As Long
```

ScaleToGray

This property sets or returns how black-and-white images are displayed.

Set the property to TRUE to enable the ScaleToGray function, which results in a better display of images zoomed to a smaller size.

Syntax

```
ScaleToGray As Boolean
```

ScrollbarVisible

This property sets or returns whether the scroll bars are visible.

Syntax

```
ScrollbarVisible As Boolean
```

Selection

This property sets or returns the currently selected viewer object.

Syntax

```
Selection As ISCBCroViewSelection
```

SelectionBackgroundColor

This property sets or returns the background color of the selection.

Syntax

```
SelectionBackgroundColor As COLOR
```

SelectionBorderColor

This property sets or returns the border color of the selection.

Syntax

```
SelectionBackgroundColor As COLOR
```

SelectionEnable

This property sets or returns which selection mode is enabled.

Syntax

```
SelectionEnable (functionality As CroViewerSelectionFunctionality) As
Boolean
```

Possible values

See CroViewerSelectionFunctionality

ShowPageSelectButtons

This property sets or returns if the page selection buttons on the toolbar are visible.

Syntax

```
ShowPageSelectButtons As Boolean
```

ShowSelectionButton

This property sets or returns if the selection buttons are visible.

Syntax

```
ShowSelectionButton As Boolean
```

ShowToolToolbar

This property sets or returns whether the toolbar provided by the user interaction tool is visible.

Syntax

```
ShowToolToolbar As Boolean
```

ShowViewerToolbar

This property sets or returns whether the toolbar of the viewer is visible.

Syntax

```
ShowViewerToolbar As Boolean
```

Tool

This read-only property returns the tool object provided by the displayed object.

Syntax

```
Tool As Object
```

ViewMode

This property controls the default zoom mode.

Syntax

```
ViewMode As CROViewMode
```

Possible values

See CROViewMode

Zoom

This property sets or returns the zoom value of the displayed document in percent.

Syntax

```
Zoom As Double
```

ZoomStep

This property defines the zoom factor applied when the user presses the **Zoom in** or **Zoom out** button.

Syntax

```
ZoomStep As Double
```

SCBCroViewer Events

This object provides the following events.

LDFProcessActive

This event triggers when the Lura document process is active.

Syntax

```
LDFProcessActive ()
```

OnKeyDown

This event triggers when a key is pressed.

Syntax

```
OnKeyDown (lChar As Long, lFlags As Long, bHandled As Boolean)
```

Parameter	Description
lChar	ASCII sign of the pressed key.
lFlags	Contains the number an additionally pressed key.
bHandled	True when the event was processed

OnKeyUp

This event triggers when the users releases the key.

Syntax

```
OnKeyUp (lChar As Long, lFlags As Long, bHandled As Boolean)
```

Parameter	Description
lChar	ASCII sign of the released key.
lFlags	Contains the number an additionally released key.
bHandled	True when the event was processed

OnLButtonDblClk

This event triggers when the user double-clicks with the left mouse button in the viewer.

Syntax

```
OnLButtonDblClk (lFlags As Long, x As Long, y As Long, bHandled As
Boolean)
```

Parameter	Description
lFlags	Contains the number an additionally pressed key-
x	X coordinate
y	Y coordinate
bHandled	True when the event was processed.

OnLButtonDown

This event triggers when the user presses the left mouse button.

Syntax

```
OnLButtonDown (lFlags As Long, x As Long, y As Long, bHandled As Boolean)
```

Parameter	Description
lFlags	Contains the number an additionally pressed key.
x	X coordinate in pixels
y	Y coordinate in pixels
bHandled	True when the event was processed.

OnLButtonUp

This event triggers when the user releases the left mouse button.

Syntax

```
OnLButtonUp (lFlags As Long, x As Long, y As Long, bHandled As Boolean)
```

Parameter	Description
lFlags	Contains the number an additionally pressed key.
x	X coordinate in pixels
y	Y coordinate in pixels
bHandled	True when the event was processed.

OnMouseMove

This event triggers when the user moves the mouse.

Syntax

```
OnMouseMove (lFlags As Long, x As Long, y As Long, bHandled As Boolean)
```

Parameter	Description
lFlags	Contains the number an additionally pressed key.
x	X coordinate in pixels
y	Y coordinate in pixels

bHandled	True when the event was processed.

OnRButtonDblClk

This event triggers when the user double-clicks with the right mouse button in the viewer.

Syntax

```
OnRButtonDblClk (lFlags As Long, x As Long, y As Long, bHandled As
Boolean)
```

Parameter	Description
lFlags	Contains the number an additionally pressed key-
x	X coordinate
y	Y coordinate
bHandled	True when the event was processed.

OnRButtonDown

This event triggers when the user presses the right mouse button.

Syntax

```
OnRButtonDown (lFlags As Long, x As Long, y As Long, bHandled As Boolean)
```

Parameter	Description
lFlags	Contains the number an additionally pressed key.
x	X coordinate in pixels
y	Y coordinate in pixels
bHandled	True when the event was processed.

OnRButtonUp

This event triggers when the user releases the right mouse button.

Syntax

```
OnRButtonUp (lFlags As Long, x As Long, y As Long, bHandled As Boolean)
```

Parameter	Description
lFlags	Contains the number an additionally pressed key.
x	X coordinate in pixels
y	Y coordinate in pixels
bHandled	True when the event was processed.

OnSelection

This event triggers when a selection object is created or changed.

Syntax

```
OnSelection (mode As CroViewerSelectionMode, left As Long, top As Long,
width As Long, height As Long, bHandled As Boolean)
```

Parameter	Description
mode	Current state of selection event **Possible values** See CroViewerSelectionMode
left	Left coordinate in pixels
top	Top coordinate in pixels
width	Width of the selection rectangle
height	Height of the selection rectangle
bHandled	True when the event was processed.

PageChanged

This event triggers when the page is changed.

Syntax

```
PageChanged (NewPageNr As Long)
```

Parameter	Description
NewPageNr	New Page number

SetDisplayPage

This event triggers when the container requests to change the displayed page of a document.

Syntax

```
SetDisplayPage (PageNr As Long)
```

Parameter	Description
PageNr	Number of the page to be displayed.

ViewmodeChanged

This event triggers when the ViewMode of the Viewer changes.

Syntax

```
ViewmodeChanged (NewViewMode As CROViewMode)
```

Parameter	Description
NewViewMode	New ViewMode **Possible values** See CROViewMode

ZoomChanged

This event triggers when the zoom changes.

Syntax

```
ZoomChanged (NewZoom As Double)
```

Parameter	Description
NewZoom	New zoom value

Sample Code

The following sample code shows how to load and display an image file in a Cairo Viewer object.

```
Private Sub btnLoadImage_Click() Dim Img As SCBCroImage Set Img = New
SCBCroImage Img.LoadFile "C:\images\sample.tif" ' View the imagefile
sample.tif Set SCBCroViewer1.Item = Img End Sub
```

Working with CI Documents (SCBCroCiDocLib)

The SCBCroCiDocLib provides the following objects to work with coded information (CI) documents.

- SCBCroCIDOC
- SCBCroCIDOCConv

SCBCroCiDocLib Type Definitions

The SCBCroCiDocLib object provides the following type definition.

CROViewDim

This type definition specifies whether the dimensions are measured in metric units or pixels.

Type	Description
CROcDimMetric	Dimensions measured in metric units
CROcDimPixel	Dimensions measured in pixels units

CI Document Object SCBCroCIDOC

The component SCBCroCIDOC provides methods and properties to work with CI documents.

SCBCroCIDOC Methods

This object provides the following methods.

ClearAllHighlights

This method is reserved for future use and should not be used in the current version.

Syntax

```
ClearAllHighlights ()
```

CloseDocument

This method closes the current file in the viewer.

Syntax

```
CloseDocument ()
```

CreateCroImage

This method creates a CroImage object for the specified page number at a specified resolution.

Syntax

```
CreateCroImage (PageNumber As Long, Resolution As Long) As ISCBCroImage
```

Parameter	Description
PageNumber	Number of page to be drawn into CroImage **Possible values** 1 - PageCount
Resolution	Resolution of the CroImage object in dots per inch.

GetDimensions

This method returns the width and height of the currently selected document.

Syntax

```
GetDimensions (pulWidth As Long, pulHeight As Long, DimUnit As CROViewDim)
```

Parameter	Description
pulWidth	Width (output parameter)
pulHeight	Height (output parameter)
DimUnit	Specifies whether the dimensions are returned in metric units or pixels. **Possible values** CROViewDim

GetFileInfo

This method is reserved for future use and should not be used in the current version.

Syntax

```
GetFileInfo (pFileID As Long) As String
```

GetINSODisplayEngine

This method is reserved for future use and should not be used in the current version.

Syntax

```
GetINSODisplayEngine (pDispEngineID As Long) As String
```

Highlight

This method is reserved for future use and should not be used in the current version.

Syntax

```
Highlight (StartPos As Long, EndPos As Long, FColor As OLE_COLOR, BColor
As OLE_COLOR)
```

LoadFile

Loads a CI document file. If another file is already open, it is automatically closed.

Syntax

```
LoadFile (FileName As String)
```

Parameter	Description
FileName	Path and file name of the CI document to be loaded.

PrintDocument

This method is reserved for future use and should not be used in the current version.

Syntax

```
PrintDocument (ShowDialog As Boolean, StartPage As Long, EndPage As Long)
```

ReleaseDocument

This method is reserved for future use and should not be used in the current version.

Syntax

```
ReleaseDocument ()
```

SetFirstPage

This method sets the displayed page number to the first page.

Syntax

```
SetFirstPage ()
```

SetLastPage

This method sets the displayed page number to the last page.

Syntax

```
SetLastPage ()
```

SetNextPage

This method sets the displayed page number to the next page.
Note: If the page number is already set to the last page, the method does not perform an action.

Syntax

```
SetNextPage ()
```

SetPreviousPage

This method sets the displayed page number to the previous page.

Note: If the page number is already set to the first page, the method does not perform an action.

Syntax

```
SetPreviousPage ()
```

SCBCroCIDOC Properties

This object provides the following properties.

CharacterSet

This property is reserved for future use and should not be used in the current version.

Syntax

```
CharacterSet As CroCIDocCharSet
```

CharCount

This property is reserved for future use and should not be used in the current version.

Syntax

```
CharCount As Long
```

CurrentDocType

This property is reserved for future use and should not be used in the current version.

Syntax

```
CurrentDocType As CroCIDocType
```

DocTypeDetection

This property is reserved for future use and should not be used in the current version.

Syntax

```
DocTypeDetection As CroCIDocType
```

FileName

This read-only property returns the file name of the currently displayed CI document.

Syntax

```
FileName As String
```

PageCount

This read-only property returns total number of pages for the currently loaded CI document.

Syntax

```
PageCount As Long
```

PageNumber

This property sets or returns the number of the currently displayed page.

Syntax

```
PageNumber As Long
```

WordCount

This property is reserved for future use and should not be used in the current version.

Syntax

```
WordCount As Long
```

WordEndPos

This property is reserved for future use and should not be used in the current version.

Syntax

```
WordEndPos (WordIndex As Long) As Long
```

WordStartPos

This property is reserved for future use and should not be used in the current version.

Syntax

```
WordStartPos (WordIndex As Long) As Long
```

Sample Code

The following sample code shows how to set a document to the first page and how to open a CI document.

```
‘ Select the first page from the menu Private Sub mnFirstPage_Click()
CIDoc.SetFirstPage End Sub ‘ ‘ Select a file to load from the common
dialog Private Sub mnOpenCIDocument_Click() On Error GoTo lblErr With
CommonDialog1 .DialogTitle = "Load document file" .CancelError = True
.Filter = "All Files(*.*)|*.*" .ShowOpen End With CIDoc.LoadFile
CommonDialog1.FileName Set SCBCroViewer1.Item = CIDoc ' reset mouse
pointer DoEvents MousePointer = 0 Exit Sub lblErr: MsgBox Err.Description
End Sub
```

CI Document Conversion Object SCBCroCIDOCConv

The component SCBCroCIDOCConv provides methods to convert a CI document to a text file or a Worktext.

SCBCroCIDOCConv Methods

This object provides the following methods.

CreateText

This method returns the text of the CI document as string.

Syntax

```
CreateText (pCroCIDoc As ISCBCroCIDoc) As String
```

Parameter	Description
pCroCIDoc	Pointer to the CI document object.

CreateWorktext

This method creates a WorkText object from the CI document.

Syntax

```
CreateWorktext (pCroCIDoc As ISCBCroCIDoc, pWorktext As ISCBCroWorktext)
```

Parameter	Description
pCroCIDoc	Pointer to the CI document object.
pWorktext	Pointer to the WorkText object.

Working with Cairo Projects (SCBCroPROJLib)

The SCBCroPROJLib provides the following objects to work with Cairo projects.

- SCBCroProject
- SCBCroPage
- SCBCroPages
- SCBCroZone
- SCBCroZones

SCBCroPROJLib Type Definitions

The SCBCroProject object provides the following type definitions.

CROcPagePrepStep

This type definition specifies the pre-processing method groups.

Type	Description

CROcPagePrepStep_ PostLocate	All pre-processing methods are executed.
CROcPagePrepStep_ PreLocate	Only the pre-processing methods prior to locating the anchors, such as image enhancement, are executed.

CROcPageState

This type definition specifies the state of the page.

Type	Description
CROcPageStateAllZonesLocated	All zones are located
CROcPageStateAllZonesPreprocessed	All zones are preprocessed
CROcPageStateLocated	Page is located
CROcPageStatePreprocessedPostLocate	Page is preprocessed and located
CROcPageStatePreprocessedPreLocate	Page is preprocessed, but not yet located
CROcPageStateReset	Page is reset

CROcZoneState

This type definition specifies the state of a zone.

Type	Description
CROcZoneStateLocated	Zone is located
CROcZoneStatePreprocessed	Zone is preprocessed
CROcZoneStateReset	Zone reset

CroEngineType

This type definition specifies the type of the default engine.

Type	Description
CROcEngineBarcode	Barcode engine
CROcEngineOCR	OCR engine
CROcEngineOMR	OMR engine

CROcEngineUnknown	No default engine specified

CroPrepDirection

This type definition specifies the working direction for the pre-processing.

Type	Description
CroPrepDirection_Horizontal	Working direction is horizontal
CroPrepDirection_Vertical	Working direction is vertical

CROResetContext

This type definition specifies the type for which the reset is processed.

Type	Description
CROResetContextPage	All changes of the preproccessing are reset for the page
CROResetContextZone	All changes of the preproccessing are reset for the zone

CROZoneSelectState

This type definition specifies the selection state of a zone.

Type	Description
CROcSelectActive	Active selection
CROcSelectNonActive	Non-active selection
CROcSelectNone	No selection

Project Object SCBCroProject

The component SCBCroProject provides methods and properties to work with projects.

SCBCroProject Methods

This object provides the following methods.

Clear

This method deletes all entries in the page collection and triggers the OnClearProject event.

The method sets the Modified property to FALSE.

Syntax

```
Clear ()
```

GetVersionInfo

This method retrieves all available history and management information about the specified version.

The current status of the project remains unchanged.

Syntax

```
GetVersionInfo (Filename As String, Version As Long, Saved As Date,
Username As String, Computername As String, Description As String)
```

Parameter	Description
Filename	Path to a compound file containing the project definition, stored using the Save method.
Version	Project version from which the version information is loaded.
Saved	Returns the date on which the project was saved (output parameter)
Username	Returns the name of the user who saved the project (output parameter)
Computername	Returns the computer name on which the project was saved (output parameter)
Description	Returns the VersionInfo parameter specified for the Save method when the project was saved (output parameter)

Load

This method loads the project definition from the specified file and creates the corresponding project tree.

The method triggers the OnLoadProject event, so that possible project extensions can also load to load their settings from the project file.

If multiple versions are stored in the project file, the Load method always loads the latest version.

The method sets the Modified property to FALSE.

Syntax

```
Load (Filename As String, [IgnoreErrors As Boolean = FALSE])
```

Parameter	Description
Filename	Path to a compound file containing the project definition, stored using the Save method.

IgnoreErrors	Optional parameter **TRUE**: The method proceeds even if a no recognition engine or preprocessing method is registered on this computer. A missing recognition engine is replaced by the current default recognition engine, while a missing preprocessing method is removed from the definition. **FALSE**: The method fails if a recognition engine, preprocessing method or other component referenced by the project file is not registered on this computer. The default value is 0.

LoadVersion

This method loads the specified version from the file and creates the corresponding project tree.

The method triggers the OnLoadProject event so that possible project extensions can also load there settings from the project file.

The method sets the Modified property to FALSE.

Syntax

```
LoadVersion (Filename As String, Version As Long, IgnoreErrors As Boolean)
```

Parameter	Description
Filename	Path to a compound file containing the project definition, stored using the Save method.
Version	Project version to load from the file.
IgnoreErrors	**TRUE**: The Load method proceeds even if a no recognition engine or preprocessing method is registered on this computer. A missing recognition engine is replaced by the current default recognition engine, while a missing preprocessing method is removed from the definition. **FALSE**: The Load method fails if a recognition engine, preprocessing method or other component referenced by the project file is not registered on this computer.

RemoveVersion

This method removes the specified version from the version list in the project file.

The current state of the project tree remains unchanged.

Syntax

```
RemoveVersion (Filename As String, Version As Long)
```

Parameter	Description
Filename	Path to a compound file containing the project definition, stored using the Save method.

Version	Project version to remove from the file.

Save

This method saves a project definition to a file including all OCR and preprocessing settings of all pages and zones. .

The method triggers the OnSaveProject event so that possible project extensions can also save there settings to the same project file.

The method sets the Modified property to FALSE.

Syntax

```
Save (Filename As String, [CreateNewVersion As Boolean = FALSE],
[VersionInfo As String = ""])
```

Parameter	Description
Filename	Path to a compound file to store the project definition.
CreateNewVersion	Optional parameter The default value is 0 TRUE: The method does not overwrite the previous version in the file, but saves the project under a new version. FALSE: The method overwrites the previous version in the file.
VersionInfo	Optional parameter Used as description of the new version if the `CreateNewVersion` parameter is set to TRUE.

SetActivePage

This method sets the currently active page of a project.

The active page is the one which is displayed if you have set the project as a SCBCroViewer's display item.

Syntax

```
SetActivePage (Name As String)
```

Parameter	Description
Name	This parameter must be a valid name of a page in the current project.

See also

ActivePage

SetDefaultEngine

This method sets the default engine which is assigned to a newly created zone.

Syntax

```
SetDefaultEngine (EngineType As CroEngineType, EngineName As String,
ModifyGlobal As Boolean, Success As Boolean)
```

Parameter	Description
EngineType	Type of default engine to set.
EngineName	Name of the engine to set as default engine.
ModifyGlobal	Determines whether the default engine is set to local (inprocess) or global (modifying the registry)
Success	Indicates whether the engine to set as default is available.

SetLoadError

This method must be used to report an error condition to the project, if an error occurs in the OnLoadProject event .

The specified error code is used as return value from the Load or LoadVersion that has triggered the OnLoadProject event.

Syntax

```
SetLoadError (ErrorNumber As Long, ErrorDescription As String, ErrorSource
As String)
```

Parameter	Description
ErrorNumber	The ErrorNumber must be negative to specify an error code.
ErrorDescription	Description of the error.
ErrorSource	Name of the error source or module to specify the location where error occurred.

SetSaveError

This method must be used to report an error condition to the project, if an error occurs in the OnSaveProject event.

The specified error code is used as return value from the Save method that has triggered the OnLoadProject event.

Syntax

```
SetSaveError (ErrorNumber As Long, ErrorDescription As String, ErrorSource
As String)
```

Parameter	Description
ErrorNumber	Must be negative to specify an error code.
ErrorDescription	Description of the error.
ErrorSource	Name of the error source or module to specify the location where the error occurred.

SCBCroProject Properties

This object provides the following properties.

ActivePage

This property sets or returns the currently active page of the project.

The active page is displayed if you set the project as a SCBCroViewer’s DisplayItem.

Syntax

```
ActivePage As ISCBCroPage
```

See also

SetActivePage

Filename

This read-only property returns the file name of the last opened or saved project file.

Note: After creating an SCBCroProject instance or calling the Clear method, the property contains an empty string.

Syntax

```
Filename As String
```

HistoryDepth

This property sets or returns the maximum number of versions stored in the project file.

The default value is 0, which sets the number of saved project versions to unlimited.

Syntax

```
HistoryDepth As Long
```

Image

This property sets or returns the image assigned to the project.

The image is used as a background image when displaying the project in a SCBCroViewer. Every OCR operation within the project operates on this image.

Internally, the image is forwarded to each page of the project. Each following call to the SCBCroZone method Recognize is executed on this image.

To assign an image only to a single page of a project, use the SCBCroPage method Image.

Syntax

```
Image As ISCBCroImage
```

Modified

This property sets or returns the modification state of the project.

Note: Each Load, Save, or Clear call sets the property to FALSE.

Syntax

```
Modified As Boolean
```

Possible values

TRUE: The system assumes that the project has been modified.

FALSE: The system assumes that the project has not been modified.

Pages

This read-only property returns the Page Collection of the project.

The Page Collection is the root of the definition tree and contains all OCR related page, zone, recognition, and pre-processing definitions of the project.

Syntax

```
Pages As ISCBCroPages
```

See also

SCBCroPages

VersionCount

This read-only property returns the number of project versions stored in the project file.

A project file contains at least one version. A new version is added to the project each time the Save method is called with `CreateNewVersion = TRUE`.

Syntax

```
VersionCount (Filename As String) As Long
```

Parameter	Description
Filename	Path to a compound file containing the project definition, stored using the Save method.

SCBCroProject Events

This object provides the following events.

OnAnchorRenamed

This event triggers after an anchor was renamed using the properties dialog of the anchor.

Syntax

```
OnAnchorRenamed (OldName As String, NewName As String, Cancel As Boolean)
```

Parameter	Description
OldName	Old name of the anchor
NewName	New name of the anchor
Cancel	True: Cancel the renaming False: Execute the renaming

OnClearProject

This event triggers when the Clear method is executed.

Syntax

```
OnClearProject ()
```

OnLoadProject

This event triggers when the Load method is executed.

Syntax

```
OnLoadProject (AppStorage As ISCBCroStorage)
```

Parameter	Description
AppStorage	Use this interface to read from your own storage or streams inside the Cairo project file and load your settings from there.

OnPageRenamed

This event triggers after a page was renamed using the properties dialog of the page.

Syntax

```
OnPageRenamed (OldName As String, NewName As String, Cancel As Boolean)
```

Parameter	Description
OldName	Old name of the page
NewName	New name of the page

Cancel	True: Cancel renaming False: Execute renaming

OnSaveProject

This event triggers when the Save method is executed.

Syntax

```
OnSaveProject (AppStorage As ISCBCroStorage)
```

Parameter	Description
AppStorage	Use this interface to read from your own storage or streams inside the Cairo project file and load your settings from there.

OnZoneAdded

This event triggers after a zone was added using the mouse.

Syntax

```
OnZoneAdded (Page As String, Zone As String)
```

Parameter	Description
Page	Name of the page on which the zone was added
Zone	Name of the added zone

OnZoneRemoved

This event triggers when a zone is removed by pressing the DEL key.

Syntax

```
OnZoneRemoved (Page As String, Zone As String, Cancel As Boolean)
```

Parameter	Description
Page	Name of the page from which the zone was removed.
Zone	Name of the removed zone
Cancel	True: Cancel removing False: Execute removing

OnZoneRenamed

This event triggers after renaming a zone using the properties dialog of the zone.

Syntax

```
OnZoneRenamed (OldName As String, NewName As String, Cancel As Boolean)
```

Parameter	Description
OldName	Old name of the zone
NewName	New name of the zone
Cancel	True: Cancel renaming False: Execute renaming

Page Object SCBCroPage

The component SCBCroPage provides methods and properties to work with single pages.

SCBCroPage Methods

This object provides the following methods.

Locate

This method executes the locating algorithm of all anchors defined on this page and adjusts the position of all zones of the page.

If necessary, the preprocessing methods defined for the page are executed before locating the anchors.

Syntax

```
Locate ()
```

MoveAnchorBackground

This method moves an anchor to the foreground or background.

The method simplifies the manipulation of a large number of different rectangular objects, which can be crossed or overlapped.

Syntax

```
MoveAnchorBackground (strAnchorName As String, bForegroundDirection As
Boolean)
```

Parameter	Description
strAnchorName	Name of the anchor

bForegroundDirection	True: The anchor is in the foreground. False: The anchor is in the background.

MoveZoneBackground

This method moves a zone to the foreground or background.

The method simplifies the manipulation of a large number of different rectangular objects, which can be crossed or overlapped.

Syntax

```
MoveZoneBackground (strZoneName As String, bForegroundDirection As
Boolean)
```

Parameter	Description
strZoneName	Name of the zone
bForegroundDirection	True: The zone is in the foreground. False: The zone is in the background.

Preprocess

This method performs all defined preprocessing steps for the page.

The preprocessing methods are divided into two groups.

The first group executes before the anchors are located, while the second group executes afterwards.

All image enhancement methods should be executed before locating the anchors, but a line removal method should be executed after locating the anchors.

Syntax

```
Preprocess (StepType As CROcPagePrepStep)
```

Parameter	Description
StepType	Specifies which group of methods is executed. **Possible values** See CROcPagePrepStep

Reset

This method resets the internal states and releases all cached information, such as preprocessed images.

Syntax

```
Reset ()
```

SetActiveAnchor

Use this method to set the specified anchor as the currently selected object in the viewer. All previously selected objects are deselected.

Syntax

```
SetActiveAnchor (Name As String)
```

Parameter	Description
Name	A valid name of an anchor on this page.

SetActiveZone

Use this method to set the specified zone as the currently selected object in the viewer. All previously selected objects are deselected.

Syntax

```
SetActiveZone (Name As String)
```

Parameter	Description
Name	A valid name of zone on this page.

ShowDialog

This method displays the property dialog of the page.

Syntax

```
ShowDialog ()
```

ShowSelectedZonesDialog

This method displays a property dialog for editing all selected zones. If no zone is selected, no processing is performed.

Note: To select multiple zones, press CTRL and left-click on the required zones.

Syntax

```
ShowSelectedZonesDialog ()
```

SCBCroPage Properties

This object provides the following properties.

ActiveAnchor

This property sets or returns the reference of the currently selected anchor.

Use this property to deselect all other zones and anchors and make the specified anchor the selected object on the displayed page.

Note: If no anchor is selected, the property returns a NULL reference.

Syntax

```
ActiveAnchor As ISCBCroAnchor
```

See also

SetActiveAnchor

ActiveZone

This property sets or returns the reference of the currently selected zone.

Use this property to deselect all other zones and anchors and make the specified zone the selected object on the displayed page.

Note: If no zone is selected, the property returns a NULL reference.

Syntax

```
ActiveZone As ISCBCroZone
```

See also

SetActiveZone.

AllowRename

Use this property to allow or disallow interactive renaming of the page object.

If renaming is not allowed, the **Name** field appears disabled on the page properties dialog.

Syntax

```
AllowRename As Boolean
```

Anchors

This read-only property returns the collection of anchors of the page.

Syntax

```
Anchors As ISCBCroAnchors
```

Image

This property sets or returns the currently assigned image of the Page.

The image is used as background image when displaying the page within a SCBCroViewer. Each OCR operation on the page operates on this image.

Note: To assign an image to an entire project, use the SCBCroProject Image method.

Syntax

```
Image As ISCBCroImage
```

ImageRotation

This read-only property returns the rotation performed by all executed preprocessing methods to the current image.

Syntax

```
ImageRotation As Double
```

ImageTranslation

This read-only property returns the translation performed by all executed preprocessing methods to the current image.

Syntax

```
ImageTranslation (Direction As CroPrepDirection) As Long
```

Parameter	Description
Direction	Specifies whether to get the horizontal or the vertical translation.

Possible values

See CroPrepDirection

Locator

This read-only property returns the locator object that stores the information about which anchor is to be used to locate the entire page.

Syntax

```
Locator As ISCBCroLocator
```

Modified

This read-only property returns the modification state of the page.

Note: Each Load or Save call sets the property to FALSE.

Syntax

```
Modified As Boolean
```

Possible values

TRUE: The system assumes that the page has been modified.

FALSE: The system assumes that the page has not been modified.

Name

This property sets or returns the name of the page object.

Note: The name must be unique throughout the Cairo project.

Syntax

```
Name As String
```

PageState

This read-only property returns the current processing state of the page.

Before OCR can be executed, several task need to be performed at page or zone level. To optimize the runtime, these steps are cached by the page, so that `PageState` reflects the current processing state of the page.

Syntax

```
PageState As CROcPageState
```

PrepArray

This property sets or returns the preprocessing array that stores all preprocessing methods defined for this page.

Syntax

```
PrepArray As ISCBCroPrepArray
```

Zones

This read-only property provides access to the zone collection of the current page.

Syntax

```
Zones As ISCBCroZones
```

Pages Object SCBCroPages

The component SCBCroPages provides methods and properties to work with all contained page objects.

SCBCroPages Methods

This object provides the following methods.

Add

This method adds a new page with the specified name to the page collection.

The collection triggers the OnChange event.

The method returns the index of the inserted item within the collection. Use this index to access the item within the collection.

Syntax

```
Add (NewItem As ISCBCroPage, ItemName As String) As Long
```

Parameter	Description
NewItem	Pointer to a SCBCroPage object to be added to the pages collection.
ItemName	Name of the page item within the collection. Use this name to access the item within the collection.

Clear

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

If there are no further references to the collection items, this can result in all elements being released from memory.

The internal collection triggers the OnChange event.

Syntax

```
Clear ()
```

ItemExists

This method returns TRUE if an item with the specified name exists within the collection.

Syntax

```
ItemExists (Name As String) As Boolean
```

Parameter	Description
Name	Name of item to search for.

MoveItem

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

The method causes all items between OldIndex and NewIndex to change their index after successful operation.

The collection triggers the OnChange event.

Syntax

```
MoveItem (OldIndex As Long, NewIndex As Long)
```

Parameter	Description
OldIndex	Index of the item to move **Possible values** 1 to Count

NewIndex	New index of the item after moving it **Possible values** 1 to Count

Remove

This method removes the item specified by the `Name` parameter from the collection and releases the reference count to this item.

If there are no other references to the item, this may cause the item to be freed from memory.

The collection triggers the OnChange event.

Syntax

```
Remove (ItemName As String)
```

Parameter	Description
ItemName	Name of item to remove.

RemoveByIndex

Removes the item specified by the `Index` parameter from the collection and releases the reference count to this item.

If there are no other references to the item, this may cause the item to be freed from memory.

The collection triggers the OnChange event.

Syntax

```
RemoveByIndex (Index As Long)
```

Parameter	Description
Index	**Possible values** 1 to Count: Index of the item to remove

Rename

This method renames the item specified by `OldName` from `OldName` to `NewName`.

Syntax

```
Rename (OldName As String, NewName As String)
```

Parameter	Description

OldName	Name of item to rename.
NewName	New name of item in the collection.

SCBCroPages Properties

This object provides the following properties.

Collection

This read-only property returns the collection used internally to store the pages.

Access to the collection may be required to connect to the collection and receive change events.

During this method, AddRef is called to the interface, so the caller must release the returned interface pointer when it is no longer needed.

Syntax

```
Collection As ISCBCroCollection
```

Count

This read-only property returns the number of items within the page collection.

Syntax

```
Count As Long
```

Item

This read-only property returns a specified item from the collection.

The `Item` property is the default property of the ISCBCroPages interface.

During this method, AddRef is called to the interface, so the caller must release the returned interface pointer when it is no longer needed.

Syntax

```
Item (Index As Variant) As ISCBCroPage
```

Parameter	Description
Index	**Possible values** 1 to Count: Long value specifying the index within the collection A string specifying the item by name

ItemByIndex

This read-only property returns an item from the collection specified by the `Index` parameter.

During this method, AddRef is called to the interface, so the caller must release the returned interface pointer when it is no longer needed.

Syntax

```
ItemByIndex (Index As Long) As ISCBCroPage
```

Parameter	Description
Index	Index of the item to retrieve from the collection **Possible values** 1 to Count

ItemByName

This read-only property returns an item from the collection specified by the `Name` parameter.

During this method, AddRef is called to the interface, so the caller must release the returned interface pointer when it is no longer needed.

Syntax

```
ItemByName (Name As String) As ISCBCroPage
```

Parameter	Description
Name	Name of the item to retrieve from the collection.

ItemIndex

This read-only property returns the index of the item specified by the `Name` parameter.

Syntax

```
ItemIndex (Name As String) As Long
```

Parameter	Description
Name	Name specifying an item in the collection.

ItemName

This read-only property returns the name of an item specified by the `Index` parameter.

Syntax

```
ItemName (Index As Long) As String
```

Parameter	Description

Index	Index specifying an item in the collection **Possible values** 1 to Count

Zone Object SCBCroZone

The component SCBCroZone provides methods and properties to work with zones.

SCBCroZone Methods

This object provides the following methods.

LoadFromString

This method initializes a zone from a string generated by `SaveToString`.

This is useful if you do not want to save an entire Cairo project, but only a single zone with its settings.

Syntax

```
LoadFromString (LoadString As String)
```

Parameter	Description
LoadString	Generate this string using SaveToString.

Locate

This method executes the location algorithm of all anchors defined in this zone.

If necessary, the preprocessing methods defined for the page are executed before locating the anchors.

Syntax

```
Locate ()
```

Preprocess

This method performs all defined preprocessing steps for the zone.

Usually, the Recognize method calls the call Preprocess method.

Syntax

```
Preprocess ()
```

Recognize

This method performs the OCR within the coordinates of the zone using the image assigned to the page or passed as a parameter to this method.

The OCR result is stored in the Worktext object passed as first parameter. Depending on the assigned recognition settings object, the type of recognition can be either character recognition, barcode recognition or

Optical Mark Recognition (OMR).

Syntax

```
Recognize (Result As ISCBCroWorktext, [Image As ISCBCroImage = FALSE],
[PageNumber As Long = 1])
```

Parameter	Description
Result	Must contain a valid reference to a Worktext object, which can either be uninitialized or already contain data from a previously executed Recognize method. This method appends all new data, so that results from previous recognitions do not get lost.
Image	Optional parameter The default value is 0. **Possible values** **NULL**: Recognition performs on the image assigned to the page. **Reference to an image object**: Recognition performs on this image. To use preprocessing or locating features, assign the image to the page and set this parameter to NULL.
PageNumber	Optional parameter The default value is 0. This parameter is forwarded to the AppendImage method of the worktext object. To append the recognition results of several pages to one worktext object, you can specify a page number here that will then appear in the worktext object.

Reset

This method resets the internal states and releases all cached information, such as preprocessed images.

Syntax

```
Reset ([Context As CROResetContext = CROResetContextPage])
```

SaveToString

This method saves a zone to a string including all recognition and preprocessing settings.

The generated string can later be used as an argument for the LoadFromString method.

This is useful if you do not want to save an entire Cairo project, but only a single zone with its settings.

Syntax

```
SaveToString (SaveString As String)
```

Parameter	Description
SaveString	Receives a string containing all stored information in unencoded format.

ShowDialog

This method displays the property dialog of the zone.

Syntax

```
ShowDialog ()
```

SCBCroZone Properties

This object provides the following properties.

AllowRename

This property sets or returns whether interactive renaming of the zone object is allowed.

If renaming is not allowed, the **Name** text box is disabled in the properties dialog of the zone.

Syntax

```
AllowRename As Boolean
```

BottomAutoFittingRange

This property sets or returns the number of pixels to change the bottom border size of a zone when `AutoFitting` is enabled.

Syntax

```
BottomAutoFittingRange As Long
```

EnableZoneAutoFitting

This property sets or returns the flag for AutoFitting.

Syntax

```
EnableZoneAutoFitting As Boolean
```

Possible values

True: Enables the recognition engine to resize a zone for an analyzed word to improve recognition.

False: The zone is not resized for recognition.

EngineType

This property sets or returns the engine type related to a zone.

Syntax

```
EngineType As CroEngineType
```

FitToImgHeight

This property sets or returns whether the zone is set to the height of the image before performing OCR.

Note: If set to True, the Top and Height properties can change when calling the Recognize method.

Syntax

```
FitToImgHeight As Boolean
```

See also

FitToImgWidth

Height

This property sets or returns the height of the zone in pixels.

Note: Internally, the value is stored in millimeters, so that the value can change when an image with a different resolution is assigned to the page.

Syntax

```
Height As Long
```

Left

This property sets or returns the left coordinate of the zone in pixels.

Note: Internally, the value is stored in millimeters, so that the value can change when an image with a different resolution is assigned to the page.

Syntax

```
Left As Long
```

LeftAutoFittingRange

This property sets or returns the number of pixels to change the size of tze left border of a zone when AutoFitting is enabled.

Syntax

```
LeftAutoFittingRange As Long
```

Locator

This read-only property provides access to the locator object that stores the information about which anchor is to be used to locate the zone.

Syntax

```
Locator As ISCBCroLocator
```

Modified

This read-only property returns the modification state of the zone.

Note: Each Load or Save call sets the property to FALSE.

Syntax

```
Modified As Boolean
```

Possible values

TRUE: The system assumes that the zone has been modified.

FALSE: The system assumes that the zone has not been modified.

Name

This property sets or returns the name of the zone object.

Note: The name must be unique throughout the page.

Syntax

```
Name As String
```

PrepArray

This property sets or returns the preprocessing array that stores all preprocessing methods defined for the current zone.

Syntax

```
PrepArray As ISCBCroPrepArray
```

RecoSettings

This property sets or returns the recognition settings of the zone.

The actual type of this object depends on the selection made when creating or editing the zone.

Syntax

```
RecoSettings As Object
```

RightAutoFittingRange

This property sets or returns the number of pixels to change the right border size of a zone when `AutoFitting` is enabled.

Syntax

```
RightAutoFittingRange As Long
```

SelectState

This read-only property returns the current selection state of the zone.

Syntax

```
SelectState As CROZoneSelectState
```

ShowZoneBorder

This property sets or returns whether the zone borders are visible.

Syntax

```
ShowZoneBorder As Boolean
```

ShowZoneLabel

This property sets or returns whether the zone label which displays the zone name is visible.

Syntax

```
ShowZoneLabel As Boolean
```

Top

This property sets or returns the top coordinate of the zone in pixels.

Internally, the value is stored in millimeter, so the value may change if an image with a different resolution is assigned to the page.

Syntax

```
Top As Long
```

TopAutoFittingRange

This property sets or returns the number of pixels to change the top border size of a zone when `AutoFitting` is enabled.

Syntax

```
TopAutoFittingRange As Long
```

Visible

This property sets or returns whether the zone is visible inside a Viewer control.

Syntax

```
Visible As Boolean
```

Width

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

Internally, the value is stored in millimeters, so the value may change if an image with a different resolution is assigned to the page.

Syntax

```
Width As Long
```

ZoneColor

This property sets or returns the zone color used to display the zone area within a Viewer control.

Syntax

```
ZoneColor As OLE_COLOR
```

ZoneState

This read-only property returns the state of the zone.

Note: Before OCR, several task need to be performed at page or zone level. To optimize runtime cost, these steps are cached by the Zone, so that the `ZoneState` reflects the current processing state of the zone.

Syntax

```
ZoneState As CROcZoneState
```

Zones Object SCBCroZones

The component SCBCroZones provides access to all contained zone objects.

SCBCroZones Methods

This object provides the following methods.

Add

This method adds a new zone with the specified name to the zone collection. The collection triggers the OnChange event.

The method returns the index of the inserted item within the collection. Use this index to access the item within the collection.

Syntax

```
Add (NewItem As ISCBCroZone, ItemName As String) As Long
```

Parameter	Description
NewItem ItemName	Pointer to a SCBCroZone object to be added to the collection.
ItemName	Name of the zone item within the collection. Use this name to access the item within the collection.

Clear

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

If there are no other references to the item, this may cause the item to be freed from memory.

The internal collection trigger the OnChange event.

Syntax

```
Clear ()
```

ItemExists

This method returns TRUE if an item with the specified name exists inside the collection.

Syntax

```
ItemExists (Name As String) As Boolean
```

Parameter	Description
Name	Name of the item to search for

MoveItem

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

The method causes all items between OldIndex and NewIndex to change their index after successful operation.

The collection triggers the OnChange event.

Syntax

```
MoveItem (OldIndex As Long, NewIndex As Long)
```

Parameter	Description
OldIndex	Index of the item to move **Possible values** 1 to Count
NewIndex	New index of the item after moving it **Possible values** 1 to Count

Remove

This method removes the item specified by its name from the collection and releases its reference count. If there are no other references to the item, this may cause the item to be freed from memory. The collection triggers the OnChange event.

Syntax

```
Remove (ItemName As String)
```

Parameter	Description
ItemName	Name of the item to remove.

RemoveByIndex

Removes the item specified by its index from the collection and releases its reference count.

If there are no other references to the item, this may cause the item to be freed from memory.

The collection triggers the OnChange event.

Syntax

```
RemoveByIndex (Index As Long)
```

Parameter	Description
Index	Index of the item to remove **Possible values** 1 to Count

Rename

This method renames the item specified by `OldName` from `OldName` to `NewName`.

The collection triggers the OnChange event.

Syntax

```
Rename (OldName As String, NewName As String)
```

Parameter	Description
OldName	Name of the item to rename.
NewName	New name of the item within the collection.

SCBCroZones Properties

This object provides the following properties.

Collection

This read-only property returns the collection used internally to store the zones.

Access to the collection may be required to connect to the collection and receive change events.

Syntax

```
Collection As ISCBCroCollection
```

Count

This read-only property returns the number of items within the Zones collection.

Syntax

```
Count As Long
```

Item

This read-only property returns a specified item from the collection.

The `Item` property is the default property of the ISCBCroCollection interface.

Syntax

```
Item (Index As Variant) As ISCBCroZone
```

Parameter	Description
Index	**Possible values** 1 to Count: Long value specifying the index within the collection A string specifying the item by name

ItemByIndex

This read-only property returns an item from the collection specified by the `Index` parameter.

Syntax

```
ItemByIndex (Index As Long) As ISCBCroZone
```

Parameter	Description
Index	Index of the item to retrieve from the collection **Possible values** 1 to Count

ItemByName

This read-only property returns an item from the collection specified by the `Name` parameter.

Syntax

```
ItemByName (Name As String) As ISCBCroZone
```

Parameter	Description

Name	Name of the item to retrieve from the collection.

ItemIndex

This read-only property returns the index of the item specified by the `Name` parameter.

Syntax

```
ItemIndex (Name As String) As Long
```

Parameter	Description
Name	Name specifying an item in the collection.

ItemName

This read-only property returns the name of an item specified by the `Index` parameter.

Syntax

```
ItemName (Index As Long) As String
```

Parameter	Description
Index	Index specifying an item in the collection **Possible values** 1 to Count

Working with Collections (SCBCroCollLib)

The SCBCroCollLib provides the following object to work with collections.

Collection Object SCBCroCollection

The component SCBCroCollection provides methods and properties to work with collections.

This Collection object is used for different types of objects, such as Zones and Pages. It provides information about the names, indices, existence of items, as well as methods to modify them.

SCBCroCollection Methods

This object provides the following methods.

Add

This method adds a new item with the specified name to the collection.

The collection triggers the OnChange event.

The method returns the index of the inserted item within the collection. Use this index to access the item within the collection.

Syntax

```
Add (NewItem As Object, ItemName As String)
```

Parameter	Description
NewItem	IDispatch interface pointer to an object to be added to the collection.
ItemName	Name of the item within the collection. Use this name to access the item within the collection.

Clear

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

If there are no further references to the collection items, this can result in all elements being released from memory.

The collection triggers the OnChange event.

Syntax

```
Clear ()
```

GetMovedIndex

This method receives information about changes within the OnChange event.

If the `ChangeInfo` parameter of the OnChange event is set to `CollItemMoved`, you can use this method to get the old and new index of a moved item.

Syntax

```
GetMovedIndex (OldIndex As Long, NewIndex As Long)
```

Parameter	Description
OldIndex	Index of changed item before moving it.
NewIndex	Index of changed item after moving it.

GetRenameInfo

This method receives information about changes within the OnChange event.

If the `ChangeInfo` parameter of the OnChange event is set to `CollItemRenamed`, you can use this method to get the old and new name of a renamed item.

Syntax

```
GetRenameInfo (OldName As String, NewName As String)
```

Parameter	Description
OldName	Name of the changed item before renaming it.
NewName	Name of the changed item after renaming it.

ItemExists

This method checks whether the item exists inside the collection.

Syntax

```
ItemExists (Name As String) As Boolean
```

Parameter	Description
Name	Name of item to search for.

MoveItem

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

The method causes all items between `OldIndex` and `NewIndex` to change their index after successful operation.

The collection triggers the OnChange event.

Syntax

```
MoveItem (OldIndex As Long, NewIndex As Long)
```

Parameter	Description
OldIndex	Index of the item to move. **Possible values** 1 to Count
NewIndex	New index of the item after the moving it. **Possible values** 1 to Count

Remove

This method removes the item specified by the `Name` parameter from the collection and releases the reference count to this item.

If there are no other references to the item, this may cause the item to be freed from memory.

The collection triggers the OnChange event.

Syntax

```
Remove (ItemName As String)
```

Parameter	Description
ItemName	Name of the item to remove.

RemoveByIndex

This method removes the item specified by the `Index` parameter from the collection and releases the reference count to this item.

If there are no other references to the item, this may cause the item to be freed from memory.

The collection triggers the OnChange event.

Syntax

```
RemoveByIndex ( As Long)
```

Parameter	Description
Index	Index of item to remove **Possible values** 1 to Count

Rename

This method renames the item specified by `OldName` from `OldName` to `NewName`.

The collection triggers the OnChange event.

Syntax

```
Rename (OldName As String, NewName As String)
```

Parameter	Description
OldName	Name of the item to rename
NewName	New name of the item within the collection

SCBCroCollection Properties

This object provides the following properties.

ChangedItemIndex

This read-only property returns information about changes within the OnChange event.

To get the index of the changed item , retrieve the `ChangeInfo` parameter of the OnChange event to check if it is `CollItemAdded`, `CollItemPreRemove`, `CollItemRemoved`, or `CollItemRenamed`.

If you call this method outside the `OnChange` event, it returns -1.

Syntax

```
ChangedItemIndex As Long
```

Count

This read-only property returns the number of items within the collection.

Syntax

```
Count As Long
```

Item

This read-only property returns a specified item from the collection.

The `Item` property is the default property of the ISCBCroCollection interface.

The return value also contains an interface pointer to the specified item.

During this method, AddRef is called to the interface, so the caller must release the returned interface pointer when it is no longer needed.

Syntax

```
Item (Index As Variant) As Object
```

Parameter	Description
Index	**Possible values** 1 to Count: Long value specifying the index within the collection A string specifying the item by name

ItemByIndex

This read-only property returns an item from the collection specified by the `Index` parameter.

During this method, AddRef is called to the interface, so the caller must release the returned interface pointer when it is no longer needed.

Syntax

```
ItemByIndex (Index As Long) As Object
```

Parameter	Description
Index	Index of the item to retrieve from the collection **Possible values** 1 to Count

ItemByName

This read-only property returns an item from the collection specified by the `Name` parameter.

The return value also contains an interface pointer to the specified item.

During this method, AddRef is called to the interface, so the caller must release the returned interface pointer when it is no longer needed.

Syntax

```
ItemByName (Name As String) As Object
```

Parameter	Description
Name	Name of the item to retrieve from the collection.

ItemIndex

This read-only property returns the index of the item specified by the `Name` parameter.

Syntax

```
ItemIndex (Name As String) As Long
```

Parameter	Description
Name	Name specifying an item in the collection.

ItemName

This read-only property returns the name of an item specified by the `Index` parameter.

Syntax

```
ItemName (Index As Long) As String
```

Parameter	Description

Index	Index specifying an item in the collection **Possible values** 1 to Count

Tag

This property sets or returns a variant for each item of the collection.

This value is not saved.

Syntax

```
    Tag (Index As Long) As Variant
```

Parameter	Description
Index	Index specifying an item in the collection **Possible values** 1 to Count

SCBCroCollection Events

This ISCBCroCollectionEvents object provides the following event.

OnChange

This event triggers each time the collection is changed.

Syntax

```
    OnChange (ChangeInfo As CollChangeInfo)
```

Parameter	Description
ChangeInfo	**Possible values** See CollChangeInfo

Type Definition CollChangeInfo

This type definition provides detailed information on the type of changes made to the collection.

Type	Description
CollItemAdded	The item was added to the collection using the Add method. To get the index of the new item, use ChangedItemIndex.

CollItemMoved	The item was moved within the collection using MoveItem To get the old and the new index of the moved item, use GetMovedIndex.
CollItemPreRemove	To get the current index of the item before removing it from the collection, use ChangedItemIndex. You can access the item within this event, see also `CollItemRemoved` below.
CollItemPreReset	This type is set before the collection is reset by the Clear, see also `CollItemReset` below.
CollItemRemoved	The item was removed from the collection. To get the index of the removed item, use ChangedItemIndex. It is impossible to access the item by this index, because it has already been removed. See also `CollItemPreRemove` above.
CollItemRenamed	The item was renamed using the Rename method. To get the index of the renamed item, use ChangedItemIndex. To get the old and the new name of the renamed item, use GetRenameInfo.
CollItemReset	Collection was reset using the Clear method. References to all items have been released. The Count property has been set to 0

Working with Worktext Objects (SCBCroWorktextLib)

OCR recognition results and text from CIDocs can be saved in a worktext object. The worktext object contains the raw text including geometric information.

The SCBCroWorktextLib provides the following objects.

- SCBCroWorktext

SCBCroWorktextLib Type Definitions

The SCBCroWorktextLib object provides the following type definitions.

CroWorktextDirection

This type definition specifies the worktext direction.

Type	**Description**
CroWorktextDirection_Horizontal	Worktext direction is horizontal.
CroWorktextDirection_Vertical	Worktext direction is vertical.

DimensionInfo_t

This type definition specifies the location in millimeters (mm) or pixels.

Type	Description
MM_Height	Height in mm
MM_Left	Left in mm
MM_Top	Top in mm
MM_Width	Width in mm
PX_Height	Height in pixels
PX_Left	Left in pixels
PX_Top	Top in pixels
PX_Width	Width in pixels

PosInfo_t

This type definition specifies the position of the character.

Type	Description
POS_Col	Column index
POS_Img	Image index
POS_Line	Line index
POS_Zone	Zone index

WktConversionType

This type definition specifies the worktext conversion type.

Type	Description
WorktextConvLowerCase	Worktext is converted to lower case.
WorktextConvUpperCase	Worktext is converted to upper case.

SCBCroWorktext

The SCBCroWorktext library provides methods and properties to modify the worktext object.

SCBCroWorktext Methods

This object provides the following methods:

AppendAlternative

This method appends one or more alternative characters to the active character. The active character is the last one added to the worktext object.

Note: A string longer than one character added to a worktext object using the AppendString method or the Text property makes the last character of the string to the active character.

Syntax

```
AppendAlternative (Value as String, Confidence as Long, Top as Long, Left
as Long, Height as Long, Width as Long)
```

Parameter	Description
Value	Alternative characters.
Confidence	Confidence of the alternative characters. Optional parameter. The default value is 100.
Top	Top position of the alternative characters. Optional parameter. The default value is 0.
Left	Left position of the alternative characters. Optional parameter. The default value is 0.
Height	Height of the alternative characters. Optional parameter. The default value is 0.
Width	Width of the alternative characters. Optional parameter. The default value is 0.

AppendImage

This method adds a new page reference to the worktext.

The page reference contains various zones. To add a new zone to a page reference, use AppendZone.

Zones contain different lines. To add a new line to a zone, use AppendLine.

Lines contain text. To add text to a line, use AppendString or AppendReject.

The added zones, lines and strings always refer to the page added by AppendImage.

Syntax

```
AppendImage (ImgNr as Long, XRes as Long, YRes as Long)
```

Parameter	Description
ImgNr	Number of the current image.
XRes	Resolution in horizontal direction in dots per inch.
YRes	Resolution in vertical direction in dots per inch.

AppendLine

This method adds a new line.

Text added to the worktext object using AppendString or AppendReject always refers to the line generated by AppendLine. To add text to a new line, use the AppendLine method first.

Syntax

```
AppendLine (Top as Long, Left as Long, Height as Long, Width as Long)
```

Parameter	Description
Top	Top position of the line in pixels. Optional parameter. The default value is 0.
Left	Left position of the line in pixels. Optional parameter. The default value is 0.
Height	Height of the line in pixels. Optional parameter. The default value is 0.
Width	Width of the line in pixels. Optional parameter. The default value is 0.

AppendReject

This method appends a rejected character to the current worktext.

A reject is a symbol that a recognition engine could not recognize.

You need to specify the page, zone and line indices before by using AppendImage, AppendZone and AppendLine.

Syntax

```
AppendReject (Top as Long, Left as Long, Height as Long, Width as Long)
```

Parameter	Description
Top	Top position of the reject in pixels. Optional parameter. The default value is 0.
Left	Left position of the reject in pixels. Optional parameter. The default value is 0.
Height	Height of the reject in pixels. Optional parameter. The default value is 0.
Width	Width of the reject in pixels. Optional parameter. The default value is 0.

AppendString

This method appends one or more characters to the worktext object and makes the last character of the string to the active character. You need to specify the page, zone and line indices before using AppendImage, AppendZone and AppendLine. Methods and properties such as AppendAlternative refer to the active character.

Syntax

```
AppendString (Value as String, Confidence as Long, Top as Long, Left as
Long, Height as Long, Width as Long)
```

Parameter	Description
Value	The string to add to the worktext, can consist of one or more characters.
Confidence	Current confidence of the value. Optional parameter. The default value is 100. **Possible values** 0 to 100, where 0 means that the recognition is unconfident, and 100 means that the recognition is confident.
Top	Top position of the zone in pixels. Optional parameter. The default value is 0.
Left	Left position of the zone in pixels. Optional parameter. The default value is 0.
Height	Height of the zone in pixels. Optional parameter. The default value is 0.

Width	Width of the zone in pixels Optional parameter. The default value is 0.

AppendTag

This method appends a tag to the taglist and assigns it to the specified character.

A tag contains information associated to a single character.

Syntax

```
AppendTag (CharIndex as Long, TagType as Long, TagValue as Variant)
```

Parameter	Description
CharIndex	Zero-based index of the character being described by the tag.
TagType	Describes the meaning of the TagValue value.
TagValue	Value that describes the character.

AppendTo

This method appends some or all characters including the character information to another worktext object.

Syntax

```
AppendTo (pDestination as ISCBCroWorktext, StartPos as Long, CharCount as
Long)
```

Parameter	Description
pDestination	Interface pointer of the destination Worktext object.
StartPos	Index of the first character of the source worktext object to append to the destination worktext object.
CharCount	Number of characters to copy from the source worktext object to the destination worktext object.

AppendZone

This method adds a zone to the current page.

Note: Text added to a worktext object using AppendString or AppendReject always refers to the line added by AppendZone.

Syntax

```
AppendZone (Top as Long, Left as Long, Height as Long, Width as Long)
```

Parameter	Description
Top	Top position of the zone in pixels. Optional parameter. The default value is 0.
Left	Left position of the zone in pixels. Optional parameter. The default value is 0.
Height	Height of the zone in pixels. Optional parameter. The default value is 0.
Width	Width of the zone in pixels. Optional parameter. The default value is 0.

ApplyBackwardLanguageConversion

This method applies backward language conversion to support non-western languages.

Syntax

```
ApplyBackwardLanguageConversion()
```

ApplyForwardLanguageConversion

This method applies forward language conversion to support non-western languages.

Syntax

```
ApplyForwardLanguageConversion(Language as Long, UseSingleChar as Boolean)
```

Parameter	Description
Language	Language value
UseSingleChar	True / False

CharDim

This method returns the dimension of the specified character.

Syntax

```
CharDim (dInfo as DimensionInfo_t, CharIndex as Long)
```

Parameter	Description
dInfo	Parameter to specify which kind of information returns. See also DimensionInfo_t

CharIndex	Index of the character.

Clear

This method removes all information contained in the worktext object.

Syntax

```
Clear ()
```

ConvertCharacters

This method converts the worktext to upper or lower case.

Syntax

```
ConvertCharacters (ConvType as WktConversionType)
```

Parameter	Description
ConvType	Specifies the worktext conversion type. See also WktConversionType

Copy

This method copies characters from a source worktext object to a destination worktext object.

Syntax

```
Copy (pDestination as ISCBCroWorktext, StartPos as Long, CharCount as
Long)
```

Parameter	Description
pDestination	Interface pointer of the destination worktext object.
StartPos	Index of the first character of the source worktext object to copy.
CharCount	Number of characters to copy to the destination worktext object.

CorrectPageReferences

This method corrects the page references.

Syntax

```
CorrectPageReferences(Start as Long, Offset as Long)
```

Parameter	Description
Start	Start position
Offset	Offset

FindString

This method searches a string or a substring in the current worktext.

Syntax

```
FindString (StartPos as Long, Search as String) as Long
```

Parameter	Description
StartPos	Position to start searching.
Search	Text to search.

Return value

Index of the first character in the worktext, where the search string is found.

or

"-1" if no string is found.

GetAlternativeChar

This method returns the alternative item for the item specified by the CharIndex parameter.

Syntax

```
GetAlternativeChar (CharIndex as Long, AltNr as Long) as String
```

Parameter	Description
CharIndex	Index of the character for which the alternative is required.
AltNr	Index of the alternative.

GetAlternativeCharConfidence

This method returns the confidence level of an alternative item.

Syntax

```
GetAlternativeCharConfidence (CharIndex as Long, AltNr as Long) as Long
```

Parameter	Description
CharIndex	Index of the character to which the alternative character is related to.
AltNr	Index of the alternative character.

GetAlternativeCount

This method returns the number of alternative characters for a character specified by the CharIndex parameter.

Syntax

```
GetAlternativeCount (CharIndex as Long) as Long
```

Parameter	Description
CharIndex	Index of the character for which the number of alternatives returns.

GetBoostReference

This method returns the attributes of the boost character alternative.

Syntax

```
GetBoostReference(CharIndex as Long, pCharCode as Long, pConfidence as
Long, pAttributes as Long)
```

Parameter	Description
CharIndex	Index of the character.
pCharCode	Character code.
pConfidence	Current confidence of the value. **Possible values** 0 to 100, where 0 means that the recognition is unconfident, and 100 means that the recognition is confident.
pAttributes	Attributes

GetCharAttributes

This method returns the character, character attributes, and the confidence for a character in the worktext specific by `CharIndex`.

Syntax

```
GetCharAttributes(CharIndex as Long, pChar as Long, pAttributes as Long,
    pConfidence as Long)
```

Parameter	Description
CharIndex	Index of the character.
pChar	Character
pAttributes	Attributes
pConfidence	Current confidence of the value. **Possible values** 0 to 100, where 0 means that the recognition is unconfident, and 100 means that the recognition is confident.

GetCharConfidence

This method returns the confidence level of a character.

Syntax

```
    GetCharConfidence (CharIndex as Long) as Long
```

Parameter	Description
CharIndex	Index of the item.

Return value

Possible return values are between 0 and 100.

Sample return values

- 0: The recognition is unconfident.
- 50: The recognition has a low confidence.
- 100: The recognition is confident.

GetCharIndex

This method returns the index of a character in the worktext object defined by Lineindex and Columnindex.

Syntax

```
    GetCharIndex (Line as Long, Col as Long) as Long
```

Parameter	Description
Line	Line of the character.
Col	Column of the character.

GetCharInfo

This method returns the page, zone, line, or column index of a character.

Syntax

```
GetCharInfo (posInfo as PosInfo_t, CharIndex as Long) as Long
```

Parameter	Description
posInfo	Parameter to choose the type of information returned. See also PosInfo_t
CharIndex	Index of the character.

GetTag

This method returns a tag of the taglist.

A tag contains information associated to a single character.

Syntax

```
GetTag (TagIndex as Long, CharIndex as Long, TagType as Long, TagValue as
Variant)
```

Parameter	Description
TagIndex	Index specifying the tag's index.
CharIndex	Index of the character described by the tag.
TagType	Type of the tag.
TagValue	The tag itself.

InsertAfter

This method inserts a text in the worktext object after the specified index.

Syntax

```
InsertAfter (CharIndex as Long, value as String, [Confidence as Long =
100], [Top as Long = FALSE], [Left as Long = FALSE], [Height as Long =
eFALSE], [Width as Long = FALSE])
```

Parameter	Description
CharIndex	Character index to insert.

Value	String or character value to add.
Confidence	Confidence of the added value. **Possible values** 0 to 100, where 0 means that the recognition is unconfident, and 100 means that the recognition is confident. Optional parameter. The default value is 100.
Top	Top position of the text in pixels. Optional parameter. The default value is 0.
Left	Left position of the text in pixels. Optional parameter. The default value is 0.
Height	Height of the text in pixels. Optional parameter. The default value is 0.
Width	Width of the text in pixels Optional parameter. The default value is 0.

InsertBefore

This method inserts a text in the worktext object before the specified index.

Syntax

```
InsertBefore (CharIndex as Long, value as String, Confidence as Long, Top
as Long, Left as Long, Height as Long, Width as Long)
```

Parameter	Description
CharIndex	Character index before which the text is inserted.
Value	Added string or character value.
Confidence	Confidence of the added value. **Possible values** 0 to 100, where 0 means that the recognition is unconfident, and 100 means that the recognition is confident. Optional parameter. The default value is 100.
Top	Top position of the characters in pixels. Optional parameter. The default value is 0.
Left	Left position of the characters in pixels. Optional parameter. The default value is 0.

Height	Height of the characters in pixels. Optional parameter. The default value is 0.
Width	Width of the characters in pixels. Optional parameter. The default value is 0.

InsertRejectAfter

This method inserts a reject character in the worktext after the specified position.

Syntax

```
InsertRejectAfter (CharPos as Long, Top as Long, Left as Long, Height as
Long, [Width as Long)
```

Parameter	Description
CharPos	Reject character to insert.
Top	Top position of the reject character in pixels. Optional parameter. The default value is 0.
Left	Left position of the reject character in pixels. Optional parameter. The default value is 0.
Height	Height of the reject character in pixels. Optional parameter. The default value is 0.
Width	Width of the reject character in pixels. Optional parameter. The default value is 0.

InsertRejectBefore

This method inserts a reject character in the worktext before the specified character position.

Syntax

```
InsertRejectBefore (CharPos as Long, Top as Long, Left as Long, Height as
Long, Width as Long)
```

Parameter	Description
CharPos	Position to insert before.
Top	Top position of the reject character in pixels. Optional parameter. The default value is 0.

Left	Left position of the reject character in pixels. Optional parameter. The default value is 0.
Height	Height of the reject character in pixels. Optional parameter. The default value is 0.
Width	Width of the reject character in pixels. Optional parameter. The default value is 0.

IsCharValid

This method returns TRUE if the worktext entry specified by CharIndex is a valid character, or FALSE if the entry is a reject.

Syntax

```
IsCharValid (CharIndex as Long) as Boolean
```

Parameter	Description
CharIndex	Index of requested character.

JoinNextLine

This method removes the CRLF of the specified line.

After calling JoinNextLine, the line that had the index (LineIndex + 1) before the call receives the index LineIndex.

A successful call of JoinNextLine reduces the number of lines by 1.

Syntax

```
JoinNextLine (LineIndex as Long)
```

Parameter	Description
LineIndex	Index of the line.

LineDim

This method returns the dimension of the specified line.

Syntax

```
LineDim (dInfo as DimensionInfo_t, LineIndex as Long) as Long
```

Parameter	Description
dInfo	Parameter to determine the type of information returned. See also DimensionInfo_t
LineIndex	Index of the line.

LineText

This method returns the text of a specified line.

Syntax

```
LineText (LineIndex as Long) as String
```

Parameter	Description
LineIndex	Index of the line.

Load

This method loads the worktext from a file.

Syntax

```
Load (Filename as String)
```

Parameter	Description
Filename	Path and filename of the file that contains the worktext.

PackRejects

This method combines a sequence of rejects to a single reject.

Syntax

```
PackRejects (PackLimit as Long, Confidence as Long, StartPos as Long,
Length as Long)
```

Parameter	Description
PackLimit	Minimal length of the reject sequence to combine to a single reject.

Confidence	Confidence limit for a character. Characters with a lower confidence are treated as rejects. **Possible values** 0 to 100
StartPos	Index of the first character.
Length	Number of characters to process.

Read

This method returns characters of a worktext object as string.

Syntax

```
Read (StartPos as Long, Length as Long) as String
```

Parameter	Description
StartPos	Position of the first character to return.
Length	Number of characters to return. Optional parameter. The default value is 1.

RemDelimiters

This method removes characters from the worktext object.

Syntax

```
RemDelimiters (Delimiters as String)
```

Parameter	Description
Delimiters	Characters to remove from the worktext. To combine all lines to a single line, use the symbol for CRLF as parameter in this method.

Remove

This method removes the specified number of characters from the worktext.

Syntax

```
Remove (CharIndex as Long, Length as Long)
```

Parameter	Description
CharIndex	The first character to remove.
Length	Number of characters to remove. Optional parameter. The default value is 1.

RemoveLine

This method removes a line from the worktext.

Syntax

```
RemoveLine (LineIndex as Long)
```

Parameter	Description
LineIndex	Index of the line.

ReplaceLineLocation

This method replaces the location of the specified line.

Syntax

```
ReplaceLineLocation(PositionType as DimensionInfo_t, _LineIndex as Long, _
NewPosition as Long)
```

Parameter	Description
PositionType	Type of the position. See also DimensionInfo_t
LineIndex	Index of the line.
NewPosition	New position of the line.

ReplaceLineText

This method replaces the text of the specified line.

Syntax

```
ReplaceLineText(LineIndex as Long, NewLineText as String)
```

Parameter	Description
LineIndex	Index of the line.

NewLineText	New text of the line.

Save

This method saves the worktext object to a file.

Note: The method overwrites existing files.

Syntax

```
Save (Filename as String)
```

Parameter	Description
Filename	Path and name of the file to create or overwrite.

SetBoostReference

This method assigns the attributes of the boosted character alternative.

Syntax

```
SetBoostReference(CharIndex as Long, CharCode as Long, Confidence as Long,
Attributes as Long)
```

Parameter	Description
CharIndex	Index of requested character.
CharCode	Character code.
Confidence	Confidence of the character. Optional parameter. The default value is 100. **Possible values** 0 to 100, where 0 means that the recognition is unconfident, and 100 means that the recognition is confident.
Attributes	Attributes to assign.

SetCharAttributes

This method assigns attributes to a character.

Syntax

```
SetCharAttributes(CharIndex as Long, Char as Long, Attributes as Long,
Confidence as Long)
```

Parameter	Description
CharIndex	Index of requested character.
Char	The character
Attributes	Attributes to assign.
Confidence	Confidence of the character. Optional parameter. The default value is 100. **Possible values** 0 to 100, where 0 means that the recognition is unconfident, and 100 means that the recognition is confident.

SetConfidence

This method sets the confidence of a character.

Syntax

```
SetConfidence (CharIndex as Long, NewValue as Long)
```

Parameter	Description
CharIndex	Index of the character which confidence is set.
NewValue	Confidence of the character. **Possible values** 0 to 100, where 0 means that the recognition is unconfident, and 100 means that the recognition is confident.

Substitute

This method substitutes a part of the characters in the worktext object.

Note: The method substitutes only the characters, but not the information associated to the characters, such as position or tag.

Syntax

```
Substitute (CharIndex as Long, Length as Long, value as String)
```

Parameter	Description
CharIndex	Position of the first character to substitute.
Length	Number of characters to substitute.

Value	String that replaces the worktext characters.

TransformCoordinates

This method transforms the coordinates of the worktext objects like zones, lines, and characters.

Note: Undoing the coordinate transformation can be performed in a second call of TransformCoordinates using –Dx, -Dy, -RotAngle as parameters if Dx, Dy, RotAngle were the first call of TransformCoordinates parameters.

Syntax

```
TransformCoordinates (ImgNr as Long, Dx as Long, Dy as Long, RotAngle as
Double)
```

Parameter	Description
ImgNr	Index of the image.
Dx	Specifies the translation in horizontal direction.
Dy	Specifies the translation in vertical direction.
RotAngle	Specifies the rotation angle. The center point is the top left corner of the document.

ZoneDim

This method returns the dimension of the specified zone.

Syntax

```
ZoneDim (dInfo as DimensionInfo_t, ZoneIndex as Long) as Long
```

Parameter	Description
dInfo	Parameter to choose specific information. See also DimensionInfo_t
ZoneIndex	Index of the zone.

SCBCroWorktext Properties

This object provides the following properties.

BulkUpdate

This property sets or returns if the worktext object executes the change events.

If the BulkUpdate property is set to true, the worktext object does not trigger the OnChange events.

Use this property to improve performance when adding or changing multiple worktext characters.

Syntax

```
BulkUpdate as Boolean
```

CountBase

This property sets or returns the character index offset.

To start all indices at 0, set CountBase to 0. To start all indices at n, set CountBase to n.

Syntax

```
CountBase as Long
```

GetTagCount

This read-only property returns the number of available tags.

Syntax

```
GetTagCount as Long
```

ImageCount

This read-only property returns the number of images assigned to the worktext object.

Syntax

```
ImageCount as Long
```

ImageRotation

This property sets or returns the clockwise rotation angle of the image. The rotation angle is the angle around which the image was rotated before recognition. The center point is the left top corner of the image.

Syntax

```
ImageRotation (ImgIdx as Long, newVal as Double)
```

Parameter	Description
ImgIdx	Index of the image.
newVal	Rotation angle **Note:** The parameter value adds to the current rotation, it does not replace it.

ImageTranslation

This property sets or returns the image translation.

Syntax

```
ImageTranslation (ImgIdx as Long, Direction as CroWorktextDirection) as
Long
```

Parameter	Description
ImgIdx	Index of the image.
Direction	Specifies the coordination type. See also CroWorktextDirection

ImageXRes

This read-only property returns the image resolution in horizontal direction.

Syntax

```
ImageXRes (ImgIdx as Long) as Long
```

Parameter	Description
ImgIdx	Index of the image.

ImageYRes

This read-only property returns the image resolution in vertical direction.

Syntax

```
ImageYRes (ImgIdx as Long) as Long
```

Parameter	Description
ImgIdx	Index of the image.

LanguageTranslationMethod

This property sets or returns the current language transcode method identifier.

Syntax

```
LanguageTranslationMethod as Long
```

LineCount

This read-only property returns the number of lines of the worktext.

Syntax

```
LineCount as Long
```

LineLength

This read-only property returns the length of a line in pixels.

Syntax

```
LineLength (LineIndex as Long) as Long
```

Parameter	Description
LineIndex	Index of the line.

LineStart

This read-only property returns the character index of the first character in the line.

Syntax

```
LineStart (LineIndex as Long) as Long
```

Parameter	Description
LineIndex	Index of the line.

RejectChar

This property sets or returns the character used to symbolize rejects.

Note: If a recognition engine cannot identify a symbol, it returns a reject. The worktext stores the rejects. The default reject character is "?".

Syntax

```
RejectChar as String
```

RejectCount

This read-only property returns the number of rejects.

Syntax

```
RejectCount as Long
```

Tag

This property sets or returns the content of the tag.

Syntax

```
Tag as String
```

Text

This property sets or returns the characters stored in the worktext object.

Note: Setting new characters initializes the values describing the characters, such as position, with default values.

Syntax

```
Text as String
```

TextLength

This read-only property returns the number of characters stored in the worktext object.

Use this property instead of the WrinWrap method `Len`.

Syntax

```
TextLength as Long
```

Value

This property sets or returns a string that contains the complete text information contained in the worktext object.

Syntax

```
Value as String
```

ZoneCount

This read-only property returns the number of zones assigned to the worktext object.

Syntax

```
ZoneCount as Long
```

SCBCroWorktext Events

This object provides the following event.

OnChange

This event triggers each time the Worktext is changed.

The `ChangeInfo` parameter provides detailed information about the type of change that took place.

Syntax

```
OnChange (CharIndex As Long, ChangeInfo As Long)
```

Parameter	Description
CharIndex	Zero-based index of the character for which the change took place.
ChangeInfo	Provides detailed information, about the type of change made to the worktext

Working with Verification Forms (DISTILLERVERIFIERCOMPLib)

The Cedar Verifier Component library DISTILLERVERIFIERCOMPLib provides methods and properties to work with verification forms and verification form elements.

DISTILLERVERIFIERCOMPLib Type Definitions

The DISTILLERVERIFIERCOMPLib provides the following type definition.

CdrVerifierFieldType

This type definition specifies the Verifier field types. 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

SCBCdrVerificationForm

Use this interface to set specific verification form properties, as well as to set default properties for embedded elements, such as verification fields, labels, tables, and buttons.

In Web Verifier, use these methods in the VerifiedFormatLoad event.

SCBCdrVerificationForm Methods

The SCBCdrVerificationForm provides the following method.

SetFieldFocus

This method sets the focus to the specified field or table cell and updates the `HighlightField`, `HighlightColumnIndex`, and `HighlightRowIndex` settings.

The method returns an error message if the specified field or table cell is hidden or does not exists on the verification form.

Do not use this method in either VerifierFormLoad or in any Validate field or table events. These events often execute in sequence and may affect the focus following each event, independent of SetFieldFocus.

Carefully use this method within the FocusChanged or CellFocusChanged events, as an endless loop may result.

Syntax

```
SetFieldFocus(BSTR FormName, ISCBCdrWorkdoc pWorkdoc, BSTR bstrFieldName,
BSTR bstrTableColumnName, long lTableRowIndex)
```

Parameter	Description
FormName	Verification form name
pWorkdoc	The current workdoc
bstrFieldName	Field name
bstrTableColumnName	Column name
lTableRowIndex	Row index **Note:** The SetFieldFocus action is cancelled when the RowIndex value is "-1".

Sample Code

The following sample code sets the focus to the field "Table" in the first row of the column "Quantity".

```
Private Sub Document_OnAction(pWorkdoc as SCBCdrPROJLib.ISCBCdrWorkdoc,
ByVal ActionName as String) If ActionName = "GoToField" Then
Project.SetFieldFocus("Form_Invoices_1", pWorkdoc, "Table", "Quantity", 0)
 End If End Sub
```

SCBCdrVerificationForm Properties

The SCBCdrVerificationForm provides the following properties.

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.

Syntax

```
DefaultElementBackgroundColorInvalid 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.

Syntax

```
DefaultElementBackgroundColorValid as OLE_COLOR
```

DefaultFieldFont

This property sets or returns the default font for all verification field elements available on this verification form.

Syntax

```
DefaultFieldFont as StdFont
```

DefaultFieldFontColor

This property sets or returns the default color for all verification field elements available on this verification form.

Syntax

```
DefaultFieldFontColor as OLE_COLOR
```

DefaultLabelBackgroundColor

This property sets or returns the default background color for all label elements available on this verification form.

Syntax

```
DefaultLabelBackgroundColor as OLE_COLOR
```

DefaultLabelFont

This property sets or returns the default font for all label elements available on this verification form.

Syntax

```
DefaultLabelFont as StdFont
```

DefaultLabelFontColor

This property sets or returns the default color for all label elements available on this verification form.

Syntax

```
DefaultLabelFontColor as OLE_COLOR
```

Sample Code

```
Dim clrDefaultColor as OLE_COLOR clrDefaultColor = -1 the
Form.VerificationLabels.ItemByIndex(lNextLabelIndex).FontColor =
clrDefaultColor
```

FormBackgroundColor

This property sets or returns the background color for the form.

Syntax

```
FormBackgroundColor as OLE_COLOR
```

FormBackgroundColorDI

This property sets or returns the background color for the Direct Input control on the form that means for the area around the Direct Input field.

Syntax

```
FormBackgroundColorDI as OLE_COLOR
```

SCBCdrVerificationField

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.

SCBCdrVerificationField Properties

The CdrVerifiedFieldType provides the following properties.

AutoCompletionEnabled

This property sets or returns the auto-completion setting for a verification field.

Syntax

```
AutoCompletionEnabled as Boolean
```

Sample Code

The following sample code turns Auto Completion on for the Invoice Number field.

```
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
```

BackgroundColorInvalid

This property sets or returns 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.

Syntax

```
BackgroundColorInvalid as OLE_COLOR
```

Sample Code

The following sample code sets the background color for the Invoice Number field to gray if it is Invalid.

```
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)
```

BackgroundColorValid

This property sets or returns 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.

Syntax

```
BackgroundColorValid as OLE_COLOR
```

Sample Code

The following sample code sets the color for the Invoice Number field to gray if it is Valid.

```
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").BackgroundColorValid = RGB (192, 129, 129)
```

See also

BackgroundColorInvalid

Font

This property sets or returns the font for the content of the verification field.

Sample Code

The following sample code sets the font for the InvoiceNo field.

```
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").Font
= DefaultFieldFont
```

FontColor

This property sets or returns the font color for the content of the verification field.

Sample Code

The following sample code sets the font color for the InvoiceNo field to gray.

```
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)
```

See also

Font

Invisible

This property sets or returns if the field is visible or hidden from the Verifier or Web Verifier form. The developer uses script options to hide or display the field from the verifier user. In Web Verifier, you can use this property in the VerifierFormload event only.

Syntax

```
Invisible as Boolean
```

Sample Code

The following sample code hides the InvoiceNo field from the verifier user.

```
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
```

See also

VerifierFormLoad

Left

This property sets or returns the left position of the field on the Verifier form.

Sample Code

The following sample code returns the left position of the Invoice Number field from the Verifier form.

```
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
```

See also

- Top
- Width

Name

This read-only property returns the name of the field on the Verifier form.

Sample Code

The following sample code returns the name of the Invoice Number field from the Verifier form.

```
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
```

ReadOnly

This property sets or returns 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.

Syntax

```
ReadOnly as Boolean
```

Sample Code

The following sample code sets the Invoice Number field as read-only on the Verifier form.

```
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 ' Update the form UI
theVerificationForm.RepaintControls
```

TabIndex

This property sets or returns 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.

Sample Code

The following sample code sets the Invoice Number field tab sequence on the Verifier form.

```
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
```

Top

This property sets or returns the top position coordinates of the field on the Verifier form.

The scripter can 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.

Sample Code

The following sample code returns the Top position of the Invoice Number field from the Verifier form.

```
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
```

Type

This read-only property returns the field type information of the field on the Verifier form.

The scripter can choose to review information based on the field type.

Sample Code

The following sample code returns the field type information of the InvoiceNo field from the Verifier form.

```
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
```

See also

CdrVerifierFieldType

Width

This property sets or returns the Width size information of the field on the Verifier form.

The scripter can 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.

Sample Code

The following sample code returns the width information of the Invoice Number field from the Verifier form.

```
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
```

SCBCdrVerificationTable

This interface is used to identify verification properties specific for table validation elements.

SCBCdrVerificationTable Properties

The SCBCdrVerificationTable provides the following properties.

FontFont

This property sets or returns the font settings for the individual table field element.

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.

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.

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).

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.

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.

Syntax

```
HeaderBackgroundColor as OLE_COLOR
```

SCBCdrVerificationButton

Use this interface to set verification properties specific for all custom buttons defined on a verification form.

SCBCdrVerificationButton Properties

The SCBCdrVerificationButton object provides the following properties.

Font

This property sets or returns the font settings, such as name, type and style, for the individual custom button control.

Syntax

```
Font as StdFont
```

FontColor

This property sets or returns the font color for the individual custom button control.

Syntax

```
FontColor as OLE_COLOR
```

BackgroundColor

This property sets or returns background color for the individual custom button control.

Syntax

```
BackgroundColor as OLE_COLOR
```

SCBCdrVerificationLabel

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.

SCBCdrVerificationLabel Properties

The SCBCdrVerificationLabel provides the following properties.

BackgroundColor

This property sets or returns the color for the verification text label to display to the user. By default, the background color of the field is gray.

Syntax

```
BackgroundColor as OLE_COLOR
```

Sample Code

The following sample code sets the InvoiceNo label color to gray.

```
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").BackgroundColor = RGB (192, 129, 129)
```

Font

This property sets or returns the font for the content of the verification field label.

Note: To get the StdFont object, add OLE Automation as a reference.

Sample Code

The following sample code sets the font for Invoice Number field label.

```
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
```

FontColor

This property sets or returns the font color for the content of the verification field label.

Note: To get the OLE_COLOR object, add OLE Automation as a reference.

Sample Code

The following sample code sets the font color for the InvoiceNo field label to blue.

```
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)
```

Invisible

This property sets or returns if the field label is visible or hidden on the Verifier form. The developer can script options to hide or display the field label from the verifier user.

Syntax

```
Invisible as Boolean
```

Sample Code

The following sample code hides the Invoice Number field label from the verifier user.

```
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").Invisible = True ' Update the form
theVerificationForm.RepaintControls
```

Left

This property sets or returns the left position of the field on the Verifier form.

Sample Code

The following sample code returns the left position of the Invoice Number field label from Verifier Form.

```
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
```

Name

This read-only property provides the Name of the field label on the Verifier form.

Sample Code

The following sample code returns the name of the Invoice Number field label from the Verifier Form.

```
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
```

Text

This property sets or returns the text of the verification field label on the Verifier form.

Sample Code

The following sample code sets the Invoice Number field label text on the Verifier Form.

```
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"
```

Top

This property sets or returns the top position coordinates of the field label on the Verifier form.

The scripter can 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.

Sample Code

The following sample code returns the top position of the Invoice Number field label from the Verifier Form.

```
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
```

Width

This property sets or returns the Width size information of the field label on the Verifier form.

Sample Code

The following sample code returns the width of the Invoice Number field label from the Verifier form.

```
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
```

Printing Documents (SCBCroPrintLib)

The SCBCroPrintLib provides the following objects to configure the printing process.

SCBCroPrint

SCBCroPrintLib Type Definitions

The SCBCroPrint object provides the following type definitions.

CROImgFileTypes

This type definition specifies the image file type.

Type	Description
CROcBMP	Microsoft Windows Bitmap (.bmp) Uncompressed monochrome or 1- to 32-bit color binary bitmap **Multiple Images**: No
CROcBMP_ RLE	Microsoft Windows Bitmap (.bmp) RLE-compressed monochr. or 1- to 32-bit color binary bitmap **Multiple Images**: No
CROcBRK_G3	Brooktrout file (.brk) Group 3 (G3) compressed monochrome (1-bit) file **Image Type**: Bitmap **Data Encoding**: Binary **Multiple Images**: No
CROcBRK_ G3_2D	Brooktrout file (.brk), Group 3 (G3)
CROcCAL	CALS Raster (.CAL, .RAS, .CALS) Monochrome bitmap, to provide a standardized graphics interchange for electronic publishing, such as technical graphics, CAD/CAM, and image processing applications.
CROcCLP	Windows Clipboard (.CLP) RLE or uncompressed up to 24-bit color bitmap **Multiple Images**: No Used to hold any of several data types that can be stored by the Windows Clipboard

CROcDCX	DCX (.DCX), RLE Monochrome Color table (16 or 256 entries) 24-bit RGB Bitmap **Multiple Images**: Yes Allows to store multiple PCX files in one file.
CROcEPS	Encapsulated PostScript (.EPS), (.EPI) 1-bit monochrome MetafileStorage and interchange of single images (up to one page) across a wide range of platforms.
CROcICA_G3	Image Object Content Architecture .(ICA) Monochrome, 4, 8, 16-bit palette gray-scale and color, RGB, YCrCb, & YCbCr Bitmap, G3 compr. **Multiple Images**: No
CROcICA_G4	Image Object Content Architecture .(ICA) Monochrome, 4, 8, 16-bit palette gray-scale and color, RGB, YCrCb, & YCbCr Bitmap, G4 compr. **Multiple Images**: No
CROcICO	ICOn resource file(.ICO), Monochrome, 3, 4, 8, 24-bit palette Bitmap, **Multiple Images**: Yes Storage and interchange of iconic bitmaps (icons), across many Windows applications, regardless of the pixel resolution and color scheme of the display hardware.
CROcIFF	Electronic Art’s Interchange File Format (.iff) RGB palette up to 24-bit, binary Bitmap **Multiple Images**: No RLE or uncompressed
CROcIMT	IMNET graphics,
CROcJPG	JPEG File Interchange Format (.JPG) 8-bit gray scale or 24-bit YCrCb color 2D raster, JPEG compr. **Multiple Images**: No Interchange of .JPG files between applications on different platforms.
CROcLDF	Lura Document format

CROcMOD_G3	Mixed Object Document Content Architecture Monochrome, 4, 8, 16-bit palette gray-scale and color, RGB, YCrCb, & YCbCr Bitmap **Multiple Images**: Yes G3 compression Allows multiple IOCA images to be stored in one file.
CROcMOD_G4	Mixed Object Document Content Architecture Monochrome, 4, 8, 16-bit palette gray-scale and color, RGB, YCrCb, & YCbCr Bitmap **Multiple Images**: Yes G4 compression Allows to store multiple IOCA images in one file.
CROcNCR	NCR Image
CROcNCR_G4	NCR Image, compression G4
CROcPCT	Macintosh PICT (.PCT) Monochrome, 48-bit color palette, up to 24-bit RGB metafile **Multiple Images**: No Used to capture a picture as a set of Macintosh QuickDraw library function calls
CROcPCX	PC Paintbrush File Format (.PCX) Monochrome, 2 - 8-bit color table, 24-bit RGB Bitmap **Multiple Images**: No Storing images for ZSoft Corporation's paint program, such as PC Paintbrush
CROcPNG	Portable Network Graphics (.PNG) Truecolor up to 48 bpp, Grayscale up to 16 bpp, Palette up to 256 colors Bitmap / raster, **Multiple Images**: No Effective lossless compression of truecolor images'.
CROcPSD	Adobe Photoshop (.PSD), all colors including RGB, CMYK, multi-channel Bitmap Multiple Images: NoStoring bitmaps in the Adobe Photoshop graphical editing application.
CROcRAS	Sun Raster Data Format (.RAS), 1-bit monochrome, 8-bit gray scale or colormapped, 24 or 32-bit RGB colormap or Truecolor Bitmap **Multiple Images**: No Bitmap format for Sun UNIX platforms.

CROcSGI	SGI Image File Format (.SGI), RLE or umcompr. monochrome, gray-scale, and color including RGB, and RGB with alpha channel Bitmap **Multiple Images**: No Display of images from the SGI image library
CROcTGA	Truevision Targa (.TGA), RLE (usually uncomp.) 8, 16, 24-bit palette or RGB, Alpha data 1 or 8-bits Bitmap (2D raster)
CROcTIF	Tagged Image File Format (.TIF), Monochrome, 4 to 8-bit gray scale, 24-bit palette or RGB Bitmap **Multiple Images**: Yes Standardized data storage and interchange of images obtained from scanners and incorporated into desktop publishing
CROcTIF_G3	Tagged Image File Format (.TIF), Monochrome, 4 to 8-bit gray scale, 24-bit palette or RGB Bitmap **Multiple Images**: Yes G3 compr. Standardized data storage and interchange of images obtained from scanners and incorporated into desktop publishing
CROcTIF_G3_ 2D	Tagged Image File Format (.TIF), Monochrome, 4 to 8-bit gray scale, 24-bit palette or RGB Bitmap **Multiple Images**: Yes G3 compr., 2-dim. Standardized data storage and interchange of images obtained from scanners and incorporated into desktop publishing
CROcTIF_G4	Tagged Image File Format (.TIF), Monochrome, 4 to 8-bit gray scale, 24-bit palette or RGB Bitmap **Multiple Images**: Yes G4 compr. Standardized data storage and interchange of images obtained from scanners and incorporated into desktop publishing
CROcTIF_HUF	Tagged Image File Format (.TIF), Monochrome, 4 to 8-bit gray scale, 24-bit palette or RGB Bitmap **Multiple Images**: Yes Huffmann compr. Standardized data storage and interchange of images obtained from scanners and incorporated into desktop publishing
CROcTIF_JPG	Tagged Image File Format (.TIF), Monochrome, 4 to 8-bit gray scale, 24-bit palette or RGB Bitmap **Multiple Images**: Yes JPEG compr. Standardized data storage and interchange of images obtained from scanners and incorporated into desktop

CROcTIF_LZW	Tagged Image File Format (.TIF), Monochrome, 4 to 8-bit gray scale, 24-bit palette or RGB Bitmap **Multiple Images**: Yes LZW compr. Standardized data storage and interchange of images obtained from scanners and incorporated into desktop publishing
CROcTIF_ PACKED	Tagged Image File Format (.TIF), Monochrome, 4 to 8-bit gray scale, 24-bit palette or RGB Bitmap **Multiple Images**: Yes Packed standardized data storage and interchange of images obtained from scanners and incorporated into desktop publishing
CROcWMF	Microsoft Windows Metafile (.WMF), Monochrome, Color lookup table, RGB up to 24-bit Metafile (bitmap and vector) **Multiple Images**: No Uncompressed. Convenient storage and interchange for applications under MS Windows.
CROcXBM	X BitMap (.xbm), Monochrome Bitmap **Multiple Images**: Yes Uncompressed. Primarily for storing cursor and icon bitmaps for the X System graphical user interface.
CROcXPM	X PixMap (.XPM), monochrome, 2-bit and up gray scale, all colors Bitmap **Multiple Images**: Yes Uncompressed. To store X Window Pixmap information to disk
CROcXWD	X Window Dump (.xwd), monochrome, 2 - 15-bit palette, 16, 24 or 32 TrueColor Bitmap **Multiple Images**: No Uncompressed. Storing screen dumps from the X Window System

CroPrintPageRange

This type definition specifies the page range of the document to print.

Type	Description
ALL	All pages
PAGETOFROM	Page range (From - To)

CroPrintPageSelection

This type definition specifies the options of the document area to print.

Type	Description
ALLPAGES	Print all pages
RANGE	Print range of pages
SELECTION	Print only selection

Orientation

This type definition specifies the page orientation when printing

Type	Description
LANDSCAPE	Landscape orientation
PORTRAIT	Portrait orientation

ResizeImage

This type definition specifies the zoom options for the document to print.

Type	Description
ACTUALSIZE	Refreshes the pane
CUSTOM	Resize defined by user
FITTOHEIGHT	Resize document to fit to height of page
FITTOPAGE	Resize document to fit to width and height of page
FITTOWIDTH	Resize document to fit to width of page

Print Object SCBCroPrint

The component SCBCroPrint provides methods and properties to configure the printing process.

SCBCroPrint Properties

This object provides the following properties.

BottomMargin

This property sets or returns the margin in inches from the bottom of the physical page to the bottom of the printable area.

The default value is the minimum value allowed by the printer.

Syntax

```
BottomMargin As Double
```

Collate

This property sets or returns whether the printed document is collated.

Syntax

```
Collate As Boolean
```

Possible values

- True: The printed document is collated.
- False: The printed document is not collated.

Copies

This property sets or returns the number of copies to print.

The default value is 1.

Syntax

```
Copies As Long
```

DocumentName

This property sets or returns the name of the file to print.

Syntax

```
DocumentName As String
```

FromPage

This property sets or returns the start page number when a range of page is to be printed.

The default start page number is 1.

Syntax

```
FromPage As Long
```

LeftMargin

This property sets or returns the margin in inches from the left margin of the physical page to the left margin of the printable area.

The default value is the minimum value allowed by the printer.

Syntax

```
LeftMargin As Double
```

PageRange

This property sets or returns the page range of the document to be printed.

The default value is set to `PageToFrom`.

Syntax

```
PageRange As CroPrintPageRange
```

See also

CroPrintPageRange

PaperOrientation

This property sets or returns the page orientation for printing.

The default value is `Portrait`.

Syntax

```
PaperOrientation As Orientation
```

See also

Orientation

Printer

This read-only property returns the printer name.

Syntax

```
Printer As String
```

PrinterPort

This read-only property returns the printer port.

Syntax

```
PrinterPort As String
```

RightMargin

This property sets or returns the margin in inches from the right margin of the physical page to the right margin of the printable area.

The default value is the minimum value allowed by the printer.

Syntax

```
RightMargin As Double
```

ShowPrintDialog

This property sets or returns whether the **Print** dialog is displayed before printing.

Use this property to switch between **Interactive** and **Non-Interactive** mode.

The default value is True, that is, the dialog is displayed.

Note: In **Non-Interactive** mode, the printer used is the default printer connected to the user's computer and all default properties are used. You cannot set the properties programmatically as in **Interactive** mode.

Syntax

```
ShowPrintDialog As Boolean
```

Example

The following sample code shows how to set the **non-interactive** mode by choosing not to display the **Print Dialog** box.

```
'Hide the Print dialog. PrintObject.ShowPrintDialog = false ' After hiding
the print dialog box, the printing of a CroImage or CroCIDOC object (with
or without ' annotations) to the printer or an image (Single page or
Multi-page) can be done using the same code as in Interactive mode.
```

ShowPrintPageNrDialog

This property sets or returns whether the **Printing Page** dialog is displayed while the print job is in progress.

The default value is `True`, that is, the dialog is displayed.

Syntax

```
ShowPrintPageNrDialog As Boolean
```

ToPage

This property sets or returns the start page number when a range of page is to be printed.

The default value is 1.

Syntax

```
ToPage As Long
```

TopMargin

This property sets or returns the margin in inches from the top of the physical page to the top of the printable area.

The default value is the minimum value allowed by the printer.

Syntax

```
TopMargin As Double
```

XResolution

This property sets or returns the horizontal resolution of the printer or image in dots per inch.

The default value is 300 DPI.

Syntax

```
XResolution As Long
```

YResolution

This property sets or returns the vertical resolution of the printer or image in dots per inch.

The default value is 300 DPI.

Syntax

```
YResolution As Long
```

SCBCroPrint Methods

This object provides the following methods.

PrintDocument

This method prints the document on either the default printer or on the printer selected by the user in the **Print** dialog.

Syntax

```
PrintDocument (itemToPrint As Object, howToResize As RESIZEIMAGE,
zoomValue As Long)
```

Parameter	Description
itemToPrint	Reference to the object to print.
howToResize	Specifies how to zoom the image when printing.
zoomValue	The zoom value. This parameter is only taken into account when the `howToResize` parameter is set to `CUSTOM`.

See also

ShowPrintDialog

ResizeImage

Examples

The following sample code shows how to print a **CroImage** object without annotations,

The resize parameter is set to `FitToWidth`, so that the image uses the full width of the paper.

```
SrcImage.LoadFile CommonDialog1.FileName PrintObject.PrintDocument
SrcImage, FitToWidth, 0
```

The following code shows how to print a **CroCIDOC** object without annotations

The resize parameter is set to `FitToHeight`, so that the image uses the full height of the paper.

```
SrcCIDoc.LoadFile CommonDialog1.FileName PrintObject.PrintDocument
SrcCIDoc, FitToHeight, 0
```

PrintMultiPageImage

This method prints the document in multi-page format to an image file instead of to the physical printer.

Syntax

```
PrintMultiPageImage (ItemToPrint As Object, FileName As String, FileType
As CROImgFileTypes)
```

Parameter	Description
ItemToPrint	Reference to the item to print as image.
FileName	Path of the output image.
FileType	Type of the output file. For a list of supported file types, see CROImgFileTypes.

Examples

The following sample code shows how to print a **CroImage** object without annotations to a multi-page TIFF image with **LZW compression**, which is applicable for all bit-count images, such as 1, 4, 8, and 24.

The resolution of the output image is specified by the `Resolution` property.

```
SrcImage.LoadFile CommonDialog1.FileName PrintObject.PrintMultiPageImage
SrcImage, CommonDialog2.FileName, CROcTIF_LZW
```

PrintSinglePageImage

This method prints the document as a single page image to an image file instead of to the physical printer.

Syntax

```
PrintSinglePageImage (ItemToPrint As Object, Path As String, BaseNumber As
Long, FileType As CROImgFileTypes)
```

Parameter	Description
ItemToPrint	Reference to the item to print as image.
Path	Path of the output image.

BaseNumber	Specifies the output file name, so that appending `BaseNumber` to the `Path` parameter forms the full file name.
FileType	Type of the output file. For a list of supported file types, see CROImgFileTypes.

Example

The following code sample shows how to print a CroImage object without annotations to a single-page BMP image without compression. The resolution of the output image is specified by the `Resolution` property.

```
SrcImage.LoadFile CommonDialog1.FileName PrintObject.PrintSinglePageImage
SrcImage, "C:\temp", 100, CROcBMP
```

The following code sample shows how to print a **SrcCIDoc** object without annotations.

```
SrcCIDoc.LoadFile CommonDialog1.FileName PrintObject.PrintSinglePageImage
SrcCIDoc, "C:\temp", 100, CROcBMP
```

ShowPageSetupDialog

This method shows the **Page** setup dialog box so that the user can configure properties such as paper size, paper source, page orientation, and the margins of the page.

Syntax

```
ShowPageSetupDialog ()
```

ShowPrintDialogBox

This method displays the **Print Dialog** before printing.

Syntax

```
ShowPrintDialogBox (PageSelectionEnabled As CroPrintPageSelection,
ActualPageSelection As CroPrintPageSelection, StartPage As Long, EndPage
As Long, Copies As Long)
```

Parameter	Description
PageSelectionEnabled	**Possible values** See CroPrintPageSelection
ActualPageSelection	[in,out] Actual page **Possible values** See CroPrintPageSelection
StartPage	[in,out] Number of the first page to be printed

EndPage	[in,out] Number of the last page to be printed
Copies	[in,out] Number of copies to be printed

Error Handling

On Error

The **On Error** statement enables you to handle runtime errors.

Without an **On Error** statement, every runtime error leads to an error message, and the execution terminates.

Notes

- An error handler is enabled by an **On Error** statement.
- An *active* error handler is an enabled handler that is in the process of handling an error.
- If an error occurs while an error handler is active, that is, between the occurrence of the error and a **Resume**, **Exit Sub**, **Exit Function**, or **Exit Property** statement, the error handler of the current procedure cannot handle the error.

  The control returns to the calling procedure. If the calling procedure has an enabled error handler, it is activated to handle the error.

  If the error handler of the calling procedure is also active, control passes back through the previous calling procedures until an enabled but inactive error handler is found.

  If no inactive, enabled error handler is found, the error is fatal at the point where it actually occurred.
- Each time the error handler returns control to a calling procedure, that procedure becomes the current procedure.
- When an error is handled by an error handler in any procedure, execution continues in the current procedure at the location specified by the **Resume** statement.
- An error-handling routine is not a subprocedure or function procedure. It is a section of code identified by a line label or line number.

On Error GoTo *[line]*

If an error occurs, the processing continues at the specified label or line number and activates the error handler

On Error Resume Next

If an error occurs, the processing continues with the statement immediately following the statement that caused the runtime error, or with the statement immediately following the last call of the procedure containing the `On Error Resume Next` statement.

This allows the processing to continue despite a runtime error. You can then set up the error handling routine within the procedure.

On Error GoTo *0*

This statement disables any enabled error handler in the current procedure.

ErrObject

The ErrObject module provides methods and properties to identify and handle runtime errors using the **Err** object.

You can use them at any place in your code.

The **ErrObject** contains information about runtime errors and accepts the Raise and Clear methods for generating and clearing runtime errors.

Err.Clear

Clears all property settings of the **Err** object.

Err.Description

Sets or returns the string expression containing a descriptive string associated with an object.

Err.HelpContext

Sets or returns a context ID for a topic in a Help file.

Err.HelpFile

Sets or returns a fully qualified path to a Help file.

Err.LastDLLError

Sets or returns a system error code from a call to a DLL.

Err.Number

This is the default property that sets or returns a numeric value that represents an error.

An Automation object can use this property to return an SCODE.

When a runtime error occurs, the properties of the **Err** object contain information that uniquely identify the error and can be used to handle it.

Err.Raise

Generates a user-defined runtime error.

Err.Source

Sets or returns the name of the object that caused the error.

Sample Code

The following sample code sample shows how to use **On Error Goto** and the **ErrObject**.

```
Private Sub mnTest_Click() On Error GoTo lblErr '. . . Exit Sub lblErr:
MsgBox Err.Description End Sub
```

Troubleshoot Scripting Issues

To provide script dumps for script issues, such as compilation problems that occur in Web Verifier, complete the following steps. This feature requires advanced product knowledge.

Note: If you enable this feature, encrypted script pages are not exported out of the project.

1. In **Windows** registry, complete one of the following substeps.
    1. For a 32-bit machine, navigate to `HKEY_LOCAL_MACHINE\SOFTWARE\Oracle\Cedar`.
    2. For a 64-bit machine, navigate to `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Oracle\Cedar`.
2. In the right pane, right-click and then click **New > DWORD (32-bit) Value**.
3. In the **Name** field, type `DumpProjectScriptCode` and then click **OK**.
4. Right-click the **DumpProjectScriptCode** key and click **Modify**.
5. 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` (zero).
    - To enable the feature, type `3`.

Encryption

Password Encryption for Database Connection Strings

The WebCenter Forms Recognition architecture makes it important to be able to hide sensitive security information, such as database passwords stored in WebCenter Forms Recognition or custom project configuration files.

This requirement also applies to the database connection strings in the WebCenter Forms Recognition project INI files that often contain multiple connection strings to different database instances, such as for Visibility reporting or custom databases with unencrypted passwords. These INI files may not reside directly on the local Verifier workstation, but could be easily accessed by the Verifier users, because at least read-only access to the WebCenter Forms Recognition project directory is required.

WebCenter Forms Recognition allows password encryption in an INI or CONFIG file by using RSA-

1024 or RSA-3072 encryption. The default and recommended encryption method is RSA-3072.

Notes:

- To decrypt a password encrypted with an RSA-1024 public key, use the associated RSA-1024 private key.
- To decrypt a password encrypted with an RSA-3072 public key, use the associated RSA-3072 private key.

Password length

The maximum character length for a password to encrypt using **RSA-1024** is 30.

The maximum character length for a password to encrypt using **RSA-3072** is 280.

Longer passwords do not encrypt correctly.

Encrypt a password

Password encryption for use in custom INI and CONFIG files is optional, but highly recommended. WebCenter Forms Recognition supports encryption using RSA encryption keys of length 1024 or 3072. To encrypt a password, complete the following steps.

1. Contact the Oracle Technical Support group and request a pair of RSA encryption keys.

Note: Keep your private key safe and do not share it with anyone else.

2. In the ***C:\Program Files (x86)\Oracle\WebCenter Forms Recognition*** directory, create a new batch file and give it a meaningful name, such as **CreateEncryptedPassword.bat**.
3. To encrypt the password, copy the following line to the batch file, replacing `MyPassword` with the password you want to encrypt and `[Custom Public Key]` with your public key.

   Example

   ```
   DstCrypt.exe /text "MyPassword" /key "<RSAKeyValue><Modulus>[Custom
   Public Key]</Modulus><Exponent>AQAB</Exponent></RSAKeyValue>" >>
   EncryptedPW.txt
   ```

   Notes:

   - The maximum character length for a password to encrypt using **RSA-1024** is 30.
   - The maximum character length for a password to encrypt using **RSA-3072** is 280.

4. Save and close the file.
5. In **Windows Explorer**, double-click the batch file.
6. From ***C:\Program Files (x86)\Oracle\WebCenter Forms Recognition***, open **EncryptedPassword.txt** in a text editor. The encrypted password can be safely included in custom INI and CONFIG files.

Decrypt a password

Encrypted passwords stored in custom INI and CONFIG files can later be decrypted in script. To decrypt an encrypted password, complete the following steps.

1. In **Designer** open the project.
2. Switch to **Definition Mode**.
3. In the left pane, select the class where you want to implement connection string encryption.
4. On the toolbar, click **Show/hide script** .
5. In the **Script View** dialog box, click **Edit > References**.
6. In the **References** dialog box, select **Cedar Crypt Library (5.80)** and then click **OK**.
7. To decrypt the encrypted password and adapt the connection string read from the *[Project]*.ini file, modify your script according to the following example, replacing `[Custom Private Key]` with the private key corresponding to the public key used when encrypting the password.

   Example

   ```
   Dim theCedarCryptographyHelper as New CdrCrypt.RSACodecInt Dim
   strEncryptedPassword as String Dim strOpenPassword as String Dim
   strPrivateKey as String strPrivateKey = "<RSAKeyValue><Modulus>[Custom
   Private Key]</D></RSAKeyValue>" strEncryptedPassword = DicVal("01" &
   "ConnectionPassword", "SQL") If Len(strEncryptedPassword) > 0 Then
   strOpenPassword = theCedarCryptographyHelper.Decode
   (strEncryptedPassword, strPrivateKey) End If Len(strOpenPassword) > 0
   Then strConnection = strConnection + ";Password=" + strOpenPassword End
   If
   ```

8. To protect the private key, encrypt the script page that contains the code above through the standard

script code encryption feature.

9. When you release your project, distribute the public key along with the project.

   Note: Professional Services representatives will use this public key to encrypt the passwords when installing the project.

Encryption keys for testing purposes

You can use one of the following encryption key pairs for testing purposes. Request and use a new pair before distributing the project.

RSA-1024

Test Public Key

```
<RSAKeyValue><Modulus>vJ+W7SuXuvOrWVoy4tPrbfLCuoHElo750cpTuEzLPk6iz6bHAodP
VgLFaOEK+XMMS2G5z+6961vuQsDGUt+O1Ag1PiTXCa6rrAaeCaaDO4HI8Mmpw0OkUZEfCZpTTY
CYQPfZlgokwomF6VDSB9dlUS430IT0gctQY1b5iM4MqT0=</Modulus><Exponent>AQAB</Exponent>
</RSAKeyValue>
```

Test Private Key

```
<RSAKeyValue><Modulus>vJ+W7SuXuvOrWVoy4tPrbfLCuoHElo750cpTuEzLPk6iz6bHAodP
VgLFaOEK+XMMS2G5z+6961vuQsDGUt+O1Ag1PiTXCa6rrAaeCaaDO4HI8Mmpw0OkUZEfCZpTTY
CYQPfZlgokwomF6VDSB9dlUS430IT0gctQY1b5iM4MqT0=</Modulus><Exponent>AQAB</Ex
ponent><P>8SRHEvT5Bn2paRHSDR9yCQb7WGYE9PbeHzuqwH6iWa0LNYJrSrhhUeCEpwlPLQWQ
q10KmMZgG0+Br4nuBMmMHQ==</P><Q>yD7l9fjB/MJWYaV3LcEzY286Q+Xvo74i6THvHkKqB1N
KYGcN9xF9d8XbiUQNgBZ/4F02T6mFeYDO32KFVRXHoQ==</Q><DP>nRDTFn7nwRmSgfRwi8min
kyk5DQ3IFO35EIZ+x3Ao4Z52ZWkStwDz6/c12vR3XJVg7irkU0NBlzoDK1bklSw5Q==</DP><D
Q>B3xieGmORva05/2ZkPpSA3ubAALOjJ6FC5a0S7tOQ+vXMfdoTD45JIsfA+ipYIp2yVpyt1Ot
C7fHBA7Y0S95QQ==</DQ><InverseQ>4S1xqlXK9f1rawGCbFWOVp6lz1fCoQ8RfyDE87/G/pU
ilHRJV2acBAcngY3c/MRMKrXQb8lx99k7dENUYc8ywQ==</InverseQ><D>KAL6cwkCQKgbuvK
FRNSLZmFOqV2JpB5kI/p1U+0GWAs6Qi4wnPqy+53O3naOa2faPctXLSKJqvlvSz21VDMUCsyph
vOSxBtc1cZHJp4ueQPA7u+qrIJaDY1RhlAVoqNfCJFX6+McVJ+I/X+mZOCtdUaCuAoNn014UYO
aMujYDQE=</D></RSAKeyValue>
```

RSA-3072

Test Public Key

```
<RSAKeyValue><Modulus>u67vJnR9O6JNO4r8rE7V2DVYZMcxe1w6k2rGdeLEpVBw1lLhoA09
qJvvMlfz4uod89XIBMRbrT6nPJmx1n7noTTkypUK/q2DoQCOW5Ls/hvlo4G+rJPtVDKpWHM9sO
ypV9YVg+DvKbPMgOpPS6IM9dHoZNt6tyatB0pyjiJj4GM4ooNQSl9vfbNIy2/87OCAHB2y76c5
HvSI4AFADJcmv/oF1ZAwxteuaWpkbwAgxfPUUwneNv1mTX2h0MgVFHO/HDwHzU0aKqzYQCouNB
yvUt/TirSwlSgDPWFnZg0dRTrDofJl0QrSC8hzbf8KYCyHYdG+0iUZriFYSAgt8iMuxl8xei95
KylvZFKniryfbL/4KRkcVgx63a/1lnDXfwfg7kk6xFN5W0AWYz8U1CHJNL+GIFSHTzqnr1/30z
TFbrzR7mWZFRuB0duZoaJ8D1Tc6x7f6E0YmkCpNuDMb9fB77dfMVg6wRI34YgogjUv0ke1W6RZ
31XHGVNHeYyd7ma5</Modulus><Exponent>AQAB</Exponent></RSAKeyValue>
```

Test Private Key

```
<RSAKeyValue><Modulus>u67vJnR9O6JNO4r8rE7V2DVYZMcxe1w6k2rGdeLEpVBw1lLhoA09
qJvvMlfz4uod89XIBMRbrT6nPJmx1n7noTTkypUK/q2DoQCOW5Ls/hvlo4G+rJPtVDKpWHM9sO
```

```
ypV9YVg+DvKbPMgOpPS6IM9dHoZNt6tyatB0pyjiJj4GM4ooNQSl9vfbNIy2/87OCAHB2y76c5
HvSI4AFADJcmv/oF1ZAwxteuaWpkbwAgxfPUUwneNv1mTX2h0MgVFHO/HDwHzU0aKqzYQCouNB
yvUt/TirSwlSgDPWFnZg0dRTrDofJl0QrSC8hzbf8KYCyHYdG+0iUZriFYSAgt8iMuxl8xei95
KylvZFKniryfbL/4KRkcVgx63a/1lnDXfwfg7kk6xFN5W0AWYz8U1CHJNL+GIFSHTzqnr1/30z
TFbrzR7mWZFRuB0duZoaJ8D1Tc6x7f6E0YmkCpNuDMb9fB77dfMVg6wRI34YgogjUv0ke1W6RZ
31XHGVNHeYyd7ma5</Modulus><Exponent>AQAB</Exponent><P>xwc0XsNUlFu86iyz1sUd
BWFLLaDbiugbzrTGxLKznQ8Sy4UhB+q7AifLuSIft7lRfdxbXx3OZdsGI4l/9lij4CcUhizRwg
6O2j0b4r3S5lGo/JpmNEHygAneotKFE9Ym/BkrYajR0QFXSFlibO6xsuhDz/wUo5ccVYLVuA8X
EJmL4IAh5e13aG+sstC4x3n2Sue5tWawRo0vhRhiXzgsgPkXXtCJgnWrDlWNCXG400RfbYMyBi
d9KOn/mMh0x/cj</P><Q>8Whhp4U1To/3y6soJqnhYEgx8U0XINFpsn9T1t3Zg2VSwfo6oYf64
IdXFeK1XA5le+hTuW0MCQ09uSIadYdtfBWuidK8af7lDg2li8yozuZcS884Z9REJc1mVjDZh4I
9YXvvCzYF2iC8MnP28ESr78KEFtc9unuxTAywcX+CXt2ptQiPqTXV4RYi99Y1SWn6bf0Kn0knM
McdhpC79/lZN/nn7TrbSZ/fW0hh1XA8CLE7al3BAFFDkk2o11QqMzZz</Q><DP>XLW+b2YNwA3
7shmgrCqlmhv4KQkFO29jhKYw8O9NO5wJ5UPeXxG4iwh+nX6Hx81LaJV8+pGn00pzUZp7QQWJU
KDjrpyyGPMvF/nfQ3+o/iCL1x8U/MK7c2kljDMESnk3L7eJvbQmCHXXcpwwfQ2zOON4syl5c7q
wOCT56QMee10LC0ikgbZcKQncKG+Ro1nsoHfQye9LrZGW0SjKqhBAqtNdT41UgLDPPZ4cVInkz
oTK7vhuFO/QHX7i0QiPXZaB</DP><DQ>ePedN5yAgcWSnfQ+QmGOFfkXTZsbo85L8K3tDRZOEJ
GNTrZc+uRwymIHOWhJCMAOCnD4w0npZf/EliiIqawbKLqUo2t+Dn8zuD8OXWdIj6TPSw2jbiPn
oimyxi3kUwroRxkbR7riY9NTetSxQOSoY5RNLETfIbsptLoHEFics3zPtW98zYW0jv3Qt/AW85
qvar4JvJejk8Wy/eM4zaGdG3eWyEFd8R6zhcJXg+ShsJle9IXWWJpgCcfwezYpj1AR</DQ><In
verseQ>YvqVh4VWJccDc40MExy3BQgTQUiw2xBmPaSoqzrSDrmiwo+84JqW4ljOkFDdeSrjVIS
VLC4EssvH7FD+p/IqDV3Rmj+HH0Gg7UB9CTiPdKqGzVFkrIl3cZVysWfOBtDWB7bKtKxqFx/4y
3UUWxF1wLP6c3Cz4adW18Bz5VTIkLOc9h+jYV7PKoEtlANlg3Jy5nV954P8l+dz6lmDLiGW/7P
9IBl2W7WVYleKbvyNb0u3kFF4tcJFmiIpWRENdo6p</InverseQ><D>glGrmSYVtwyxWADgBb0
P23rfD8sIXsHhom7nsv9sz+UVFJykMgA/qg3sfyZ3ID6z4ZW1tympBpiUgw2tYGTp7RPPMaNf6
QsOdQLv1kIk8STpxSjX/vkQOACded87HNiXlXf/pHzRVEGWZIVN3hR9r0q7dRy5Dud6Ca/17x+
N/kanEqDfZWTzA3Mzvp7aHpicG+m7T5DrpsThLblwgc2KiI7Q8kOWutsULfpEFpceKpVt/uLBO
6mkGXZ8n33utS7QeLWC41AV5t0DEo6qq3DbgzrRszBiue2QvNMvmQH5DK/4r3ZZvLhwBVGgm2L
dE1iFJVq8aoHiHMJDnXDAtt9vUzzpMi5nnBAQBUVu0HE4P73UNZNlV3TaVXTO3eEfqek2Mzv/o
PpJMwMDvecK5kASzXI9y6wd2L11OVZdNhE+R5hMQZ/R20JHJ8t521c8d0ptl0qlz0/H5Ln+Hbw
h1NfIw7aDZ7J8dgnlYXhv4SpvrI3B/2/0v+xo+XtwaJstijFJ</D></RSAKeyValue>
```

Script Encryption

For information on script encryption, see "Script encryption" in the *Oracle WebCenter Forms Recognition Designer User's Guide*.