MedicSDK

Medical SDK Library Description

Dec 2013

TABLE OF CONTENTS

Introduction

MedicSDK.ocx is an ActiveX component that encapsulates all the functionalities needed to scan, process, parse and export medical insurance documents. MedicSDK.ocx exposes two independent interfaces to the application:

User Interface: The scanned document is processed and displayed on the host application GUI while each word is highlighted and can be dragged using the mouse and dropped into the proper fields on the host application.

Data Interface: The text on the insurance card is automatically parsed and assigned to pre-defined fields. The host application can access these property fields and copy the proper fields on the host application.

MedicSDK is fully automatic and can analyze and extract all the relevant data fields from the medical insurance card including:

Member Name
Member ID
Plan Provider
Group Number
Payer ID
Copay OV
Copay SP
Copay UC
Copay ER
Effective Date
Expiry Date
Date of Birth
Tel
Email
Web
Address

Scanning and Processing Medical Insurance Documents

Scanning medical insurance documents and analyzing its content is done in the following automated sequence:

- **Detect document placement on the scanner tray:** MedicSDK automatically monitors the scanner paper sensor and sends the event to the application once a document is placed on the scanner's tray.
- **Scan the document:** MedicSDK automatically starts the document scan and sends the event to the application once the scan ends. When a duplex scanner is used, both document sides may be scanned in a single document pass.
- **Process the document:** MedicSDK automatically analyzes the scanned document as soon as the scan ends. If a duplex image is acquired, the processing algorithm processes both sides simultaneously. The result data is loaded with MedicSDK properties.
- Retrieve the analyzed data from the ActiveX properties.

MedicSDK is defaulted to perform steps 1-4 automatically without the application intervention. However, the application can also customize MedicSDK to trigger each step by the application.

Functions, Events and Properties

MedicSDK.ocx communicates with the hosting application using functions, events and properties.

Function Name	Operation
InitSdk	Initialize the SDK with the license key
ScanToFile	Scan the document
ScanToFileEx	Scan the document while displaying progress bar
ProcessMedical	Extract the information from the medical insurance document
ShowSide	Display side A or side B of the document (in *User Interface* mode)
ScanSize	Set the scan area width and height
CalibrateScanner	Calibrate the scanner
Clean	Clean the scanner
ReformatImage	Convert the image properties

Event Name	Indication

ButtonPressed	Indicates user pressed on the scanner's button
PaperInTray	Indicates document placement on the scanner's tray
ScanIsDone	Indicates document scan has ended
SetCurrentText	Indicates user changed the selected text (in *User Interface* mode)

Control Properties	Usage
Zoom	Set image zoom factor in window (in *User Interface* mode)
PaperInTrayAutoStartScan	Automate scan upon document placement on scanner's tray
NotifyOnPaperInTray	Enable\Disable firing paper-in-tray event
NotifyOnButtonPress	Enable\Disable firing button press event
PropDuplexScan	Set scanner to scan in simplex\duplex mode (for ScanShell® 800DX\3000 models only)
PropIsNeedCalibration	Short
PropIsScannerValid	Short
PropResolution	Short
PropScanWidth	Short
PropScanHeight	Short
PropScannerColorScheme	Short
PropScannerType	Short

Data Properties	Type
PropFrontSide	BOOL
PropPlanProvider	string
PropMemberName	string
PropMemberID	string
PropGroupNumber	string
PropPayerID	string
PropCopayOV	string
PropCopaySP	string
PropCopayUC	string
PropCopayER	string
PropEffectiveDate	string
PropExpireDate	string
PropDateOfBirth	string
PropOther	string
PropTelTotalItems	long
PropTelLabel	string
PropTelValue	string
PropEmailTotalItems	long
PropEmailLabel	string
PropEmailValue	string
PropWebTotalItems	long

PropWebLabel	string
PropWebValue	string
PropAddressTotalItems	long
PropAddressFull	string
PropAddressStreet	string
PropAddressCity	string
PropAddressState	string
PropAddressZip	string
propRawText	string
propContractCode	string
propPlanType	string
propDeductible	string
propRxBin	string
propRxPCN	string
propRxGroup	string
propEmployer	string
propCoverage	string
propPlanCodeTotalItems	long
propPlanCode	string
propRxId	string
propPlanAdmin	string
propGroupName	string
propIssuerNumber	string
propNameFirst	string
propNameMiddle	string
propNameLast	string
propNamePrefix	string
propNameSuffix	string
PropDeductibleTotalItems	long
PropDeductibleLabel	string
PropDeductibleValue	string

Functions

InitSdk

Format

```
InitSdk (License As String) As Long
```

Parameters

[in] **License** – Null terminated string that holds license key value.

Return

LICENSE_VALID: License is valid and the library is ready to be used.
LICENSE_INVALID: The license is invalid. All scanner operations are disabled.
LICENSE_EXPIRED: License has expired. All scanner operations are disabled.
LICENSE_DOES_NOT_MATCH_LIBRARY: The license is invalid for this library. All library operations are disabled.
GENERAL_ERR_PLUG_NOT_FOUND: This error returns if no valid scanner is attached to the PC.
SLIB_LIBRARY_ALREADY_INITIALIZED: The *InitSdk* function call is ignored since the library is already loaded.

Remarks

Use this function to initialize the MedicSDK ActiveX. This function must be called before calling any other function in the library.

ScanToFile

Format

```
ScanToFile (FileName As String) As Long
```

Parameters

[in] **FileName** – Null terminated string that holds the full path of the scanned image.

Return

If function succeeds, the return value is **SLIB_ERR_NONE**
If function fails, the return number may be one of the following:
SLIB_ERR_SCANNER_BUSSY - The scanner is still busy executing the previous scanner command.
LICENSE_INVALID – Library was not initialized with proper license.
SLIB_ERR_SCANNER_NOT_FOUND – No attached scanner was found.

SLIB_ERR_SCANNER_GENERAL_FAIL
SLIB_ERR_SCANNER_NOT_FOUND
SLIB_ERR_HARDWARE_ERROR
SLIB_ERR_PAPER_FED_ERROR
SLIB_ERR_SCANABORT
SLIB_ERR_NO_PAPER
SLIB_ERR_PAPER_JAM
SLIB_ERR_FILE_IO_ERROR
SLIB_ERR_PRINTER_PORT_USED
SLIB_ERR_OUT_OF_MEMORY

Remarks

Scan document to the internal image buffer and, at the same time, export it to a bitmap file named “File Name” in the local disk. The operation result can be tested for good completion by reading the *LastErrorStatus* property.
Note that it is important to scan the image in True color and 300 dpi for OCR recognition.

ScanToFileEx

Format

ScanToFile (**FileName** As String) As Long

Parameters

[in] **FileName** – Null terminated string that holds the full path of the scanned image.

Remarks

This function displays the scanning progress bar while scanning the document. Beyond this feature, this function is identical to the function *ScanToFile.*

ProcessMedical

Format

ProcessMedical (**ImageFileSideA** As String, **ImageFileSideB** As String) As Long

Parameters

[in] **ImageFileSideA** – Null terminated empty string – reserved.
[in] **ImageFileSideB** – Null terminated empty string – reserved.

Remarks

This function processes the medical insurance image and extracts the different fields in the image. The processed image is taken from the internal image buffer in the memory that was loaded in the last scan. If a duplex scanner is used to scan both sides of the insurance card in a single pass, both images (front and back) are saved into two internal image buffers in the memory. These images are processed in a single function call and the information from both sides is extracted and loaded to the SDK properties.
The two parameters of the functions are ignored and should be empty, null terminated strings.

Return

If function succeeds, the return value is equal or larger than 0 and contains the number of times that the image was rotated by 90 degrees until it was aligned properly in the memory:
0: The image of side A was not rotated.
1: The image of side A was rotated by 90 degrees by the function.
2: The image of side A was rotated by 180 degrees by the function.
3: The image of side A was rotated by 270 degrees by the function.

If the function fails, the return value is smaller than 0 and may be one of the following values:
INVALID_INTERNAL_IMAGE – No internal image is loaded. This value returns when attempting to use this function without scanning an image first.
GENERAL_ERR_PLUG_NOT_FOUND: This error returns if the image was not scanned by a CSSN scanner model.

ProcessMedicalSide

Format

ProcessMedicalSide (**Side** As Integer, **ImageFile** As String, **Reserved** As Integer) As Long

Parameters

[in] **Side** – 0 for front side image, 1 for back side image.
[in] **ImageFile** – Null terminated empty string – reserved.
[in] **Reserved** - 0.

Remarks

This function is similar to the function ProcessMedical but can process one side and not two sides at once to make the process time shorter.

Return

If function succeeds, the return value is equal or larger than 0 and contains the number of times that the image was rotated by 90 degrees until it was aligned properly in the memory:

0: The image was not rotated.
1: The image was rotated by 90 degrees by the function.
2: The image was rotated by 180 degrees by the function.
3: The image was rotated by 270 degrees by the function.

If the function fails, the return value is smaller than 0 and may be one of the following values:
INVALID_INTERNAL_IMAGE – No internal image is loaded. This value returns when attempting to use this function without scanning an image first.
GENERAL_ERR_PLUG_NOT_FOUND: This error returns if the image was not scanned by a CSSN scanner model.

ResetFields

Format

```
ResetFields ()
```

Parameters

None

Remarks

Use this function to clear data from all the fields in the medical object.

ScanSize

Format

```
ScanSize (width As long, Height As long) As long
```

Parameters

[in] **Width** – The document width in milli-inches.
[in] **Height** – The document width in milli-inches.

Remarks

This function sets the scan width and height in units of milli inches. For example, to scan a document size of 1" x 2.5" you need to call this function with the values 100 and 250 respectively.
If a pass-through scanner is used, the scan size may also be set to automatically detect the document size during the scan. This is done by setting both scan width and the scan height to ‘-1’. Pass-through scanner models are ScanShell® 800X\2000X\3000.

CalibrateScanner

Format

```
CalibrateScanner ()
```

Return value

Void.

Remarks

This function calibrates the scanner using the calibration card. The calibration results are stored in a file inside the windows directory. The operation result can be tested for good completion by reading *LastErrorStatus* property. This property may store one of the following values:
SLIB_ERR_SCANNER_BUSSY - The scanner is still busy executing the previous scanner command.
LICENSE_INVALID – Library was not initialized with proper license.
SLIB_ERR_SCANNER_NOT_FOUND – No attached scanner was found.
SLIB_ERR_INVALID_SCANNER – The attached scanner is invalid.
SLIB_FALSE – The operation failed (Mostly because no calibration card was found)
SLIB_TRUE – Operation succeeded.

Clean

Format

```
Clean ()
```

Return value

SLIB_ERR_SCANNER_BUSSY - The scanner is still busy executing the previous scanner command.

Remarks

This function cleans the scanner lens by running the cleaning pad (supplied in the scanner kit) back and forth. This function applies only to scanner models ScanShell® 800X\2000X.

ReformatImage

Format

```
ReformatImage ( _
    SourceImage As String, _
    toColor As Integer, _
    toDpi As Integer, _
    DestImage As String _
    )
```

Parameters

[in] **SourceImage** – Full path name of the original image file. If this string is empty the rotation is performed on the internal image.

[in] **toColor** – One of five values:
LICENSE_INVALID – Library was not initialized with proper license.
IMAGE_SAME_COLOR – No modification in the image color scheme.
IMAGE_BW – Convert to black and white color scheme.
IMAGE_GRAY_256 – Convert to 256 gray scale color scheme.
IMAGE_COLOR_256 – Convert to 256-color scheme.
IMAGE_COLOR_TRUE – Convert to true color scheme.

[in] **toDpi** – Set the new Image DPI. A value of 0 indicates no DPI modification.

[in] **DestImage** – Full path name of the destination file. If this value is an empty string no save will be performed.

Return

If function succeeds, it returns the value **IMG_ERR_SUCCESS.**
If the function fails it returns one of the following values:
IMG_ERR_BAD_COLOR – Bad **toColor** parameter value.
IMG_ERR_BAD_DPI – Bad **toDpi** parameter value.
IMG_ERR_FILE_OPEN – Cannot open input file. This value is returned if the **SourceImage** string is not empty but it points to a missing or invalid source image file.
INVALID_INTERNAL_IMAGE – This value is returned if the **SourceImage** string is empty but no document was scanned so there is no internal image in the memory.
IMG_ERR_FILE_SAVE_TO_FILE – Cannot save destination file.
IMG_ERR_FILE_SAVE_TO_FILE – Cannot save destination file due to invalid destination file or disk save error.

Remarks

Use this function to modify the image color scheme and resolution and save it to a file in any one of seven formats. The manipulated image may be loaded from an external file (if **SourceImage** string holds a string value equal to the source image file name) or performed on the internal image buffer (if **SourceImage** string is empty). When using a file as the image source, it is important to use the proper file extension to indicate the image format. Proper extension types are: BMP, JPG, TIFF, PCX, PNG, TGA and PSD. If an image has an unrecognizable extension due to an error (e.g. TIFF instead of TIF) the function refers to the file as BITMAP.

Image reformat can be done either on the image color scheme or the image dpi or both. Notice that changing the image format may lose the image color information (e.g., when converting from 24 bit true color to 256 gray scale). Modifying an image format from 256 gray scales to 24 bit true color will (obviously) not add color to the image but it will save the image with the proper RGB format (no color map) instead of using the 256 gray scale palette.
After the image is reformatted it can be exported to the external image file. The destination file name may be one of the seven file formats indicated above. If the destination file name has an unrecognizable extension, the function exports to the file in a BITMAP format (the default format). If no destination image file name is given (empty string), no save is done.

Important: The 256 colors scheme is NOT supported for JPG and TIF files.

ScanProcessDuplexBatch

Format

```
ScanProcessDuplexBatch ( _
    FrontImage As String,
    BackImage As String)
```

Parameters

[in] **FrontImage** – Full path name of existing front side image file. If this string is empty the rotation is performed on the internal image buffer acquired by the scanner of the front side. Front side is where the patient name is printed.

[in] **BackImage** – Full path name of existing back side image file. If this string is empty the rotation is performed on the internal image buffer acquired by the scanner of the back side. Back side is simply the side of the card were the client info is not printed.

Remarks

This starts asynchronous scanning\processing sequence of both sides. Once the command is executed it starts a scanning thread in the background and returns immediately.
The function terminate once the event **FireDataCaptureIsDone** is fired.
The SDK scan the front side and then the back side (if simplex scanner or camera is used) or both sides in one pass (if duplex scanner is used such as ScanShell800DX).
The SDK is then process both sides and extract the information from the card.
The scan of both side and the processing is done in separate threads to expedite the processing, and the SDK report on the progress by firing the following events to the application:

- **FireImageAStarted**: scan of the front side started.
- **FireImageAReady** : scan of the front side is done.
- **FireImageBStarted**: scan of the back side started.
- **FireImageBReady** : scan of the back side is done.
- **FireDataBReady** : Processing of front side is done and its data is available.
- **FireDataCaptureIsDone**: Processing of back side is done and the entire data of the card is retrieved.

Important: When using simplex scanner, the function expect that the front side is scanned FIRST and the back side scanned NEXT.

By default, the function tries to extract all possible data from the card. You may expedite the function operation by disabling the processing of specific fields, thus allow the SDK to save their processing time. This is done by calling the function **FieldsToExtract.**

ScanProcessAndSaveDuplexBatch

Format

```
ScanProcessAndSaveDuplexBatch ( _
    FrontImage As String,
    BackImage As String)
```

Parameters

[in] **FrontImage** – Full path name to use for saving the front side image. If this string is empty the rotation is performed on the internal image buffer acquired by the scanner of the front side. Front side is where the patient name is printed.

[in] **BackImage** – Full path name to use for saving the back side image. If this string is empty the rotation is performed on the internal image buffer acquired by the scanner of the back side. Back side is simply the side of the card were the client info is not printed.

Remarks

This function is similar to ScanProcessDuplexBatch with one exception: The parameters of this function indicate were to save the images acquired by the scanner. The images can be displayed by the application as soon as the events **FireImageAReady** and **FireImageAReady** are set.
If the input parameters are empty string, the function use the images aquired by the scanner without saving them to the disk, and works identically like the function ScanProcessDuplexBatch.

Events

MedicSDK file events to the application to indicate a changing event as follows:

Event Name	Parameter	Reason for event
SetCurrentText	String	The user changed the selected word\sentence.
PaperInTray	none	Indicates paper placement on the scanner tray. This event fires once or repeatedly according to the setting of the property **NotifyOnPaperInTray**
ScanIsDone	none	Indicates end of scan
ButtonPressed	Short value with one of the following values: • TOP_BUTTON • MIDDLE_BUTTON • BOTTOM_BUTTON	Indicates the user pressed the scanner's button. This event fires once or repeatedly according to the setting of the property **NotifyOnButtonPress**

Properties

Control Properties

PaperInTrayAutoStartScan

Type:

Boolean.

Direction:

Read, Write.

Remarks

If true, the control continuously monitors the paper insertion sensor and automatically starts to scan when a document is placed on the scanner's tray.

NotifyOnPaperInTray

Type:

short.

Direction:

Read, Write.

Remarks

The control may monitor the scanner's paper sensor and fire event upon document placement on the scanner's tray. Setting ***NotifyOnPaperInTray*** to the following values yields these results:
0: Ignore the paper sensor.
1: Fire the event **PaperInTray** upon document placement only once.
2: Fire the event **PaperInTray** upon document placement every 300 ms.

NotifyOnButtonPress

Type:

short.

Direction:

Read, Write.

Remarks

The control may monitor the scanner's paper sensor and fire event upon document placement on the scanner's tray. Setting ***NotifyOnButtonPress*** to the following values yields these results:
0: Ignore the paper sensor.
1: Fire the event **ButtonPressed**upon document placement only once.
2: Fire the event **ButtonPressed**upon document placement every 300 ms.

PropDuplexScan

Type:

Boolean.

Direction:

Read, Write.

Remarks

Set the scanner to scan both sides of the document if true. This operation is possible only for duplex scanner models such as ScanShell® 800DX\800DXN\3000D. This property is ignored if a simplex scanner is connected.

PropScannerModel

Type:

Boolean.

Direction:

Read, Write.

Remarks

Setting this property to the connected scanner model number, speeds up the process of the function **InitSDK.** The setting of this property should be done **BEFORE** calling the function InitSDK.
Reading this value returns the connected scanner type. The following lists the supported scanners:

- CSSN_600 (1)
- CSSN_800 (2)
- CSSN_800N (3)
- CSSN_1000 (4)
- CSSN_2000 (5)
- CSSN_2000N (6)
- CSSN_3000 (9)
- CSSN_4000 (10)
- CSSN_5000 (12)
- CSSN_800DX (14)
- CSSN_800DXN (15)
- CSSN_IDR, CSSN_FDA, CSSN_WMD, CSSN_TWN (13)

Data Fields Properties

PropFrontSide

Type:

Boolean.

Direction:

Read Only.

Remarks

True if the scanned image is the image of the front side of the medical insurance document.

PropPlanProvider

Type:

String.

Direction:

Read Only.

Remarks

Plan provider name

PropMemberName

Type:

String.

Direction:

Read Only.

Remarks

Member\Card holder name

PropMemberID

Type:

String.

Direction:

Read Only.

Remarks

Member\Card holder ID number

PropGroupNumber

Type:

String.

Direction:

Read Only.

Remarks

Group number value.

PropPayerID

Type:

String.

Direction:

Read Only.

Remarks

Payer identification number.

PropCopayOV

Type:

String.

Direction:

Read Only.

Remarks

Copay O/V rate.

PropCopaySP

Type:

String.

Direction:

Read Only.

Remarks

Copay S/P rate.

PropCopayUC

Type:

String.

Direction:

Read Only.

Remarks

Copay U/C rate.

PropCopayER

Type:

String.

Direction:

Read Only.

Remarks

Copay E/R rate.

PropEffectiveDate

Type:

String.

Direction:

Read Only.

Remarks

Effective date value.

PropExpireDate

Type:

String.

Direction:

Read Only.

Remarks

Expiration date value.

PropDateOfBirth

Type:

String.

Direction:

Read Only.

Remarks

Birth date value.

PropTelTotalItems

Type:

long.

Direction:

Read Only.

Remarks

This property holds the amount of telephone number fields found on the card.

PropTelLabel

Type:

String

Direction:

Read Only.

Remarks

This property holds the label string of the corresponding telephone field index. To retrieve this field you must supply the telephone label index value. This index value can be between 0 to *PropTelTotalItems-1*

PropTelValue

Type:

String

Direction:

Read Only.

Remarks

This property holds the telephone number of the corresponding telephone field index. To retrieve this field you must supply the telephone label index value. This value can be between 0 to *PropTelTotalItems-1.*

PropEmailTotalItems

Type:

long.

Direction:

Read Only.

Remarks

This property holds the amount of email fields found on the card.

PropEmailLabel

Type:

String

Direction:

Read Only.

Remarks

This property holds the label string of the corresponding email field index. To retrieve this field you must supply the email label index value. This index value can be between 0 to *PropEmaiTotalItems-1*

PropEmailValue

Type:

String

Direction:

Read Only.

Remarks

This property holds the email address of the corresponding email field index. To retrieve this field you must supply the email index value. This index value can be between 0 to *PropEmailTotalItems-1*

PropWebTotalItems

Type:

long.

Direction:

Read Only.

Remarks

This property holds the amount of Web fields found on the card.

PropWebLabel

Type:

String

Direction:

Read Only.

Remarks

This property holds the label string of the corresponding Web field index. To retrieve this field you must supply the Web label index value. This index value can be between 0 to *PropEmaiTotalItem-1*

PropWebValue

Type:

String

Direction:

Read Only.

Remarks

This property holds the Web address of the corresponding Web field index. To retrieve this field you must supply the Web index value. This index value can be between 0 to *PropWebTotalItem-1*

PropAddressTotalItems

Type:

long.

Direction:

Read Only.

Remarks

This property holds the amount of Address fields found on the card.

PropAddressFull

Type:

String

Direction:

Read Only.

Remarks

This property holds the full address string of the corresponding address field index. To retrieve this field you must supply the address index value. This index value can be between 0 to *PropAddressTotalItems-1*

PropAddressStreet

Type:

String

Direction:

Read Only.

Remarks

This property holds the street string of the corresponding address field index. To retrieve this field you must supply the address index value. This index value can be between 0 to *PropAddressTotalItems-1*

PropAddressCity

Type:

String

Direction:

Read Only.

Remarks

This property holds the city string of the corresponding address field index. To retrieve this field you must supply the address index value. This index value can be between 0 to *PropAddressTotalItems-1*

PropAddressState

Type:

String

Direction:

Read Only.

Remarks

This property holds the state string of the corresponding address field index. To retrieve this field you must supply the address index value. This index value can be between 0 to *PropAddressTotalItems-1*

PropAddressZip

Type:

String

Direction:

Read Only.

Remarks

This property holds the zip string of the corresponding address field index. To retrieve this field you must supply the address index value. This index value can be between 0 to *PropAddressTotalItems-1*

PropOther

Type:

String

Direction:

Read Only.

Remarks

This property holds all the strings that were not assigned to other fields.

propRawText

Type:

String

Direction:

Read Only.

Remarks

This property holds all the text on the card as bulk text.

propContractCode

Type:

String

Direction:

Read Only.

propPlanType

Type:

String

Direction:

Read Only.

propDeductible

Type:

String

Direction:

Read Only.

propRxBin

Type:

String

Direction:

Read Only.

propRxPCN

Type:

String

Direction:

Read Only.

propRxGroup

Type:

String

Direction:

Read Only.

propEmployer

Type:

String

Direction:

Read Only.

propCoverage

Type:

String

Direction:

Read Only.

propPlanCodeTotalItems

Type:

Long

Direction:

Read Only.

Remarks

This property contains the number of Plan Code values found on the card.

propPlanCode

Type:

String

Direction:

Read Only.

Remarks

This property holds the Plan Code of the corresponding field index. To retrieve this field you must supply the Plan Code index value. This index value can be between 0 to *PropPlanCodeTotalItems-1.*

propRxId

Type:

String

Direction:

Read Only.

propPlanAdmin

Type:

String

Direction:

Read Only.

propGroupName

Type:

String

Direction:

Read Only.

propIssuerNumber

Type:

String

Direction:

Read Only.

propNameFirst

Type:

String

Direction:

Read Only.

propNameMiddle

Type:

String

Direction:

Read Only.

propNameLast

Type:

String

Direction:

Read Only.

propNamePrefix

Type:

String

Direction:

Read Only.

propNameSuffix

Type:

String

Direction:

Read Only.

PropDeductibleTotalItems

Type:

Long

Direction:

Read Only.

Remarks

This property contains the number of Deductible values found on the card.

PropDeductibleLabel

Type:

String

Direction:

Read Only.

Remarks

This property holds the Deductible label of the corresponding field index. To retrieve this field you must supply the Deductible index value. This index value can be between 0 to *PropDeductibleTotalItems-1.*

PropDeductibleValue

Type:

String

Direction:

Read Only.

Remarks

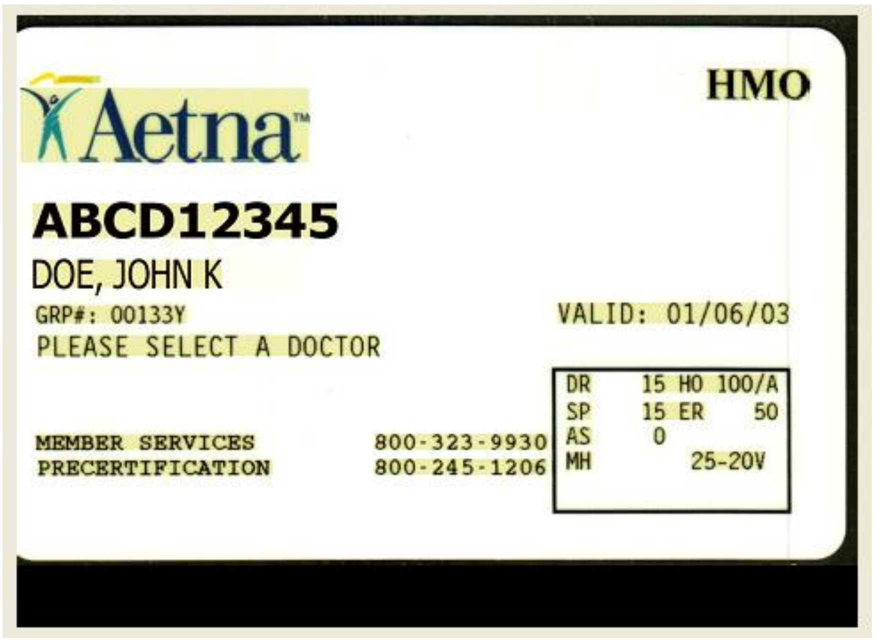
This property holds the Deductible value of the corresponding field index. To retrieve this field you must supply the Deductible index value. This index value can be between 0 to *PropDeductibleTotalItems-1.*

User Interface

Processing medical insurance cards is done by analyzing the text on the card and assigning the text into the relevant fields. MedicSDK ActiveX also offers to assign the textual data to their destination fields in the hosting application manually, and this is done after the component analyzes the image and highlights each word on the document.
The user may now select the right words by clicking on them and dragging the text to the destination. Once a "drop" is done, the destination file will be filled with the dragged text.

Selecting and dragging a single word

The following image show a card image as displayed on the control immediately after calling the function *ProcessMedical*:



Aetna™ HMO

ABCD12345

DOE, JOHN K

GRP#: 00133Y VALID: 01/06/03

PLEASE SELECT A DOCTOR

MEMBER SERVICES 800-323-9930

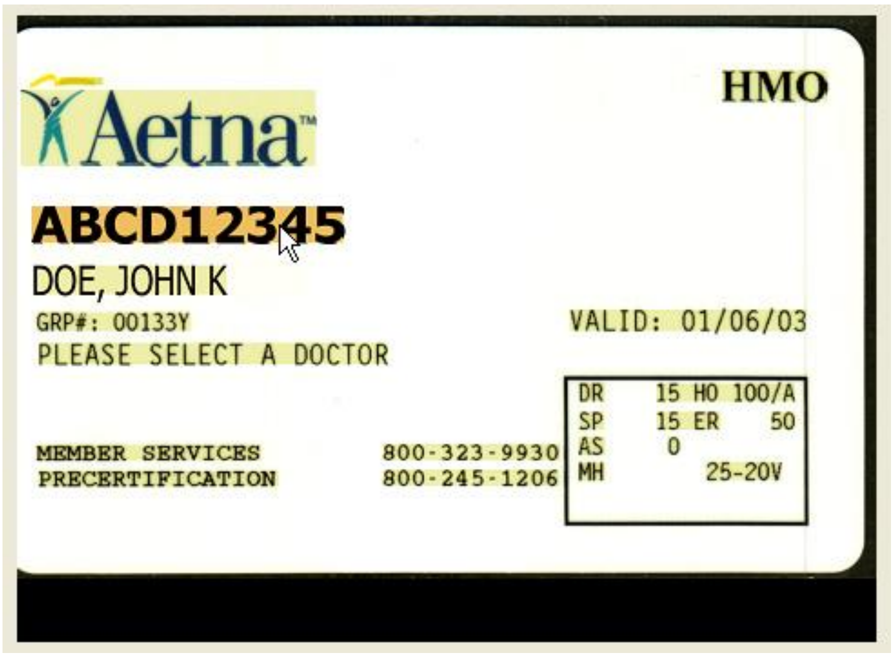
PRECERTIFICATION 800-245-1206

DR	15	HO	100/A
SP	15	ER	50
AS	0		
MH			25-20V

Figure 1: Card image shown by the control immediately after processing

Notice that the function *ProcessMedical* automatically marked each word.

Once the user moves the mouse over a word, its color changes to indicate to the user that it can be selected:



Aetna™ HMO

ABCD12345

DOE, JOHN K

GRP#: 00133Y VALID: 01/06/03

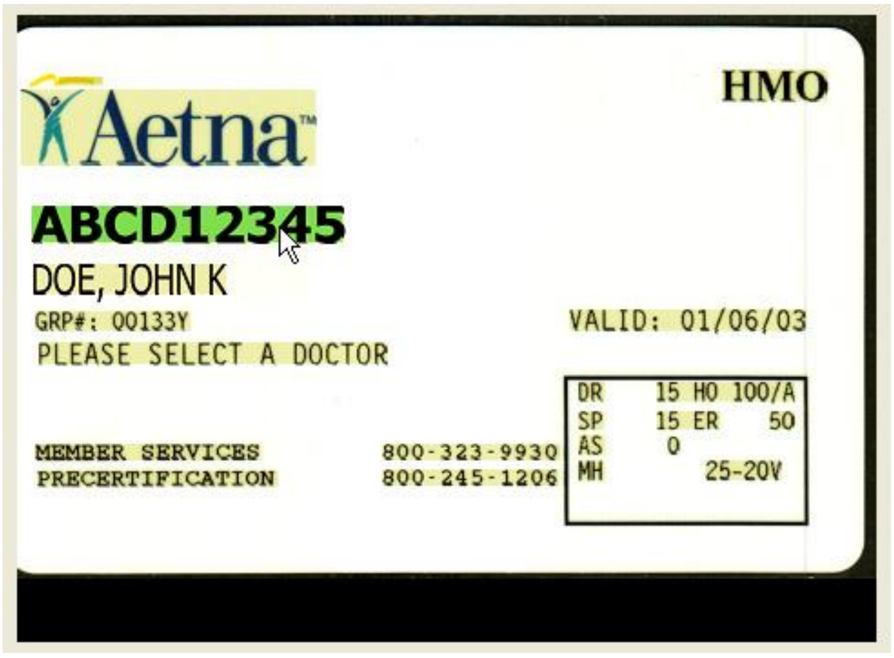
PLEASE SELECT A DOCTOR

MEMBER SERVICES 800-323-9930

PRECERTIFICATION 800-245-1206

DR	15	HO	100/A
SP	15	ER	50
AS	0		
MH			25-20V

Once the user clicks on the word with the mouse, the word is marked in green. In addition, the component fires the event ***SetCurrentText*** to the application with the content of the selected word.



Aetna™ HMO

ABCD12345

DOE, JOHN K

GRP#: 00133Y VALID: 01/06/03

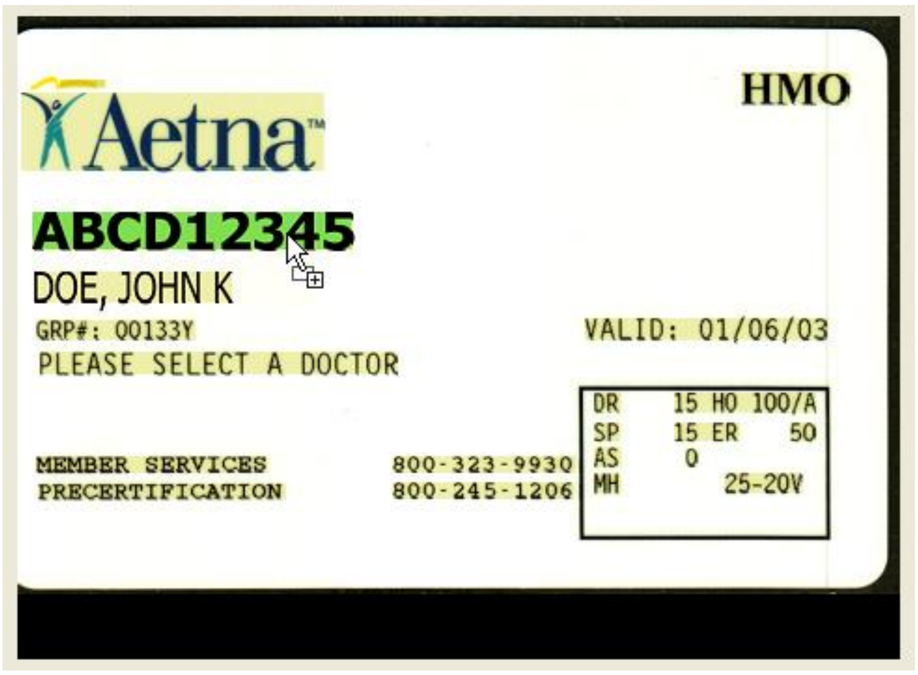
PLEASE SELECT A DOCTOR

MEMBER SERVICES 800-323-9930

PRECERTIFICATION 800-245-1206

DR 15 HO 100/A
SP 15 ER 50
AS 0
MH 25-20V

Moving the mouse while holding the left button pressed changes the mouse cursor shape into a dragging cursor and the user can now drag the mouse over the destination filed and release the mouse button to drop the text.



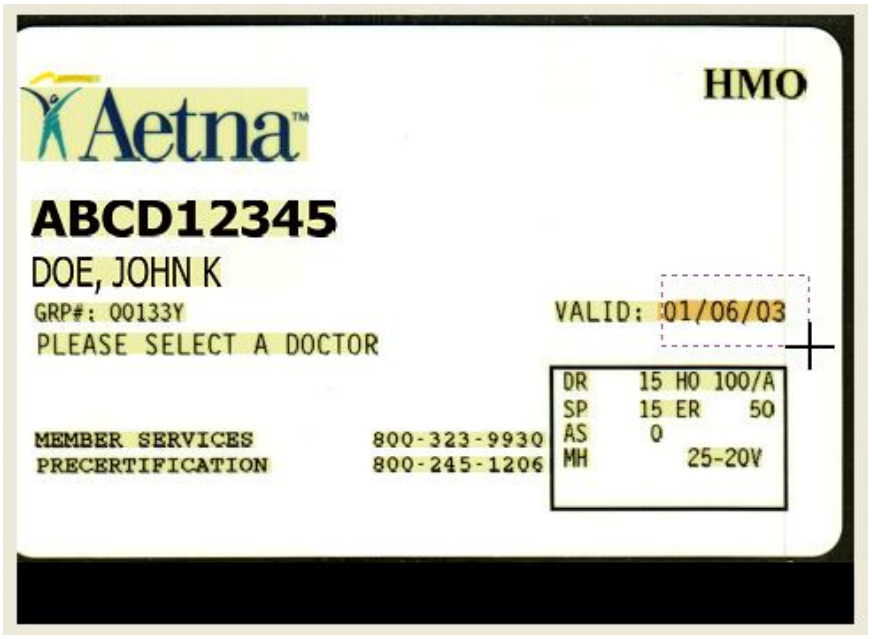
ABCD12345
DOE, JOHN K
GRP#: 00133Y
VALID: 01/06/03
PLEASE SELECT A DOCTOR

DR	15	HO	100/A
SP	15	ER	50
AS	0		
MH			25-20V

MEMBER SERVICES 800-323-9930
PRECERTIFICATION 800-245-1206

Selecting and dragging a partial word

The user may select a partial word to remove unnecessary letters. During the mouse drag, the selected letters accumulate to a word and are highlighted in orange. Once the desired letters are selected, the user can drag the word to the destination field.



Aetna™

HMO

ABCD12345

DOE, JOHN K

GRP#: 00133Y

VALID: 01/06/03

PLEASE SELECT A DOCTOR

DR	15	HO	100/A
SP	15	ER	50
AS	0		
MH			25-20V

MEMBER SERVICES 800-323-9930

PRECERTIFICATION 800-245-1206

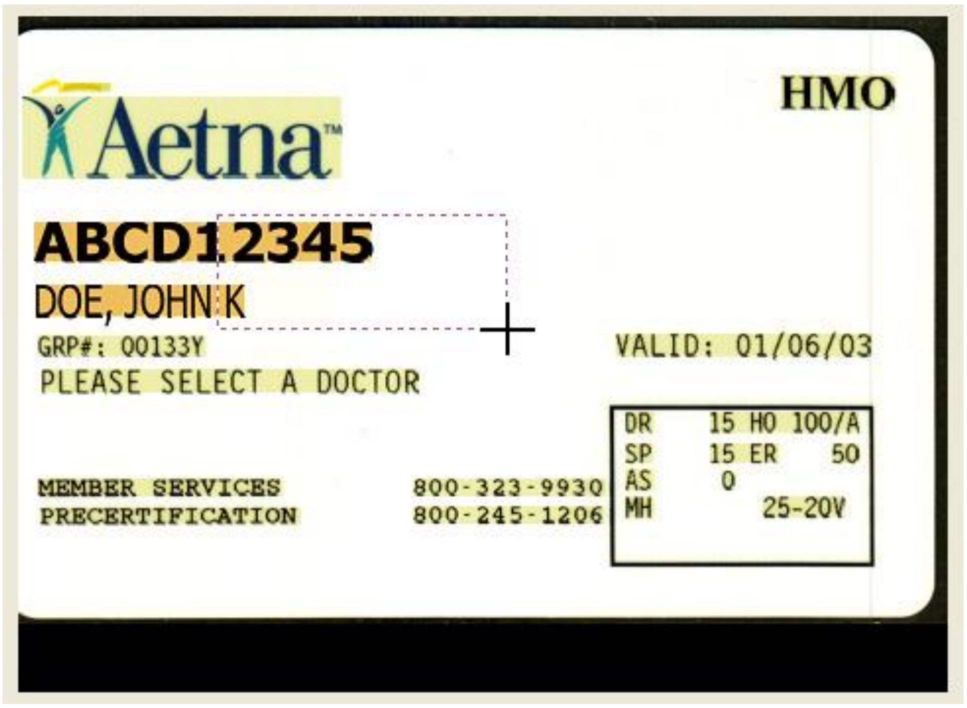
Selecting and dragging multiple words

The user may select multiple words and drag them to the destination field. This is useful to extract multiple-words fields such as address. There are two ways to select multiple words:

- Defining an area using the mouse and collecting all the words in the marked area.
- Clicking on different words while holding the SHIFT button.

In both ways the control concatenates the selected words by their order of selection. Once a word is selected, its color changes into green and the control fires the event ***SetCurrentText*** to the application with the content of the selected words.

The following displays how to select multiple words using the first method:



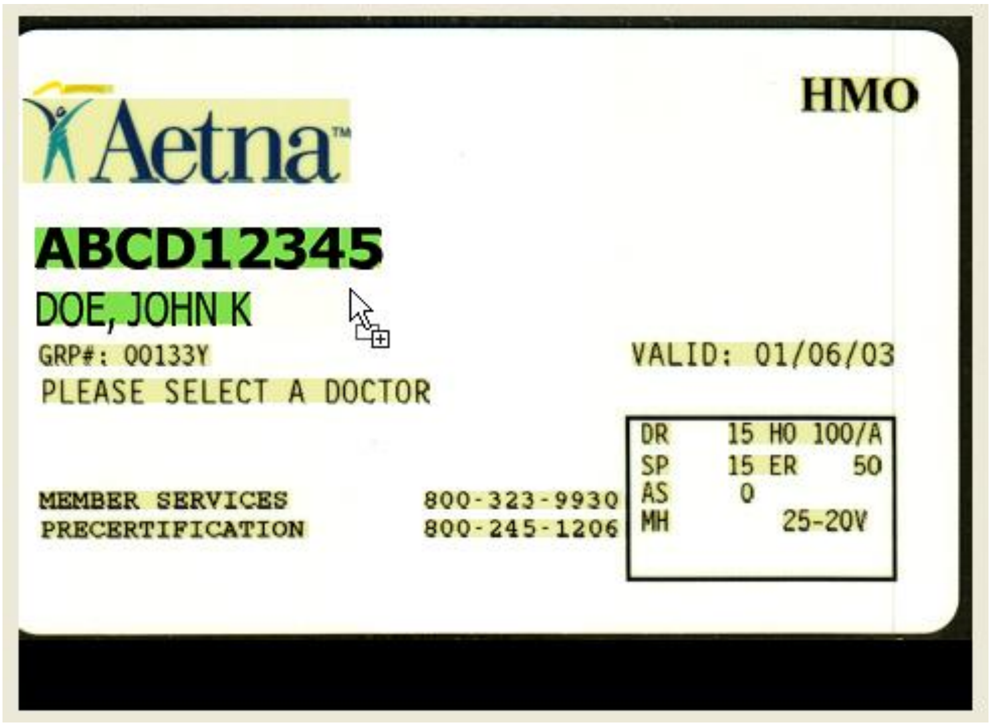
Aetna™
HMO

ABCD12345
DOE, JOHN K
GRP#: 00133Y
VALID: 01/06/03
PLEASE SELECT A DOCTOR

MEMBER SERVICES	800-323-9930
PRECERTIFICATION	800-245-1206

DR	15	HO	100/A
SP	15	ER	50
AS	0		
MH			25-20V

Once the mouse button is released the selected sentence is set to **ABCD12345 DOE, JOHN K**, the text color changes into green to indicate the selection. At this point the user can drag this sentence and drop it over the application:



Aetna™

HMO

ABCD12345

DOE, JOHN K

GRP#: 00133Y

VALID: 01/06/03

PLEASE SELECT A DOCTOR

DR	15	HO	100/A
SP	15	ER	50
AS	0		
MH			25-20V

MEMBER SERVICES 800-323-9930
PRECERTIFICATION 800-245-1206

Appendix A – SDK Installation and Registration

Installing the SDK package

The SDK files are packed in a single setup file (*medicscan_SDK.exe*). Installing the setup file extracts the following file list:

File name	Destination directory	Functionality
imgForm.DLL	Windows\system32	Image processing tools collection
Dic.dll	Sdk destination directory	Dictionary library
CImage.dll	Sdk destination directory	
ImageCtrl.dll	Sdk destination directory	Image processing tools collection
Lib2.lib	Sdk destination directory	
Lib3.lib	Sdk destination directory	
OCR_PreProc.dll	Sdk destination directory	Driver's license image analyzer
Medical.bin		
MedicSDK.ocx		ActiveX main module
MedLib.dll		
SLib.dll	Sdk destination directory	Controls the scanner activity
Msvbvm60.dll		
test5.gar	Sdk destination directory	OCR library
test5.n3s	Sdk destination directory	OCR library
test5.qnp	Sdk destination directory	OCR library
test5.teh	Sdk destination directory	OCR library
TOCRR.ini	Sdk destination directory	OCR library
TOCRRdll.dll	Sdk destination directory	OCR library
TOCRRService.exe	Sdk destination directory	OCR library
SOCRdll.dll	Sdk destination directory	OCR library
ATL.DLL	Windows\system32	Part of Windows system (**Installs only if newer**)
MSI.DLL	Windows\system32	Part of Windows system (**Installs only if newer**)

After the extraction, the installation program registers MedicSDK.ocx.

Manual Registration 1:

Another method to register the MedciSDK.ocx library is to use the mouse right click button. To use this option you must first merge the attached OCX file '**ocxdllreg.reg**' (can be found under TOOLS directory) by clicking on it and selecting the YES button when asked.
This will enable the option to register MedciSDK.ocx using the mouse when you click on the file name in Windows Explorer with the mouse right button.

Manual Registration 2:

An additional method to register Scanw.dll is to open a shell command prompt in the SDK files destination directory and to type the following command:

```
REGSVR32.EXE MedciSDK.ocx
```

Appendix B – Constant Values and Return Codes

Library SlibEx constants

```
' Scanner color scheme types
Public Const GRAY = 1
Public Const BW = 2
Public Const HT = 3
Public Const TRUECOLOR = 4

' Scanner return values
Public Const SLIB_FALSE = 0
Public Const SLIB_TRUE = 1

' Scanner general error types
Public Const SLIB_ERR_NONE = 1
Public Const SLIB_ERR_INVALID_SCANNER = -1

' Scanning failure definition
Public Const SLIB_ERR_SCANNER_GENERAL_FAIL = -2
Public Const SLIB_ERR_CANCELED_BY_USER = -3
Public Const SLIB_ERR_SCANNER_NOT_FOUND = -4
Public Const SLIB_ERR_HARDWARE_ERROR = -5
Public Const SLIB_ERR_PAPER_FED_ERROR = -6
Public Const SLIB_ERR_SCANABORT = -7
Public Const SLIB_ERR_NO_PAPER = -8
Public Const SLIB_ERR_PAPER_JAM = -9
Public Const SLIB_ERR_FILE_IO_ERROR = -10
Public Const SLIB_ERR_PRINTER_PORT_USED = -11
Public Const SLIB_ERR_OUT_OF_MEMORY = -12

Public Const SLIB_ERR_BAD_WIDTH_PARAM = -2
Public Const SLIB_ERR_BAD_HEIGHT_PARAM = -3
Public Const SLIB_ERR_BAD_PARAM = -2

Public Const SLIB_LIBRARY_ALREADY_INITIALIZED = -13
Public Const SLIB_ERR_DRIVER_NOT_FOUND = -14
Public Const SLIB_ERR_SCANNER_BUSSY  = -15

' Button definition for ScanShell1000
Public Const TOP_BUTTON = 1
Public Const MIDDLE_BUTTON = 3
Public Const BOTTOM_BUTTON = 2

' supported scanner modules
Public Const CSSN_600 = 1
Public Const CSSN_800 = 2
Public Const CSSN_800N = 3
Public Const CSSN_1000 = 4
Public Const CSSN_2000 = 5
Public Const CSSN_2000N = 6
Public Const CSSN_800E = 7
Public Const CSSN_800EN = 8
Public Const CSSN_3000 = 9
Public Const CSSN_4000 = 10
Public Const CSSN_800G = 11
Public Const CSSN_5000 = 12
Public Const CSSN_800DX = 14
Public Const CSSN_800DXN = 15
Public Const CSSN_IDR = 13
Public Const CSSN_FDA = 16
Public Const CSSN_WMD = 17
Public Const CSSN_TWN = 18
```

Library CImage constants

```
' return values
Public Const IMG_ERR_SUCCESS = 0
Public Const IMG_ERR_FILE_OPEN = -100
Public Const IMG_ERR_BAD_ANGLE_0 = -101
Public Const IMG_ERR_BAD_ANGLE_1 = -102
Public Const IMG_ERR_BAD_DESTINATION = -103
Public Const IMG_ERR_FILE_SAVE_TO_FILE = -104
Public Const IMG_ERR_FILE_SAVE_TO_CLIPBOARD = -105
Public Const IMG_ERR_FILE_OPEN_FIRST = -106
Public Const IMG_ERR_FILE_OPEN_SECOND = -107
Public Const IMG_ERR_COMB_TYPE = -108

Public Const IMG_ERR_BAD_COLOR = -130
Public Const IMG_ERR_BAD_DPI = -131
Public Const INVALID_INTERNAL_IMAGE = -132

' image saving target definition
Public Const SAVE_TO_FILE = 0
Public Const SAVE_TO_CLIPBOARD = 1

' image rotation angle definitions
Public Const ANGLE_0 = 0
Public Const ANGLE_90 = 1
Public Const ANGLE_180 = 2
Public Const ANGLE_270 = 3

' image combination options
Public Const IMAGE_COMB_VERTICAL = 0
Public Const IMAGE_COMB_HORIZONTAL = 1

' image color conversion
 Public Const IMAGE_SAME_COLOR = 0
 Public Const IMAGE_BW = 1
 Public Const IMAGE_GRAY_256 = 2
 Public Const IMAGE_COLOR_256 = 3
 Public Const IMAGE_COLOR_TRUE = 4
```

Library COcr constants

```
' return values
Public Const TOCR_SUCCESS = 1
Public Const TOCRJOBERROR = -2
Public Const TOCR_BAD_TYPE = -3

' OCR text type detection
Public Const USE_ALPHANUM = 0
Public Const USED_NUM_ONLY = 2
Public Const USE_ALPHA_CAPS_ONLY = 3
```

License related constants

```
Public Const LICENSE_VALID = 1
Public Const LICENSE_EXPIRED = -20
Public Const LICENSE_INVALID = -21
Public Const LICENSE_DOES_NOT_MATCH_LIBRARY = -22
Public Const GENERAL_ERR_PLUG_NOT_FOUND = -200
```