ScanWex.DLL

SDK Library Extension Description

January 2016

Table of Contents

Introduction

The Card Scanning Solutions SDK continuously grows with time, adding new features and capabilities. As the core SDK library interface (*ScanW.dll*) is sealed due to backward compatibility, the need for additional features and functions is implemented using this ScanW extended library. For example, ScanWex implements new features such as driver license auto detection by global region selection, such as continents, thus extending the countries auto-detection feature that is implemented on the old ScanW library.

The ScanWex library **IS NOT** a stand-alone library. It must be used only while the ScanW.dll library is installed.

Distribution

To install the SDK files at the destination computer, you simply need to copy all the SDK files that are in the SDK installation folder to the destination computer.

There are some files that will need to be registered on the destination computer such as COM\ActiveX objects. Install these files at the end of the SDK files installation since it will need the non COM\ActiveX files to exist before registration.

Here is a list of the files that need to be registered:

- ScanW.dll (Com object)
- ScanWEx.dll (Com object)
- ScanX.dll (ActiveX object) - mostly used for VB scripts

Note:

If you do not use the COM interface in your application and you use the SDK files directly like in VC++, then you do not need to install these files on the destination computer.

Library Properties and Methods

Library idData: General Functionality

idData library extends the function set introduced in the idData library of the ScanW wrapper library. The library exports the capability to automatically detect the current card type by setting the current region. The term ‘Region’ refers to a collection of countries in the same vicinity, and in the near future, as more and more countries are added to the analyzing engine database, this term will converge to the 5 major continents.

idData Library Functions

Country2Id

Format

```
Country2Id (CountryName As String) As Long
```

Parameters
[in] **CountryName** – Country name as retrieved from the SDK.
Return
1) ID_ERR_NO_MATCH: The parameter *CountryName* does not contain/ know the country name.
2) Country ID number.

Remarks
Use this helper function to get the Country ID number.

RegionSet

Format

```
RegionSet (RegionId As Integer) As Long
```

Parameters
[in] **RegionId** – The current region ID.
Return
ID_ERR_AUTO_DETECT_NOT_SUPPORTED: The parameter *RegionId* does not support auto detection.
ID_ERR_NO_MATCH: The parameter *RegionId* is not a valid region identifier.
ID_TRUE – Region setting succeeded.

Remarks
The SDK can automatically detect the card type using the following steps:

- Setting the region number.
- Running the function AutoDetectStateEx (As described in the ScanW.pdf programming reference) after scanning the image.

Currently, the regions that support auto detections are:
USA (Region Number 0); Canada (Region Number 1); Australia (Region Number 4) and Asia (Region Number 5).

The auto detection will support additional regions in the upcoming versions. For a complete list of the regions and their related states, please refer to Appendix F (In the ScanW.pdf document).

<u>**Note**</u>: Upon invocation of the library, the default region is set to USA (Region Number 0).

RegionSetDetectionSequence

Format

```
RegionSetDetectionSequence (lRegionId0 As Long, lRegionId1 As Long,
lRegionId2 As Long, lRegionId3 As Long, lRegionId4 As Long, lRegionId5 As
Long, lRegionId6 As Long) As Long
```

Parameters
[in] **RegionId0 - 6** – The region ID's.
Return
ID_ERR_AUTO_DETECT_NOT_SUPPORTED: The parameter *RegionId* does not support auto detection.
ID_ERR_NO_MATCH: The parameter *RegionId* is not a valid region identifier.
ID_TRUE – Region setting succeeded.

Remarks
Please read the description of the function RegionSet.
This function will do the same but this time the SDK will try to find the card type in all the given regions ID's in this function.
<u>Example:</u> Call this function with a 0 and 1 for USA & Canada respectively and now whenever you call the function DetectStateEx with USA or Canadian cards, it will find the card.

RegionGetFirst

Format

```
RegionGetFirst () As Long
```

Return

The value of the first region in the SDK region list.

Remarks

The SDK divides the supported countries into several major regions. The regions are organized in a list that can be retrieved by calling to this function once, and then continuously calling the function *RegionGetNext*().

Please note the following:

The list size and order may vary in the future, as the SDK will include more and more regions. For example, regions 0 and 1 (USA and Canada respectively) are designated to be unified in to a single region (North America region) in later versions. Therefore, you should never fix the region values in your code and always retrieve it using the function *RegionGetFirst* and *RegionGetNext.*
Not all the regions support auto detection. However, in following versions, more and more regions will support this feature.

RegionGetNext

Format

```
RegionGetNext () As Long
```

Return

ID_ERR_NO_MATCH: The function *RegionGetFirst* was never called prior to the call of this function.
ID_ERR_NO_NEXT_COUNTRY: The list has ended (the last region was retrieved in the previous function call).

Otherwise, the function returns the value of the next region ID.

Remarks

Use this function to retrieve the region list. To retrieve the list, do the following:

- Call the function *RegionGetFirst* which is called once.
- Call *RegionGetNext* continuously in a loop until the value **ID_ERR_NO_NEXT_COUNTRY** is returned. For each call, the function returns the ID of the next region.

RegionGetNameById

Format

```
RegionGetNameById (RegionId As Integer, RegionName As String) As
Long
```

Parameters

[in] **RegionId** – The region's numeric value.

[out] **RgionName** – A string that accepts the region name.

Return

ID_ERR_NO_MATCH – The parameter *RegionId* is not a valid region enum.
ID_TRUE – Region found. The *RegionName* string returns loaded with the region name.

Remarks

Use this helper function to convert region enum values into the region string name. This function, combined with the functions *RegionGetFirst* and *RegionGetNext* builds the region name list.

RegionGetIdByName

Format

```
RegionGetIdByName (RegionName As String, RegionId As Integer) As
Long
```

Parameters

[in] **RegionName** – The region name.

[out] **RgionId** – An integer that accepts the region ID.

Return

ID_ERR_NO_MATCH – The parameter *RegionName* is not a valid region name.
ID_TRUE – Region found. The *RegionId* returns loaded with the region ID.
RegionID – The match region ID number that was found.

Remarks

Use this helper function to convert region name values into the region ID.

RegionAutoDetectSupport

Format

```
RegionAutoDetectSupport (RegionId As Integer) As Long
```

Parameters

[in] **RegionId**– The region enum value.

Return

ID_ERR_NO_MATCH – The input parameter *RegionId* value does not contain a valid region enum.
ID_FALSE – The parameter *RegionId* represents a region that does not support auto detection.
ID_TRUE – The parameter *RegionId* represents a region that supports auto detection.

Remarks

Use this function to detect if a document of a specific state can be automatically detected by the function *AutoDetectStateEx* after scanning the document. If so, *AutoDetectStateEx* will return the proper state enum, and this value can be used as an input parameter to the function *ProcState* that retrieves the data from the image.
If *RegionAutoDetectSupport* returns **ID_FALSE,** than you should skip the call to *AutoDetectStateEx* and call *ProcState* directly while setting the state ID parameter manually.

RegionByCountry

Format

```
RegionByCountry (Country As Integer) As Long
```

Parameters

[in] **Country** – Constant value of a country.

Return value

ID_ERR_NO_MATCH - The parameter *Country* does not contain a valid country ID.

Otherwise, this function returns the region ID that contains the input country.

Remarks

Use this function to detect the region that contains the country. For a complete list of the regions, countries and states please refer to Appendix F (In the ScanW.pdf document).

RegionGetFirstCountry

Format

```
RegionGetFirstCountry (region As Integer) As Long
```

Parameters
[in] **region** – The region ID that contains the country list.

Return
If the function succeeds, the return value is **ID_TRUE**
If the function fails, the following value is returned:
ID_ERR_NO_MATCH – The value in *region* is not a valid region ID.

Otherwise, the function returns the first country ID in the region.

Remarks
Use this function (combined with *RegionGetNextCountry)* to obtain the list of countries in the region.

RegionGetNextCountry

Format

```
RegionGetNextCountry (region As Integer) As Long
```

Parameters
[in] **region** – The region ID that contains the country list.

Return
If the function succeeds, the return value is **ID_TRUE**
If the function fails, one of the following values is returned:
ID_ERR_NO_MATCH – The value in *region* is not a valid region ID.
ID_ERR_NO_NEXT_COUNTRY – There is no next country in the list – the last call retrieved the last country in the list.

Otherwise, the function returns the next country ID in the region.

Remarks
Use this function (combined with *RegionGetFirstCountry)* to obtain the list of countries in the region. You should call *RegionGetFirstCountry* once, than call *RegionGetNextCountry* continuously in a loop and store the returned value until the value **ID_ERR_NO_NEXT_COUNTRY** is returned. You can use the following strategy to build a complete region, country and state tree:

Retrieve the full region list using *RegionGetFirst* and *RegionGetNext* functions.
For each Region, retrieve the full countries list using *RegionGetFirstCountry* and *RegionGetNextCountry* functions.
For each Country, retrieve the full state list using *GetFirstStateByCountry* and *GetNextStateByCountry* functions.

Id2Country2

Format

```
Id2Country2 (id As Integer, country As String) As Long
```

Parameters
[in] **id** – The country ID number.
[out] **country** – The country name in a 2 letter format (ISO 3166).

Return
If the function succeeds, the return value is **ID_TRUE.**
If the function fails, the following value is returned:
ID_ERR_NO_MATCH – The value in *id* is not a valid country ID.

Remarks
Use this helper function to convert between the country ID number and the country name. The returned name is the country name in 2 letters as defined by ISO 3166 spec.
For the country ID number complete list, please refer to Appendix F (In the ScanW.pdf document).

Id2Country3

Format

```
Id2Country3 (id As Integer, country As String) As Long
```

Parameters
[in] **id** – The country ID number.
[out] **country** – The country name in a 3 letters format (ISO 3166).

Return
If the function succeeds, the return value is **ID_TRUE.**
If the function fails, the following value is returned:
ID_ERR_NO_MATCH – The value in *id* is not a valid country ID.

Remarks
Use this helper function to convert between the country ID number and the country name. The returned name is the country name in 3 letters as defined by ISO 3166 spec.
For the country ID number complete list, please refer to Appendix F (In the ScanW.pdf document).

GetFaceImageBuffer

Format

```
GetFaceImageBuffer (Source As String, buffer As String, FileType As
String, StateID as Integer) As Long
```

Parameters
[in] **Source** – Name of the source driver's license image file.
[in] **buffer** – A string buffer that will contain the image content upon a successful return. FileType.
[in] **FileType** – Set the buffer image format. May be one of the following strings (case insensitive):
"BMP"
"JPG"
"PNG"
"TIF"
"TGA"
"PSD"
"PCX"
[in] **StateID** – The state ID.

Return
If the function succeeds, the return value is **ID_TRUE**.
If the function fails, the value **ID_FALSE** is returned.

Remarks
Use this function to extract the face image from the document image to the given buffer. The source document image can be an existing image file (in BMP format) or the last scanned document (stored in the internal image buffer). If the parameter **SourceFile** is an empty string, then the image is taken from the internal image buffer. If this parameter contains the full name of a valid image file, then this file is used.

GetSignatureImageBuffer

Format

```
GetSignatureImageBuffer (Source As String, buffer As String, FileType As
String, StateID As Integer) As Long
```

Parameters
[in] **Source** – Name of the source driver's license image file.
[in] **buffer** – A string buffer that will contain the image content upon successful return. FileType.
[in] **FileType** – Set the buffer image format. May be one of the following strings (case insensitive):
"BMP"
"JPG"
"PNG"
"TIF"
"TGA"
"PSD"
"PCX"
[in] **StateID** – The state ID.

Return
If the function succeeds, the return value is **ID_TRUE.**
If the function fails, the value **ID_FALSE** is returned**.**

Remarks
Use this function to extract the signature image from the document image to the given buffer. The source document image can be an existing image file (in BMP format) or the last scanned document (stored in the internal image buffer). If the parameter **SourceFile** is an empty string, then the image is taken from the internal image buffer. If this parameter contains the full name of a valid image file, then this file is used.

GetFaceImageBufferEx

Format

GetFaceImageBufferEx (**Source** As String, **buffer** As String, **FileType** As String, **StateID** as Integer, **ToColor** as Integer, **ToDPI** as Integer) As Long

Parameters
[in] **Source** – Name of the source driver's license image file.
[in] **buffer** – A string buffer that will contain the image content upon a successful return. FileType.
[in] **FileType** – Set the buffer image format. May be one of the following strings (case insensitive):
"BMP"
"JPG"
"PNG"
"TIF"
"TGA"
"PSD"
"PCX"
[in] **StateID** – The state ID.
[in] **ToColor** – Set the new Image scheme. Can be one of the following values:
- **(0) IMAGE_SAME_COLOR** – No modification in the image color scheme
- **(1) IMAGE_BW** – Convert to black and white color scheme.
- **(2) IMAGE_GRAY_256** – Convert to 256 gray scale color scheme.
- **(3) IMAGE_COLOR_256** – Convert to 256-color scheme.
- **(4) IMAGE_COLOR_TRUE** – Convert to true color scheme

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

Return
If the function succeeds, the return value is **ID_TRUE**.
If the function fails, the value **ID_FALSE** is returned.

Remarks
Use this function to extract the face image from the document image to the given buffer. The source document image can be an existing image file (in BMP format) or the last scanned document (stored in the internal image buffer). If the parameter **SourceFile** is an empty string, then the image is taken from the internal image buffer. If this parameter contains the full name of a valid image file, then this file is used. You can change the color and DPI of the original image and get it with the new color and DPI according to the given values in the new buffer image.

GetSignatureImageBufferEx

Format

GetSignatureImageBufferEx (**Source** As String, **buffer** As String, **FileType** As String, **StateID** As Integer, **ToColor** as Integer, **ToDPI** as Integer) As Long

Parameters
[in] **Source** – Name of the source driver's license image file.
[in] **buffer** – A string buffer that will contain the image content upon successful return. FileType.
[in] **FileType** – Set the buffer image format. May be one of the following strings (case insensitive):
"BMP"
"JPG"
"PNG"
"TIF"
"TGA"
"PSD"
"PCX"
[in] **StateID** – The state ID.
[in] **ToColor** – Set the new Image scheme. Can be one of the following values:

- **(0) IMAGE_SAME_COLOR** – No modification in the image color scheme
- **(1) IMAGE_BW** – Convert to black and white color scheme.
- **(2) IMAGE_GRAY_256** – Convert to 256 gray scale color scheme.
- **(3) IMAGE_COLOR_256** – Convert to 256-color scheme.
- **(4) IMAGE_COLOR_TRUE** – Convert to true color scheme

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

Return
If the function succeeds, the return value is **ID_TRUE.**
If the function fails, the value **ID_FALSE** is returned**.**

Remarks
Use this function to extract the signature image from the document image to the given buffer. The source document image can be an existing image file (in BMP format) or the last scanned document (stored in the internal image buffer). If the parameter **SourceFile** is an empty string, then the image is taken from the internal image buffer. If this parameter contains the full name of a valid image file, then this file is used. You can change the color and DPI of the original image and get it with the new color and DPI according to the given values in the new buffer image.

GetFaceImageDuplex

Format

```
GetFaceImageDuplex (SourceFileName As String, BackSourceFileName
As String, DestFileName As String, StateID As Integer) As Long
```

Parameters
[in] **SourceFileName** – Null terminated string that holds the full path of the scanned ID image. If this string is empty the internal image is used as the image source.
[in] **BackSourceFileName** – Null terminated string that holds the full path of the scanned back side ID image. If this string is empty the internal image is used as the image source.
[in] **DestFileName** – Null terminated string that holds the full name of the destination image file that will contain the face image from the ID document.
[in] **stateId** – The state index value as defined in the idLibDef.bas file.

Return
If the function succeeds, the return value is **ID_TRUE.**

If the function fails, one of the following values is returned:

LICENSE_INVALID – Library was not initialized with proper license.

ID_ERR_FILE_OPEN – Failing to load source image (if SourceFileName is not empty)

INVALID_INTERNAL_IMAGE – No internal image is loaded. This value returns when attempting to use the internal image without scanning an image first.

ID_ERR_STATE_NOT_SUPORTED – The requested state is not supported.

ID_BAD_DESTINATION_FILE – Bad destination path (could not create the destination file).

ID_ERR_FILE_OPEN – Bad source image file (used only when using a file as the source image). This value returns if the source file is missing or cannot be accessed for reading.

INVALID_INTERNAL_IMAGE– Bad internal image (used only when extracting the face image from the image stored in the internal buffer). This value returns if there is no image in the buffer.

ID_FALSE– Internal processing error.

ID_ERR_FACE_IMAGE_NOT_FOUND– Returned when the analyzer cannot detect the face image in the driver's license image.

ID_ERR_CANNOT_DELETE_DESTINATION_IMAGE– Returned when a file with the same name as the destination file already exists and cannot be overwritten.

ID_ERR_CANNOT_COPY_TO_DESTONATION– Returned when the destination file cannot be opened for write on the disk**.**

Remarks

This function is for Duplex scan support that gives you the option to get the face image without knowing where it is located (front or back side of the card). Duplex scanning is available only with ScanShell® 800DX\800DXN\3100D\3100DN.

Use this function to extract the image rectangle of the person's face from the source ID image. The source image can be one of two:

Internal Image: The last scanned image (stored in the internal memory). This image will be used only if the **SourceFileName** string is empty.

External Image File: The full file name is given in **SourceFileName** parameter. If an external file is used as the source image, it must be a 24 bit image (true color) and have a resolution of 300dpi. The source and file destination can be one of the following formats: BMP, TIFF, JPG, PCX, TGA, PNG, PSD.

To set the image format, use the proper file extension (*xxx.bmp* for bitmap, *xxx.jpg* for Jpeg, etc.).

GetSignatureImageDuplex

Format

```
GetSignatureImageDuplex (SourceFileName As String,
BackSourceFileName As String, DestFileName As String, StateID As
```

Parameters

[in] **SourceFileName** – Null terminated string that holds the full path of the scanned ID image. If this string is empty, the internal image is used as the image source.

[in] **BackSourceFileName** – Null terminated string that holds the full path of the scanned back side ID image. If this string is empty, the internal image is used as the image source.

[in] **DestFileName** – Null terminated string that holds the full name of the destination image file that will contain the signature image from the ID document.

[in] **stateId** – The state index value as defined in the idLibDef.bas file.

Return

If the function succeeds, the return value is **ID_TRUE.**

If the function fails, one of the following values is returned:

LICENSE_INVALID – Library was not initialized with proper license.

ID_ERR_FILE_OPEN – Failing to load source image (if SourceFileName is not empty)
INVALID_INTERNAL_IMAGE – No internal image is loaded. This value returns when attempting to use the internal image without scanning an image first.
ID_ERR_STATE_NOT_SUPORTED – The requested state is not supported.
ID_BAD_DESTINATION_FILE – Bad destination path (could not create the destination file).
ID_ERR_FILE_OPEN – Bad source image file (used only when using a file as the source image). This value returns if the source file is missing or cannot be accessed for reading.
INVALID_INTERNAL_IMAGE– Bad internal image (used only when extracting the face image from the image stored in the internal buffer). This value returns if there is no image in the buffer.
ID_FALSE– Internal processing error.
ID_ERR_FACE_IMAGE_NOT_FOUND– Returned when the analyzer cannot detect the face image in the driver's license image.
ID_ERR_CANNOT_DELETE_DESTINATION_IMAGE– Returned when a file with the same name as the destination file already exists and cannot be overwritten.
ID_ERR_CANNOT_COPY_TO_DESTONATION– Returned when the destination file cannot be opened for write on the disk**.**

Remarks

This function is for Duplex scan support that gives you the option to get the face image without knowing where it is located (front or back side of the card). Duplex scanning is available only with ScanShell® 800DX\800DXN\3100D\3100DN.
Use this function to extract the image rectangle of the person's face from the source ID image. The source image can be one of two:
Internal Image: The last scanned image (stored in the internal memory). This image will be used only if the **SourceFileName** string is empty.
External Image File: The full file name is given in **SourceFileName** parameter. **I**f an external file is used as the source image, it must be a 24 bit image (true color) and have a resolution of 300dpi. The source and file destination can be one of the following formats: BMP, TIFF, JPG, PCX, TGA, PNG, PSD.
To set the image format, use the proper file extension (*xxx.bmp* for bitmap, *xxx.jpg* for Jpeg, etc.).

DetectProcessAndCompareEX

Format

```
DetectProcessAndCompareEX (ImageA As String, ImageB As String, state As Integer, ImageA_assignment As Integer, angleA As Integer, angleB As Integer, reserved As Integer) As Long
```

Parameters

[in] **ImageA** – Full name of the source driver's license image file (Side A).
[in] **ImageB** – Full name of the source driver's license image file (Side B).
[in] **State** – StateID can be -1 for auto detect state.
[in,out] ImageA_assignment.
[in,out] angleA – Will contain the angle that has been used to rotate image A.
[in,out] angleB – Will contain the angle that has been used to rotate image B.
[in] **reserved** – Use the value 0.

Return

If the function succeeds, the return value is **the stateID.**
If the function fails, the value **-1** is returned.

Remarks
Use this function to detect and process a card with two sides that each of the sides can contain text and\or barcode data. After using this function, you can get the fields data from the idData and Barcode classes (Using the ScanShell® 3000D\ScanShell® 800DX only).
To use the internal image (after scanning a card), *ImageA* and *ImageB* should be empty strings.

DetectProcessAndCompareEX2

Format

```
DetectProcessAndCompareEX2 (ImageA As String, ImageB As String, state As
Integer, ImageA_assignment As Integer, angleA As Integer, angleB As Integer,
defualtBarcode As Integer, reserved As Integer) As Long
```

Parameters
[in] **ImageA** – Full name of the source driver's license image file (Side A).
[in] **ImageB** – Full name of the source driver's license image file (Side B).
[in] **State** – StateID can be -1 for auto detect state.
[in,out] ImageA_assignment.
[in,out] angleA – Will contain the angle that has been used to rotate image A.
[in,out] angleB – Will contain the angle that has been used to rotate image B.
[in] defaultBarcode – This value will tell the function to use the 2D or 1D barcode as the first extraction process. 0=2D 1=1D
[in] **reserved** – Use the value 0.

Return
If the function succeeds, the return value is **the stateID.**
If the function fails, the value **-1** is returned.

Remarks
Use this function to detect and process a card with two sides that each of the sides can contain text and\or barcode data. After using this function, you can get the fields data from the idData and Barcode classes (Using ScanShell® 3000D\ScanShell® 800DX only).
To use the internal image (after scanning a card), *ImageA* and *ImageB* should be empty strings.

DetectProcessDuplex

Format

```
DetectProcessDuplexEX(ImageA As String, ImageB As String, state As
Integer, ImageA_assignment As Integer, angleA As Integer, angleB As Integer, reserved
As Integer) As Long
```

Parameters
[in] **ImageA** – Full name of the source driver's license image file (Side A).
[in] **ImageB** – Full name of the source driver's license image file (Side B).
[in] **State** – StateID can be -1 for auto detect state.
[in,out] ImageA_assignment.
[in,out] angleA – Will contain the angle that has been used to rotate image A.
[in,out] angleB – Will contain the angle that has been used to rotate image B.
[in] **reserved** – Use the value 0.

Return

If the function succeeds, the return value is **the stateID.**
If the function fails, the value **-1** is returned.

Remarks
Use this function to detect the front side and to extract the data from the two sides of a card. After using this function, you can get the fields value from the idData class (Using ScanShell® 3000D\ScanShell® 800DX only).
To use the internal image (after scanning a card), *ImageA* and *ImageB* should be empty strings.

ProcMRZ

Format

```
ProcMRZ (FileName As String, rotationAngle As Integer) As Long
```

Parameters
[in] **FileName** – Full name of the source driver's license image file (Side A).
[in,out] **rotationAngle** – Will contain the angle that has been used to rotate the image.

Return

If the function succeeds, the return value is **1.**
If the function fails, the value **-1** is returned.

Remarks
Use this function to scan a full image and extract only the MRZ as raw data.
To use the internal image (after scanning a card), *FileName* should be an empty string.

ResetIDFields

Format

```
ResetIDFields()
```

Parameters
None.

Return

None.

Remarks
Use this function to clear data from all the fields in the idData class.

GetCountryIDByStateID

Format

```
GetCountryIDByStateID (StateID As Integer)
```

Parameters

[in] **StateID** – Integer value of the StateId.

Return

Integer value: Country ID.

Remarks

Use this function to get the country ID by a given state ID.

ActivateDmAuthentication

Format

```
ActivateDmAuthentication (Activate as BOOLEAN)
```

Parameters
[in] **Activate** – Boolean value to activate\deactivate watermark detection mechanism.

Return

ID_TRUE (1): Operation completed successfully.
ID_ERR_DM_LIBRARY_NOT_FOUND (-24): Could not find Digimarc™ files in the application folder.

Remarks
Activating the watermark detection, attempts to load and initialize the Digimarc™ library for watermark detection and verification. This library is installed in the SDK folder if the proper checkbox is selected while installing the SDK. The mechanism verifies the content of the following fields:

- Driver license number
- Jurisdiction (Issuing State)
- Date of birth
- Issue date

Please call this function only once (normally in the after initializing idCard library by calling *ScanW.IdCard.InitLibrary()*).

Important: It is important to locate the SDK files (and the Digimarc™ files) were the main application executable reside.

Once the mechanism is activated, the Driver license SDK will automatically verify each of the four fields after the text extraction is done. The result of this verification can be retrieved by reading the relevant properties (*LicenseAuthentication, StateAuthentication*, etc.).

RefreshAuthenticationStatus

Format

```
RefreshAuthenticationStatus ()
```

Parameters

None.

Return

None.

Remarks
This function refreshes the values of the verification properties. Call this function before reading these values of the properties (*LicenseAuthentication, StateAuthentication*, etc.) to get the current reading verification results.

GetFaceImageEx

Format

```
GetFaceImageEx (Source As String, Dest As String, idState As Long,
ImageType As Long) As Long
```

Parameters
[in] szSourceFile – Full name of the source driver's license image file.
[in] szDestFile – Full name of the destination face image.
[in] stateID – The state ID.
[in] ImageType – The face image type to extract.

Return

ID_ERR_FILE_OPEN: Cannot open input image file.
INVALID_INTERNAL_IMAGE: Invalid internal image file.
ID_FALSE: Image processing failed.
ID_ERR_CANNOT_DELETE_DESTINATION_IMAGE: Destination file already exists and cannot be overwritten.
ID_ERR_CANNOT_COPY_TO_DESTONATION: Copying face image file to destination file failed.
ID_ERR_FACE_IMAGE_NOT_FOUND: Extraction of the face image failed – could not locate the face rectangle.
ID_TRUE: Function completed successfully.

Remarks
This function is similar to the GetFace function but can extract more than one face image from the card. The imageType value can be 0 to get the regular face image, or 1 to get face image with the background. This function with ImageType other than a 0 value, will work only for supported cards. For now only “Spain Police” cards are supported by this function.

GetMRZChecksumVerified

Format

```
Int GetMRZChecksumVerified (int FieldIndex)
```

Parameters
[in] FieldIndex – Constant value of the field.

Return value

1 = Verification success.
0 = Verification failed.
-1 = Not been verified.

Remarks
Use this function to verify the field value with its checksum digit on the MRZ line (where it was taken from).

GetFaceAndSignatureImagesOnly

Format

```
GetFaceAndSignatureImagesOnly (Source As String, FaceDest As String,
SignatureDest As String, idState As Long, ImageType As Long) As Long
```

Parameters
[in] szSourceFile – Full name of the source driver's license image file.
[in] szFaceDest – Full name of the destination face image.
[in] szSignatureDest – Full name of the destination signature image.
[in] stateID – The state ID.
[in] ImageType – The face image type to extract.

Return

ID_ERR_FILE_OPEN: Cannot open input image file.
INVALID_INTERNAL_IMAGE: Invalid internal image file.
ID_FALSE: Image processing failed.
ID_ERR_CANNOT_DELETE_DESTINATION_IMAGE: Destination file already exists and cannot be overwritten.
ID_ERR_CANNOT_COPY_TO_DESTONATION: Copying face image file to destination file failed.
ID_ERR_FACE_IMAGE_NOT_FOUND: Extraction of the face image failed – could not locate the face rectangle.
ID_TRUE: Function completed successfully.

Remarks
This function is similar to the GetFace function but will extract the face and signature images. The imageType value can be 0 to get the regular face image, or 1 to get face image with the background. This function with ImageType other than a 0 value, will work only for supported cards. For now only "Spain Police" cards are supported by this function.

1.1.29 SetBrightnessForFaceImage

Format

SetBrightnessForFaceImage (int nBrightness)

Parameters
[in] nBrightness – – a number between 0 – 100 , to set the brightness.

Remarks
This function will set the brightness of the face image returned by any call to get the face image later on.

1.1.30 SetFieldToBeErasedFromImage

Format

void SetFieldToBeErasedFromImage(int nFieldIndex,bool bErase)

Parameters
[in] nFieldIndex – The index of the field to be erased. Should be corresponding to the field number as in IDcardExp.h in the SDK folder.
[in] bErase – True if you want to add this field to the list of erased fields , or false to remove it from the list.

Remarks
When called before Id card processing this function will cause the indicated fields to be erased from the image.

1.1.29 GetFrontBackData

Format

```
void GetFrontBackData(int iFrontBack, int& iTemplate, int& iMRZ, int& iBarCode)
```

Parameters

[in] iFrontBack – 0=Front, 1=Back
[out] iTemplate
[out] iMRZ
[out] iBarCode
Each one of the 3 parameters will be set by the SDK to one of the following values:
FrontBack_Data_Unknown = 0 - Default
FrontBack_Data_Always_Exist = 1
FrontBack_Data_Sometimes_Exist = 2
FrontBack_Data_Dont_Exist = 3

Remarks
Call this function after processing it to know what exist on the front/back side of the processed card.

IdData Library Properties

IdCountry
Attribute: Read only
Value: Returns the full name of the country that issued the document.

CountryShort
Attribute: Read only
Value: Returns the name abbreviation of the country that issued the document.

Original
Attribute: Read only
Value: Returns general data field from the document.

Address6
Attribute: Read only
Value: Returns general data field from the document.

DateOfBirth4
Attribute: Read only
Value: Returns the date of birth with year in 4 digits format.

ExpirationDate4
Attribute: Read only
Value: Returns expiry date with year in 4 digits format.

IssueDate4
Attribute: Read only
Value: Returns issue date with year in 4 digits format.

GeneralWaterMark
Attribute: Read only
Type: Integer
Value: Returns the status of the general watermark indication on the card. If this value indicates an error, than the values of the tested fields are not valid.
DM_SUCCESS (1): Watermark exists on the card and read successfully.
DM_ERROR_DEBUGGER_PRESENT (-24): Cannot continue to process watermark because of a debugger presence (security issue). This is not an error as the detection will work normally once the debugger is turned off.
DM_ERROR_LIB_NOT_FOUND (-1): Could not find Digimarc™ library files.
DM_ERROR_BAD_IMAGE_NAME (-2): Internal error.
DM_ERROR_WM_NOT_FOUND (-3): Could not find watermark indication on the driver license card.
DM_ERROR_WM_NOT_SUPPORTED (-4): Watermark is not supported on this driver license card.
DM_ERROR_FAIL_TO_GET_IMAGE_HANDLE (-23): Internal error.
DM_ERROR_FAIL_TO_LOAD_IMAGE (-22): Internal error.

IssueDateAuthentication
Attribute: Read only

Type: Integer
Value: This property reports the verification results of the Issue Date field. The content of this property is valid only if the property ***GeneralWaterMark*** is **DM_SUCCESS**. The property may hold one of the following values:
DM_SUCCESS (1): The field content was authenticated and matches the field value embedded in the watermark.
DM_FAILED (0): The field content was NOT authenticated and does not match the field value embedded in the watermark.
DM_ERROR_WM_NOT_FOUND (-3): Digimarc™ library is missing and therefore the authentication failed.

DobDateAuthentication
Attribute: Read only
Type: Integer
Value: This property reports the verification results of the Date of Birth field. The content of this property is valid only if the property ***GeneralWaterMark*** is **DM_SUCCESS**. The property may hold one of the following values:
DM_SUCCESS (1): The field content was authenticated and matches the field value embedded in the watermark.
DM_FAILED (0): The field content was NOT authenticated and does not match to the field value embedded in the watermark.
DM_ERROR_WM_NOT_FOUND (-3): Digimarc™ library is missing and therefore the authentication failed.

LicenseAuthentication
Attribute: Read only
Type: Integer
Value: This property reports the verification results of the License field. The content of this property is valid only if the property ***GeneralWaterMark*** is **DM_SUCCESS**. The property may hold one of the following values:
DM_SUCCESS (1): The field content was authenticated and matches the field value embedded in the watermark.
DM_FAILED (0): The field content was NOT authenticated and does not match the field value embedded in the watermark.
DM_ERROR_WM_NOT_FOUND (-3): Digimarc™ library is missing and therefore the authentication failed.

StateAuthentication
Attribute: Read only
Type: Integer
Value: This property reports the verification results of the State field. The content of this property is valid only if the property ***GeneralWaterMark*** is **DM_SUCCESS**. The property may hold one of the following values:
DM_SUCCESS (1): The field content was authenticated and matches the field value embedded in the watermark.
DM_FAILED (0): The field content was NOT authenticated and does not match to the field value embedded in the watermark.
DM_ERROR_WM_NOT_FOUND (-3): Digimarc™ library is missing and therefore the authentication failed.

DocType
Attribute: Read only
Value: Returns template name of the last processed image.

NameFirst_NonMRZ
Attribute: Read only
Value: Returns first name data from the document and not from the MRZ lines.

NameMiddle_NonMRZ
Attribute: Read only
Value: Returns middle name data from the document and not from the MRZ lines.

NameLast_NonMRZ
Attribute: Read only
Value: Returns last name data from the document and not from the MRZ lines.

NameSuffix_NonMRZ
Attribute: Read only
Value: Returns suffix name data from the document and not from the MRZ lines.

NameLast1
Attribute: Read only
Value: Returns last name1 data from the document (Mostly used in Spain cards that contain more than one last name).

NameLast2
Attribute: Read only
Value: Returns last name2 data from the document (Mostly used in Spain cards that contain more than one last name).

DatesFormat
Attribute: val – Needed date format type.
Can be one of these values:
EXTRACT_DATE_FORMAT_NONE = 0 (Use the default date extraction)
EXTRACT_DATE_FORMAT_MDY = 1 (mm-dd-yy)
EXTRACT_DATE_FORMAT_DMY = 2 (dd-mm-yy)
EXTRACT_DATE_FORMAT_YMD = 3 (yy-mm-dd)
EXTRACT_DATE_FORMAT_YDM = 4 (yy-dd-mm).
Set this property to extract the date's fields with the format you need. This property will take effect on passport dates fields as well.

MotherName
Attribute: Read only
Value: Returns Mother Name data from the document.

FatherName
Attribute: Read only
Value: Returns Father Name data from the document.

IssueDateLocal
Attribute: Read only
Value: Returns Issue Date Local data from the document (Issue date in a local format or language).

DateOfBirthLocal
Attribute: Read only

Value: Returns Date of Birth Local data from the document (Date of Birth in a local format or language).

Nationality
Attribute: Read only
Value: Returns Nationality data from the document.

PlaceOfIssue
Attribute: Read only
Value: Returns Place of Issue data from the document.

PlaceOfBirth
Attribute: Read only
Value: Returns Place of Birth data from the document.

Library SLibEx: General Functionality

SLibEx library extends the function set introduced in the SLibEx library of the ScanW wrapper library. The library exports additional methods to control the scanner functionality.

SLibEx Library Functions

UnInit

Format

```
UnInit () As Long
```

Parameters
None.
Return

SLIB_UNLOAD_FAILED_BAD_PARENT: Cannot unload the driver since another application is using/ has loaded it.
SLIB_NOT_INITILIZED: The driver is not found in the memory (hence cannot be unloaded).
SLIB_ERR_NONE– Library unloaded successfully.

Remarks
The scanner library (and all other SDK libraries) can serve a single application at a time. If you wish to run another application (that uses the SDK), you must first unload the SDK from the memory using this function. If you fail to do so, you might run into a situation that both applications will attempt to access the scanner (and other SDK resources) – an operation that might hang the application.

Use this function to unload the scanner driver from the memory (hence, to reverse the operation of the function *InitLibrary)* before initializing the SDK from the other application.

Note: See the function *InitLibrary* in Section 1.1.1 in the document ScanW.pdf).

InitEx

Format

InitEx (**license** As String, **OwnerId** As Interger, **OwnerName** As String) As Long

Parameters
[in] **license** – A 16 character SDK license key string.
[in] **Ownweld** – A unique number that identifies the application that uses the SDK on the local machine. The number must be in the range >100 as the value range 0-100 is reserved.
[in] **OwnerName** – A string that contains the name of the application that is using the SDK.

Return

LICENSE_VALID: The library initialized successfully and is ready to be used.
LICENSE_INVALID: The library failed to initialize as the license is invalid. All scanner operations are disabled.
LICENSE_EXPIRED: The library failed to initialize as the license has expired. All scanner operations are disabled.
LICENSE_DOES_NOT_MATCH_LIBRARY: The library failed to initialize as the license is invalid for this library. All library operations are disabled.
SLIB_ERR_SCANNER_NOT_FOUND: The library initialized successfully but the scanner was not found.
SLIB_ERR_DRIVER_NOT_FOUND: The library initialized successfully but the scanner was not found.
SLIB_LIBRARY_ALREADY_INITIALIZED: The library was already initialized successfully by a previous call to *InitLibraryEx*,. Hence, this call is ignored.
SLIB_LIBRARY_ALREADY_ USED_BY_OTHER_APP: The function failed to initialize since the library is already loaded and owned by another application.

Remarks
This function initializes the scanner library, and, if successful, grants the calling application the ownership to the SDK functionality. Once the SDK ownership is established, the function attempts to load the scanner driver and communicate to the scanner (if connected). If the application tries to call this function while another application is already using the SDK (i.e., owns the SDK), the function will return **SLIB_LIBRARY_ALREADY_ USED_BY_OTHER_APP** as only one application may control the SDK at a time. In such a case, the user must terminate the other application execution before attempting to initialize this library again. The application may retrieve the current SDK owner information from the properties *OwnerId* and *OwnerName* in the SLibEx library.
The application can assume that it gained ownership on the SDK only if the returned value is:
LICENSE_VALID
SLIB_LIBRARY_ALREADY_INITIALIZED
SLIB_ERR_SCANNER_NOT_FOUND
SLIB_ERR_DRIVER_NOT_FOUND
If another value is returned, you will not be able to use the SDK until you have fixed the problem.
If the scanner is not attached to the PC while calling this function, the application will be able to load the scanner by reading the property *IsScannerValid.*

Note:
This function extends the functionality function *InitLibrary* (in Section 1.1.1 in the document ScanW.pdf) and can be used instead.
The *OwnerId* number may be any number in the range >100 as the number range 0-100 is reserved for Card Scanning Solutions applications.

UnInitEx

Format

```
InitEx (OwnerId As Interger) As Long
```

Parameters

[in] **OwnweId** – A unique number that identifies the application.

Return

SLIB_ERR_NONE: The SLib library was un-initialized successfully and the scanner driver was unloaded from the memory.
SLIB_NOT_INITILIZED: The function failed since the library is not in initialized status.
SLIB_UNLOAD_FAILED_BAD_PARENT: The function failed since the parameter *OwnerId* is not the value of the current library owner.

Remarks

This function un-initializes the scanner library, and if successful, performs the following:

- Releases the library from the current owner application.
- Unloads the scanner driver from the memory.

OwnerData

Format

```
OwnerData (data As String)
```

Parameters

[in] data – Null terminated string that holds the owner application name.

Remarks

Returns the current application name that initialized the SDK using *InitEx* function.

KillSnapServer

Format

```
KillSnapServer ()
```

Parameters

None

Remarks

Calling this function will stop and close the SnapServer service that is used to use the SnapShell® camera. Call this function only after all the commands are done to close the application. This is the last function you should call when closing the application.

get_TotalConnecetedScanners

Format

```
get_TotalConnecetedScanners ()
```

Parameters

None.

Return

The number of all connected SnapShell® cameras.

Remarks

Retrieves the total number of the connected SnapShell® cameras.

AssignScanner

Format

```
AssignScanner (ScannerIndex As Integer)
```

Parameters

[in] – Use -1 to get the USB port number of the next SnapShell® camera.

Return

The number of all connected SnapShell® cameras.

Remarks

Retrieves the total number of the connected SnapShell® cameras.

ReleaseScanner

Format

```
ReleaseScanner (ScannerIndex As Integer)
```

Parameters

[in] – Use -1 to release all SnapShell® cameras.

Return

None.

Remarks

Use a loop to call this function to release each SnapShell® device or use the KillSnapServer function to release all the SnapShell® devices at once.

SetCurrentScannerIndex

Format

```
SetCurrentScannerIndex (ScannerIndex As Integer)
```

Parameters

[in] –SnapShell® device index as it returned by the function AssignScanner.

Return

None.

Remarks

Set the active SnapShell® device at a specific USB port.

GetCurrentScannerIndex

Format

```
GetCurrentScannerIndex ()
```

Parameters

None.

Return

The active USB port number of the current active SnapShell® device.

Remarks

Get the active SnapShell® device USB port.

GetFirstTwainSource

Format

```
GetFirstTwainSource (ScannerName As String)
```

Parameters

[In-out] ScannerName as string, buffer to get the first twain scanner name that is found.

Return

SLIB_TRUE - Successfully completed.
SLIB_FALSE - Function failed to find scanner.
SLIB_ERR_NO_NEXT_VALUE - No more twain scanners found.
SLIB_ERR_NO_TWAIN_INSTALLED – No twain devices found.

Remarks

Use this function to detect and get the first twain scanner that connects to your system. If this function returns SLIB_ERR_NO_NEXT_VALUE, then no more twain scanners have been detected. If the return value is SLIB_TRUE, then there are more twain scanners found and you can call the GetNextTwainSource function to get the entire twain scanners list.

GetNextTwainSource

Format

```
GetNextTwainSource (ScannerName As String)
```

Parameters

[In-out] ScannerName as string, buffer to get the next twain scanner name that is found.

Return

SLIB_TRUE - Successfully completed.
SLIB_ERR_NO_NEXT_VALUE - No more twain scanners found.

Remarks

Use this function to detect and get the next twain scanner that is connected to your system. If this function returns SLIB_ERR_NO_NEXT_VALUE, then no more twain scanners are detected. If the return value is SLIB_TRUE, then there are more twain scanners found and you can call this function again to get the entire twain scanners list.

GetTwainScanner

Format

```
GetTwainScanner (ScannerName As String)
```

Parameters

[In-out] ScannerName as string, buffer to get the current connected twain scanner name that is found.

Return

SLIB_TRUE - Successfully completed.
SLIB_ERR_NO_NEXT_VALUE - Function failed to find scanner.

Remarks

Use this function to get the current twain scanner that is connected to your system.

SetTwainScanner

Format

```
SetTwainScanner (ScannerName As String)
```

Parameters

[In] ScannerName as string.

Return

SLIB_TRUE - Successfully completed.
SLIB_FALSE - Function failed to find a scanner.

Remarks

Use this function to set the current twain scanner that is connected to your system. Use this function before you initialize the library.

CalibrateScannerEx

Format

```
CalibrateScannerEx ()
```

Return value

Void.

Remarks

This function calibrates the scanner using the calibration card similar to the CalibrateScanner function in the ScanW.pdf but this function will display the calibration progress bar dialog. 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.

GetSnapSerial

Format

```
int GetSnapSerial (out string SSnapSerial)
```

Parameters

pSerial – A buffer to contain the serial number.
nBufLen – The length of the buffer to contain the number.

If the size of nBufLen is smaller than the length of the string to be returned, then nothing is copied to pSerial.

Return

SLIB_TRUE - Successfully completed.
SLIB_FALSE - Function failed to find a scanner.

Remarks

This one returns the serial number of the SnapShell® scanner. (If the SDK is connected to a SnapShell® scanner).

SetExternalTriggerMode

Format

```
int SetExternalTriggerMode (int bVal);
```

Parameters

int bVal – 0 Cancel trigger mode. 1 – Set trigger mode.

Return

SLIB_TRUE - Successfully completed.
SLIB_FALSE - Function failed to find scanner.

Remarks

This one sets the mode of external trigger. True / False.
While in external trigger mode and working with a SnapShell® scanner, the scanning operation will not be finished until the function SetExternalTrigger is called. (The light on the scanner will not turn green again).
Returns none zero if successful.

SetExternalTrigger

Format

```
int SetExternalTrigger();
```

Remarks

Call this one to finish the scanning process, after ScanBmpFile / ScanBmpFileEx when in external trigger mode. A successful call will finish the scanning process and turn the scanner light back to green.

SetExternalOperation

Format

```
int SetExternalOperation();
```

Remarks

Call this to initiate a “fake” scan on the SnapShell® scanner when in external trigger mode. A call will turn the scanner light to blue, until another call to SetExternalTrigger will be made.

SLibEx Library Properties

CalibrationThreshold

Attribute: Read\Write
Value: Returns\Gets the Calibration Threshold value.
This value will affect the Is Need Calibration function.
Accepted values: 1 - 100, where 1 is the most sensitive and 100 is the least sensitive.

Default value: 60

Duplex
Attribute: Read\Write
Value: Returns\Gets the Duplex value.
Setting this value activates the double side scan when using scanner models ScanShell® 3000D\ScanShell® 800DX\ScanShell® 800DXN.

InOutScan
Attribute: Read\Write
Value: Returns\Gets the InOutScan value.
Setting this value causes the ScanShell® 3000D\ScanShell® 800DX\ScanShell® 800DXN to eject the scanned document in an opposite direction to the scan feed direction. This option is used to scan documents that cannot be scanned in a feed-through manner such as passports.

DefaultScanner
Attribute: Write
Value: Sets the **DefaultScanner** value.
Setting this value will set the SDK to load faster because the SLib will try to initialize the scanner type that has been set in this value first.

UseFixedModel
Attribute: Write
Value: Sets the **UseFixedModel** value.
Setting this value will set the SDK to work only with the given scanner type.

SetCenteredImage
Attribute: Write
Value: Sets the SetCenteredImage value.
Sets the alignment of the captured image to be centered or not.
SetRemoteIP
Attribute: Write
Value: Sets the remote IP address, port number and the remote scanner model.
This function enables the usage of a remote IP scanner on a different machine.
The remote machine should have IpScan installed and running.
Once the client program is running, you can setup the port it is listening to (from the tray icon choose "configuration"", if a file named IpScan.ini exists in the same folder as the .exe file then the port number will be stored in this file) and use SetRemoteIP, before initializing the library.
If the remote IP address parameter is USE_REMOTE_DESKTOP, then the SDK will use the IP address of the remote desktop client (If a remote desktop session is active).

IsRemoteScannerValid

Attribute: Read
The function returns the scanner type connected to IpScan or -1 when the scanner is not connected.
Use this function to test if the scanner connected to IpScan is valid. This function does not test the status of the communication with IpScan, for that use IsScannerValid().

GetRemoteScannerVersion

Attribute: Read

The function returns the version of the remote IpScan software or -1 if not connected.

ScanBmpFileAsync

The procedure is relevant to IpScan only. Call this function to scan an image file without waiting for the image to be transferred back to ScanW.
ScanBmpFileAsync will return immediately when the scan is finished. You can call this function again before the file arrives to the calling machine.

nDataPortNumber – Specify a port number on which the data will be transferred or -1 if you want to use the same port that was used for opening the connection.

nThumbImageResolution – If not set to -1, IpScan will send a small resolution image before the full size image that can be used for display.

nThumbImageJpgQuality – Sets the JpgQuality for the thumb.

GetFirstJob

Attribute: Read
The function returns the number of the first job in the queue or -1 if there are no jobs in the queue.

GetNextJob

Attribute: Read
The function returns the number of the next job in the queue or -1 if there are no jobs in the queue.

GetJobStatus

Attribute: Read/Write
The function returns the status of the supplied job number.
The status can be one of the following:
JOB_PENDING =0, JOB_DONE =1, JOB_THUMB_READY = 2, JOB_NOT_FOUND =3, JOB_TIMEOUT =4, JOB_FAILED =5.

If the status is JOB_DONE, JOB_TIMEOUT or JOB_FAILED it will be automatically be deleted from the job list after this call.

CountCSSNDevices
Attribute: Read
The function returns the number of CSSN devices connected to the machine or to the remote machine.
GetCSSNDevice
Attribute: Read
The function copies the name of the CSSN device connected to the machine or to the remote machine.

nIndex – The index of the device.
pStr – Pointer to a null terminated string that will contain the device name.
nBufLen – The size of the buffer pStr.

If the size of nBufLen is smaller than the length of the string to be returned, then nothing is copied to pStr.

DeviceSerialNumber
Attribute: Read
Returns the serial number of the scanning device. This property is supported only for the SnapShell® family. This property is set to 0 if the serial number cannot be read from the hardware.

MagLib Library Functions

ReleasePort

Format

```
ReleasePort()
```

Parameters
None.
Return
None.

Remarks
Releases the USB port and unutilizes the magnetic library.

ProcessEx

Format

```
ProcessEx (strData As String)
```

Parameters
strData – String buffer of the raw data.
Return
LONG_AAMVA: Standard AAMVA format (includes channel1, channel2 and channel3).
SHORT_AAMVA: Short AAMVA format (includes channel1 and channel3).
OLD_CA_DMV: Old DMV format (California).
OLD_LA_DMV: Old DMV format (Louisiana).
UNKNOWN_FORMAT: Unknown format. In such a case, no further processing is done.

Remarks
Call this function to process the given raw data. The raw data is scanned for format detection. If a specific format is detected, the data is parsed further and loads the library properties.

SetSwipePortEx

Format

```
SetSwipePortEX (portNum As Integer)
```

Parameters
[in] – portNumber - the USB port number to use.

Return

None.

Remarks

Sets the USB port number.

MagLibEx Library Properties

NameSuffix

Attribute: Read

Value: Returns the Suffix name in string format.

CPassport Library Functions

Library CPassport: General Functionality

CPassport library extends the function set introduced in the CPassport library of the ScanW wrapper library. The library exports additional methods to control the scanner functionality

GetFaceEx

Format

```
GetFaceEx (sourceFile As String, destFile As String) As
Long
```

Parameters

sourceFile: Full file name of the passport source file (including path).

destFile: Full file name of the face destination file (including path).

Return

PASS_ERR_NONE: Face extracted successfully.

PASS_ERR_CANNOT_DELETE_DESTINATION_IMAGE: Cannot create file since there is already an existing file with such name with a Read-Only attribute.

PASS_ERR_CANNOT_COPY_TO_DESTONATION– Copying the face image to the destination drive failed.

PASS_ERR_FACE_IMAGE_NOT_FOUND– Could not locate the face image in the document.

Remarks

Use this function to extract the face image from an existing passport image. The source image must be 3”x 5”, 300dpi, true color (24 bit).

GetSignature

Format

```
GetSignature (sourceFile As String, destFile As String) As
.
```

Parameters

sourceFile: Full file name of the passport source file (including path).

destFile: Full file name of the signature destination file (including path).

Return

PASS_ERR_NONE: Face extracted successfully.

PASS_ERR_CANNOT_DELETE_DESTINATION_IMAGE: Cannot create file since there is already an existing file with such name with a Read-Only attribute.

PASS_ERR_CANNOT_COPY_TO_DESTONATION– Copying the signature image to the destination drive failed.

PASS_ERR_FACE_IMAGE_NOT_FOUND– Could not locate the signature image in the document.

Remarks

Use this function to extract the signature image from an existing passport image. The source image must be 3"x 5", 300dpi, true color (24 bit).

GetFaceAndSignatureImagesOnly

Format

```
GetFaceAndSignatureImagesOnly (Source As String, FaceDest As String,
SignatureDest As String) As Long
```

Parameters

[in] szSourceFile – Full name of the source driver's license image file.
[in] szFaceDest – Full name of the destination face image.
[in] szSignatureDest – Full name of the destination signature image.

Return

PASS_ERR_FILE_OPEN: Cannot open input image file.
INVALID_INTERNAL_IMAGE: Invalid internal image file.
PASS_FALSE: Image processing failed.
PASS_ERR_CANNOT_DELETE_DESTINATION_IMAGE: Destination file already exists and cannot be overwritten.
PASS_ERR_CANNOT_COPY_TO_DESTONATION: Copying face image file to destination file failed.
PASS_ERR_FACE_IMAGE_NOT_FOUND: Extraction of the face image failed – could not locate the face rectangle.
PASS_ERR_NONE: Function completed successfully.

Remarks

This function is similar to GetFaceEx function but will extract the face and signature images.

GetFaceImageBuffer

Format

```
GetFaceImageBuffer ( Source As String, buffer As String, FileType As
String) As Long
```

Parameters

[in] **Source** – Name of the source passport image file.
[in] **buffer** – A string buffer that will contain the image content upon successful return. FileType.
[in] **FileType** – Set the buffer image format. May be one of the following strings (case insensitive):
"BMP"
"JPG"
"PNG"
"TIF"
"TGA"

"PSD"
"PCX"

Return

If the function succeeds, the return value is **IMG_ERR_SUCCESS.**

If the function fails, the value **BAD_IMAGE_DUMP** is returned**.**

Remarks

Use this function to extract the face image from the document image to the given buffer. The source document image can be an existing image file (in BMP format) or the last scanned document (stored in the internal image buffer). If the parameter **SourceFile** is an empty string, then the image is taken from the internal image buffer. If this parameter contains the full name of a valid image file, then this file is used.

GetSignatureImageBuffer

Format

```
GetSignatureImageBuffer (Source As String, buffer As String, FileType As
String) As Long
```

Parameters

[in] **Source** – Name of the source passport image file.
[in] **buffer** – A string buffer that will contain the image content upon successful return. FileType.
[in] **FileType** – Set the buffer image format. May be one of the following strings (case insensitive):
"BMP"
"JPG"
"PNG"
"TIF"
"TGA"
"PSD"
"PCX"

Return

If the function succeeds, the return value is **IMG_ERR_SUCCESS.**

If the function fails, the value **BAD_IMAGE_DUMP** is returned**.**

Remarks

Use this function to extract the Signature image from the document image to the given buffer. The source document image can be an existing image file (in BMP format) or the last scanned document (stored in the internal image buffer). If the parameter **SourceFile** is an empty string, then the image is taken from the internal image buffer. If this parameter contains the full name of a valid image file, then this file is used.

ResetPassportData

Format

```
ResetPassportData ()
```

Parameters

None.

Return

None.

Remarks

Use this function to clear data from all the fields in the passport class.

GetPassportMRZChecksumVerified

Format

```
Int GetPassportMRZChecksumVerified (int FieldIndex)
```

Parameters

[in] FieldIndex – Constant value of the field.

Return Value

1 = Verification success.
0 = Verification failed.
-1 = Not been verified.

Remarks

Use this function to verify the field value with its checksum digit on the MRZ line (where it was taken from).

CPassport Library Properties

Address2
Attribute: Read only
Value: Returns the first line of the MRZ zone.

Address3
Attribute: Read only
Value: Returns the second line of the MRZ zone.

IssueDate
Attribute: Read only
Value: Returns the Issue date if the passport is supported.

End_POB
Attribute: Read only
Value: Returns the place of birth if the passport is supported.

NameFirstNoneMRZ
Attribute: Read only
Value: Returns the first name from the passport and not from the MRZ.

NameLastNoneMRZ
Attribute: Read only
Value: Returns the last name from the passport and not from the MRZ.

DateOfBirth4
Attribute: Read only
Value: Returns the date of birth with year in 4 digits format.

ExpirationDate4
Attribute: Read only
Value: Returns expiry date with year in 4 digits format.

IssueDate4
Attribute: Read only
Value: Returns issue date with year in 4 digits format.

Library CBarcode: General Functionality

The library CBarcode in ScanWex expands the barcode image processing capabilities that were introduced in the ScanW library. This library is capable of analyzing 1D barcode in addition to the 2D PDF417 barcode in the ScanW library.

CBarcode Library Functions

Proc1DImage

Format

```
Proc1DImage (ImageFileAs String) As Long
```

Parameters
ImageFile: Empty string (Reserved).

Return
LICENSE_INVALID – Library was not initialized with a proper license.
SLIB_ERR_SCANNER_NOT_FOUND – No attached scanner was found.
SLIB_ERR_INVALID_SCANNER – No scanner was found attached to the PC.
INVALID_INTERNAL_IMAGE - Cannot process the image as the internal image buffer is empty or invalid.
BC_ERR_NO_BC_FOUND – Could not find 1D barcode printing on the image.
BC_ERR_NONE – Barcode detected successfully.

Remarks
Use this function to detect the data encapsulated in the 1D barcode. The function decodes the barcode and stores the data in the property *Data1D.* If the function failed to process the barcode, the content of the property *Data1D* is set to an empty string.

CBarcode Library Properties

Data1D
Attribute: Read only
Value: Returns the content of the recently processed 1D barcode.
Hair
Attribute: Read only
Value: Returns the Hair value in the 2D barcode data.
Eyes
Attribute: Read only
Value: Returns the Eyes value in the 2D barcode data.
Height
Attribute: Read only
Value: Returns the Height value in the 2D barcode data.
Weight
Attribute: Read only
Value: Returns the Weight value in the 2D barcode data.

Endorsements
Attribute: Read only
Value: Returns the Endorsements value in the 2D barcode data.

Address2
Attribute: Read only
Value: Returns the Address2 value in the 2D barcode data.

City2
Attribute: Read only
Value: Returns the City2 value in the 2D barcode data.

State2
Attribute: Read only
Value: Returns the State2 value in the 2D barcode data.

Zip2
Attribute: Read only
Value: Returns the Zip2 value in the 2D barcode data.

Using4DigitsYearFormat
Attribute: Read only
Value: Set this value to true to get the year in dates with 4 digits. If set to false, than 2 digits will be used.

Library CImage: General Functionality

The library CImage in ScanWex expands the image manipulating capabilities that were introduced in the ScanW library. This library is capable of converting image formats and manipulates the internal image acquired by the scanner.

CImage Library Functions

GetImageDimensions

Format

```
GetImageDimensions (filename As String, Width As Integer, Height As Integer)
```

Parameters
fileName: Image file name or an empty string if referring to the internal image.
Width: Returns the width of the image (in pixels).
Height: Returns the height of the image (in pixels).

Return
IMG_ERR_FILE_OPEN – Could not open input file.
INVALID_INTERNAL_IMAGE - Cannot process the image as the internal image buffer is empty or invalid.

IMG_ERR_SUCCESS – Function processed successfully.

Remarks
Use this function to find the image dimension in pixels.

GetImageProperties

Format

```
GetImageProperties (filename As String, color As Integer, dpi As Integer)
```

Parameters
fileName: Image file name or an empty string if referring to the internal image.
color: Returns the image color scheme.
Height: Returns the image resolution in dpi.

Return
IMG_ERR_FILE_OPEN – Could not open input file.
INVALID_INTERNAL_IMAGE - Cannot process the image as the internal image buffer is empty or invalid.
IMG_ERR_SUCCESS – Function processed successfully.

Remarks
Use this function to find the image color scheme and resolution. The color scheme may be set to one of the following:

IMAGE_COLOR_TRUE: The image is a TRUE COLOR image (24 bit).
IMAGE_COLOR_256: The image has 256 colors.
IMAGE_GRAY_256: The image has 256 gray colors.
IMAGE_BW: The image is using 2 colors – black and white.

TextStamp

Format

```
TextStamp (sourceFile As String, Text As String, textHeight As Integer,
vertLocation As Integer , textColor As Integer, destFile As String)
```

Parameters
sourceFile: Source image file name or an empty string if referring to the internal image.
Text: The text to be printed on the image.
textHeight: The height of the text (in pixels).
VertLocation: The location of the text in the image (top, middle, bottom)
textColor: The color of the text (RGB).
destFile: Destination image file name or an empty string if referring to the internal image.

Return
IMG_ERR_FILE_OPEN – Could not open input file.
INVALID_INTERNAL_IMAGE - Cannot process the image as the internal image buffer is empty or invalid.
IMG_ERR_BAD_PARAM – Bad *vertLocation* parameter.
IMG_ERR_SUCCESS – Function processed successfully.

Remarks
Use this function to stamp text on an image. The text is printed on the horizontal center of the image. The vertical alignment of the text is set by the parameter *vertLocation* as follows:
IMAGE_TOP: Print the text on the upper portion of the image
IMAGE_MIDDLE: Print the text on the middle portion of the image
IMAGE_BOTTOM: Print the text on the bottom portion of the image

StampTextEx

Format

```
TextStampEx (sourceFile As String, Text As String, color As Long, opacity As
Long, verticalPos As Long, horizontalPos As Long, fontName As String, fontSize
As Long, bold As Long, bUnderline As Long, ret As Long)
```

Parameters
Image: Source image file name or an empty string if referring to the internal image.
txt: The text to be printed on the image.
color: The color of the text (RGB).
Opacity: The opacity value of the text color.
verticalPos: The location of the text in the image (top, middle, bottom)
horizontalPos: The location of the text in the image (left, middle, right)
fontName: Name of the font to use.
fontSize: Size of the font to use.
bold: Use bold or not.
bUnderline: Use underline or not.
Return value
None.

Remarks
Use this function to stamp text on an image. The text is printed on the image per the location that you give. The vertical alignment of the text is set by the parameter *verticalPos* as follows:
IMAGE_TOP: Print the text on the upper portion of the image
IMAGE_MIDDLE: Print the text on the middle portion of the image
IMAGE_BOTTOM: Print the text on the bottom portion of the image

parameter *horizontalPos* as follows:
IMAGE_LEFT: Print the text on the left portion of the image
IMAGE_MID_HOR: Print the text on the middle portion of the image
IMAGE_RIGHT: Print the text on the right portion of the image

GetImageBufferData

Format

```
GetImageBufferData (sFileType As String, pBuf As String) As Integer
```

Parameters
sFileType: Set the buffer image format. May be one of the following strings (case insensitive):
“BMP”
“JPG”
“PNG”
“TIF”
“TGA”
“PSD”
“PCX”
“JPGIR”
“JPGUV”

Note: if the value JPGIR or JPGUV are used than the buffer will contain the IR or UV image. Otherwise, the buffer contains the visible image n the requested format.

pBuf: A string buffer that will contain the image content upon successful return. sFileType

Return
INVALID_INTERNAL_IMAGE - Cannot process the image as the internal image buffer is empty or invalid.
BAD_IMAGE_DUMP – Could not dump the image buffer due to internal error.
IMG_ERR_SUCCESS – Function processed successfully.

Remarks
Use this function to get a copy of the internal image buffer. If successful, the function will format the image according to the given image format in the parameter *sFileType* and copy it to the *pBuf*. The application can dump content of pBuf to the hard drive as a binary file while assigning it a proper file extension, creating a standard image file.

GetImageBufferDataBack

Format

GetImageBufferDataBack (**sFileType** As String, **pBuf** As String) As Integer

Parameters
sFileType: Set the buffer Back image format. May be one of the following strings (case insensitive):
“BMP”
“JPG”
“PNG”
“TIF”
“TGA”
“PSD”
“PCX”
“JPGIR”
“JPGUV”

Note: if the value JPGIR or JPGUV are used than the buffer will contain the IR or UV image. Otherwise, the buffer contains the visible image n the requested format.

pBuf: A string buffer that will contain the Back image content upon successful return. sFileType

Return
INVALID_INTERNAL_IMAGE - Cannot process the Back image as the internal image buffer is empty or invalid.
BAD_IMAGE_DUMP – Could not dump the Back image buffer due to an internal error.
IMG_ERR_SUCCESS – Function processed successfully.

Remarks
Use this function to get a copy of the internal Back image buffer. If successful, the function will format the image according to the given image format in the parameter *sFileType* and copy it to the *pBuf*. The application can dump content of pBuf to the hard drive as a binary file while assigning it a proper file extension, creating a standard image file.

ImgDespeckle

Format

ImgDespeckle (**sImageFile** As String, **maxBlobArea** As Integer) As Integer

Parameters
sImageFileType: : Source image file name or an empty string if referring to the internal image.
maxBlobArea: : The maximum area size (in square pixels) of random pixels that will be considered as noise and will be removed from the image (default is 30).

Return
IMG_ERR_FILE_OPEN- Cannot open image file specified by *sImageFile*.
INVALID_INTERNAL_IMAGE - Cannot process the image as the internal image buffer is empty or invalid.
IMG_ERR_SUCCESS – Function processed successfully.

Remarks
Use this function to clean black and white images from random pixels. Such noise may be shown when converting images with gradient color transition from color to black and white color. The higher the value set in *maxBlobArea*, the more details in the image will be considered as noise and will be removed.

SetMainImage

Format

SetMainImage (**iImageType** As Integer) As Integer

Parameters
iImageType: Set the main image buffer index. Available values are:
MAIN_IMAGE (0): The front side image from the scanner.
MAIN_BC_IMAGE (1): The front side barcode image (for SnapShell® model only).
BACK_IMAGE (2): The Back side image from the scanner.
BACK_BC_IMAGE (3): The Back side barcode image from the scanner (for SnapShell® model only).

Return
IMG_ERR_BAD_PARAM- Return if iImageType is not one of the values 0-3.
IMG_ERR_SUCCESS – Function processed successfully.

Remarks
Use this function to select the current image that the SDK will use as the main image. This allows you to do image manipulation on the back image. In such a case, you need to make sure to switch back to the main image once the back image manipulation is done.

AlwaysFlagAsCropped

Format

AlwaysFlagAsCropped (**bCrop** As Boolean)

Parameters
bCrop: Controls the images cropping process. Available values are:
TRUE: images will **not** be cropped.
FALSE: images will be cropped.

Return
S_OK.

Remarks
The system will try cropping every image loaded by default. Use this function to indicate the system to stop cropping or to restart cropping again.

SetMobileCfg

Format

SetMobileCfg(mobileType as WORD, mobileDevModel as DWORD, deviceID as BSTR, orgImageW as INT, orgImageH as INT, orgImageBitsPerPixels as INT, fRotAngle as FLOAT, cropRectangle as RECT, fScaleFactor as FLOAT, mountingType as WORD) as Long

Parameters
[in] mobileType - Mobile OS. If the mobileType is set to 0 the system exists Mobile mode.
This paramter can be one of the following values:
0: NoMobile
1: IPhone
2: Android
3: Windows Mobile
[in] mobileDevModel - Specific device model TBD, currently not in use.
[in] deviceID - Device unique id (for example Android devices :ANDROID_ID - 64-bit number (as a hex string 16chars), iPhone devices : NSUUID : 32 hex number etc.).
[in] orgImageW - Image width as captured by the device (before any process).
[in] orgImageH - Image height as captured by the device (before any process).
[in] orgImageBitsPerPixels - Camera bits per pixels resolution.
[in] fRotAngle - Camera rotation angle, use 0 if no rotation applied.
[in] cropRectangle - Crop rectangle (if the image is cropped), use 0,0,0,0 if no cropping applied.
[in] fScaleFactor - Scale factor (if the image is scaled), use 1.0 if no scaling applied.
[in] mountingType - Mounting device (if used).
This parameter can be one of the following values:
0: no mounting
1: Snap Tab

Return
If the function succeeds, the return value is ID_TRUE. If the function fails, the value ID_FALSE is returned

Remarks
Use the function to switch the SDK to mobile mode and set mobile device configuration.
Calling any subsequent processing method (as DetectState, ProcessState etc.) will use this configuration in the processing pipeline.
Setting MobileType to 0 will exit SDK mobile mode.

Library CDynamicExport: General Functionality

The library CDynamicExport in ScanWex lets you export the scanned fields to any window you wish to.

DyamicFieldExtraction also lets you export the scanned fields to any window you wish to.

There are 2 steps for using the library:

- In the first step you will have to connect each scanned field to a specific window and save this configuration. You will be doing this by using a setup dialog from our SDK.

- In the second step, the SDK uses the previously saved data, and exports the fields scanned by it directly to the chosen fields.

CDynamicExport Library Functions

OpenDynamicConfigDialog

Format

```
OpenDynamicConfigDialog (Type As Integer, sDefaultCfgFilename As String) As Integer
```

Parameters
[in] **Type** – Indicates what is the component/components you are about to configure. i.e. – IDCard, Passport etc..
Possible values are - DRIVER_LICENSE, BAR_CODE, PASSPORT, RAW_BARCODE, RAW_OCR, BUISNESS_CARD, CHEQUE, MED. These values are defined in the examples supplied with the SDK.
You can combine 1 or more values using the | (or) operator.
[in] **sDefaultCfgFilename** – A string specifying a filename to open when launching the dialog.
Return value
If the function succeeds, the return value is 1, otherwise zero.

Remarks
Use this function to setup the windows you want the fields to be exported to. When done, save the configuration file for later usage.

ExtractDynamicFieldsFromFile

```
ExtractDynamicFieldsFromFile (sCfgFilename As String, nUseWideString As Integer,
nUseWideString As Integer, nComponentType As Integer , FieldIndex As Integer) As Integer
```

Parameters
[in] **sCfgFilename -** A string specifying the configuration file to use in order to extract the scanned values.
[in] **nUseWideString** - If 0 then the fields exported are just regular strings otherwise wide strings.
[in] **nComponenetType -** Specifies what will be the extracted component. Possible choices are: DRIVER_LICENSE, BAR_CODE, PASSPORT, RAW_BARCODE, RAW_OCR, BUISNESS_CARD, CHEQUE, MED. Values can be combined with their operator. If this parameter is 0, all possible sections will be extracted. The default value is 0.
[in] **FieldIndex** - Specifies the field to be extracted. If the value of the parameter is -1, then all fields in the specified section will be extracted. Default value is -1.

Return value
If the function succeeds to open the configuration file, the return value is 1 otherwise the return value is 0. The return value does not guarantee that any field whatsoever was exported to a window.

Remarks
Use this function after you scanned the document. When the scan is done, this function will take the values in memory and will post it to the configured windows.

ExtractDynamicTextFromFile

Format

```
ExtractDynamicTextFromFile (sCfgFilename As String, sImgFile As String, type As
Integer, nUseWideString As Integer) As Integer
```

Parameters
[in] **sCfgFilename** - A string specifying the configuration file to use in order to extract the scanned values.
[in] **sImgFile** - Full path name of the original image.
[in] **type** - Instruct the OCR what type of data is written in the image. This value increases the detection accuracy and speeds up the OCR operation. This value can be one of the following values:
USE_ALPHANUM: The image contains alphanumeric characters.
USE_ALPHA_CAPS_ONLY: The image contains capital letters only.
USED_NUM_ONLY: The image contains numbers only.
[in] **nUseWideString** If 0, then the fields exported are just regular strings otherwise wide strings.

Return value
If the function succeeds in opening the configuration file, the return value is 1, otherwise the return value is 0. The return value does not guarantee that any field whatsoever was exported to a window.

Remarks
Use this function after you scanned the document. When the scan is done, this function will take the values in memory and will post it to the configured windows.

OpenMacroRecordDialog

Format

```
OpenMacroRecordDialog (sDefaultMacroFilenameAs String) As Integer
```

Parameters
[in] **sDefaultMacroFilenameAs -** A string specifying the configuration macro file.

Return value
If the function succeeds in opening the dialog, the return value is 1, otherwise the return value is 0.

Remarks
Use this function to launch the macro dialog for the purpose of recording keyboard and mouse events.
PlaySavedMacro

Format

```
PlaySavedMacro (sDefaultMacroFilenameAs String) As Integer
```

Parameters
[in] **sDefaultMacroFilenameAs** A string specifying the configuration macro file.

Return value
If the function succeeds, the return value is 1, otherwise the return value is 0.

Remarks
Use this function to play a previously recorded macro.

GetDynamicExportNeeded

Format

```
GetDynamicExportNeeded (sDefaultMacroFilenameAs String) As Integer
```

Parameters
[in] **sDefaultMacroFilenameAs** A string specifying the configuration file to use in order to extract the possible values.

Return value
This function returns a bitwise integer containing the components stored in the configuration file. Possible values are DRIVER_LICENSE, BAR_CODE, PASSPORT, RAW_BARCODE, RAW_OCR, BUISNESS_CARD, CHEQUE, MED. The values can be read using the & operator.

Library CActivation: General Functionality

The SDK core architecture prevents it from processing images that were not acquired from scanners that are manufactured by Card Scanning Solutions. This often becomes a problem for clients that already have a third-party scanner.

To enable the use of third party scanners, the SDK must be activated on the local machine. In the activation process, the SDK links the local PC and a special activation code on CSSN's server over the internet. Once the activation's success, the SDK will grant the processing of images that was acquired from any TWAIN scanner.
Note that once the activation key is used to activate the SDK on a specific machine, it will not be possible to use it for activation on other PC's.

CActivation Library Functions

ActivateOnServer

Format

```
ActivateOnServer (szActivationKey As String) As Integer
```

Parameters
[in] **szActivationKey** – A 16 character string code that grants the SDK activation on a single PC.
Return value
ACTIVATE_ERR_INVALID_LICENSE (-1): Bad szActivationKey.
ACTIVATE_ERR_NO_MAC (-2): Cannot find network card.
ACTIVATE_ERR_SERVER_REFUSED_CONNECTION (-3): Activation failed as the activation key was already used for activation on another PC.
ACTIVE_ERR_UNKNOWN_ERROR (-4): Communication error.
ACTIVE_ERR_FAIL_TO_CREATE_ACTIVATION_FOLDER (-5): Cannot create the destination folder for the activation file.
ACTIVE_ERR_FAIL_TO_DELETE_EXISTING_ACTIVATION_FILE (-11): Failed to remove exiting activation file that was generated from previous activations.
ACTIVE_ERR_FAIL_TO_SAVE_ACTIVATION_FILE (-12): Failed to save activation file.
ACTIVE_ERR_NONE (1): The activation was successful.

Remarks
Use this function to activate the SDK. The PC must be connected to the internet to complete this function successfully. Once the operation succeeds, the SDK can be used with any TWAIN scanner.
Note: Once the SDK is activated successfully, it does not require the PC to be connected to the internet.

IsActivated

Format

```
IsActivated () As Integer
```

Parameters
None.

Return value
ACTIVE_ERR_FAIL (0): The SDK is not activated.
ACTIVE_ERR_NONE (1): The SDK is activated.

Remarks
Use this function to determine if the SDK is activated on the local PC.

GetActExpiryDate
Format

```
GetActExpiryDate (string format, string* pExpirt, bool* pNever) As Void
```

Parameters

[in] **format** – A string containing the desired format of the returned expiry date value. Use any acceptable system string format or specify NULL or an empty string to get the default local settings time/date format.
[out] **pExoiry** – The Activation's expiry date output.
[out] **pNever** – Indicates if Activation never expires. True = Activation never expires (in this case pExpiry will not be left as is), False = Activation will expire in which case the pExpiry will indicate the specific date.

Return value
None

Remarks
Use this method to find out on which date your Activation expires if ever.

Library CEPassport: General Functionality

The library CEPassport in ScanWex lets you read an electronic passport or an electronic ID, according to ICAO 9303 standards. An electronic passport/ID is a passport that includes an RFID chip or a contact chip. In order to read the passport, you will need a CSSN passport reader and an activation code.

First you read the passport using ReadPassport or ParsePassportRF and then you can access each field using GetField or GetFieldData for binary data.
Extended Access control and Active Authentication is not implemented yet.

CEPassport Library Functions

ReadPassport

Format

```
int ReadPassport (string sMrzLine1, string sMrzLine2, string sMrzLine3, string sDumpFolder)
```

Parameters

[in] **sMRZLine1** – A string containing the first line of the MRZ as it appears on the passport.
[in] **sMRZLine2** – A string containing the second line of the MRZ as it appears on the passport.
[in] **sMRZLine3** – A string containing the third line of the MRZ as it appears on the ID card. If only 2 lines exist on the MRZ, then put NULL in this parameter.

[in] s**DumpFolder** – A string containing the required location for the library to output the data from the passport.

Return value
The following returned values can be returned:
NO_READER -1
NO_PASSPORT_ON_READER -2
MRZ_WRONG -3
FAILED_DG1_PROCESS -4
MRDT_ERR_NO_LICENSE -5
READ_SUCCESS 0

Remarks

Use this function to read the RFID data from the passport chip.
All passports require the MRZ info to be entered in order to get access to the passport. You can use the OCR CPassport library to scan and read the MRZ info automatically.

ParsePassportRF

Format

```
int ParsePassportRf (string pBuf, int nSize);
```

Parameters

[in] **pBuf** – A string containing the buffer to be parsed.

[in] **nSize** – An integer containing the size of the buffer to be parsed.

Return value
The following returned values can be returned:

FAILED_DG1_PROCESS -4
MRDT_ERR_NO_LICENSE -5
READ_SUCCESS 0

Remarks

This function will parse a buffer read from the passport and will put each file in its corresponding place.

1.1.137 ParsePassportRFFile

Format

```
int ParsePassportRfFile (string pFilename, int bReset);
bReset);=*sDefaultCfgFilename=NULL);
```

Parameters

[in] **pBuf** – A string pointing to the file to be parsed.

[in] **bReset** – A Boolean, if set to true then the fields will be reset before parsing the new data.

Return value
The following returned values can be returned:

FAILED_DG1_PROCESS -4
MRDT_ERR_NO_LICENSE -5
READ_SUCCESS 0

Remarks
This function will parse a passport file and will put each file in its corresponding place.

GetField

```
string GetField (int index);
```

Parameters
[in] **nIndex** an integer specifying the index of the field to be retrieved.

Return value
[out] std::string

Remarks
This function will return a string containing the value of the field. Use this function after calling ReadPassport or ReadPassportRF / ReadPassportRFFile.

1.1.139 GetFieldByName

```
string GetFieldByName (string sName);
```

Parameters
[in] **sName** - a string containing the name of the field required.

It can be one of the following:

```
COUNTRY CODE
FIRST NAME
MIDDLE NAME
LAST NAME
PASSPORT_NUMBER
PASSPORT_CHEKC_DIGIT
NATIONALITY
DOB
DOB_CHECK_DIGIT
SEX
EXPIRES
EXPIRES_CHECK_DIGIT
PERSONAL_NUMBER
PERSONAL_NUMBER_CHECK_DIGIT
COMPOSITE_CHECK_DIGIT
MRZ
```

```
FACE_IMAGE
POB
FULL_NAME
OTHER_NAME
PERSONAL_NUMBER_DG11
FULL_DOB
PERMANET_ADDRESS
TELEPHONE_NUMBER
PROFESSION
TITLE
PERSONAL_SUMMARY
PROOF_OF_CITIZENSHIP
OTHER_VALID_TD_NUMBERS
CUSTODY_INFORMATION
CONTENET_SPECIFIC_CONSTRUVTED_DATA
NUMBER_OF_OTHER_NAMES
ISSUING_AUTHORITY
ISSUE_DATE YYYYMMDD
NAME_OF_OTHER_PERSON
ENDORSMENT
TAX_EXIT_REQUIREMENTS
IMAGE_OF_FRONT_DOCUMENT
IMAGE_OF_REAR_DOCUMENT
DATE_TIME_OF_DOC_PERSONALIZATION
SERIAL_NUM_OF_PERSONALIZATION_SYSTEM
```

Return value

[out] String

Remarks

This function will return a string containing the value of the field. Use this function after calling ReadPassport or ReadPassportRF / ReadPassportRFFile. If the field is not found, the return value will be “No such field”

GetFieldName

Format

```
string GetFieldName (int index);
```

Parameters

[in] **nIndex** - An integer specifying the index of the field to be retrieved.

Return value

[out] string

Remarks

This function will return a string containing the name of the field according to the index.

GetFieldDesc

Format

```
string GetFieldDesc (int nIndex);
```

Parameters

[in] **nIndex** - An integer specifying the index of the field to be retrieved. Right now only FACE_IMAGE (16) is supported.

Return value

[out] integer – The type of the data.

Remarks

Right now only “JPG” or “JP2” is returned.

PassiveAuthentication

Format

```
int PassiveAuthenticaion (string sCerFile, string sDumpFolder)
```

[in] **pDumpFolder –** A string containing the path for the functions file to be placed.

Parameters

[in] **sCerFile –** A Null terminated string containing the name of the public certificate of the passport, if available.

[in] **pDumpFolder –** A Null terminated string containing the path for the functions file to be placed.

Return value

[out] integer – The function will return true if authentication was successful, otherwise false.

Remarks

Call this function to check if passport data is authenticated and genuine. The specification of passive authentication appears under ICAO 9303 standards.

IsCountryCertificateApproved.

Format

```
int IsCountryCertificateApproved();
```

Parameters

Return value
[out] boolean – The function will return true if the country certificate fits the checked passport.

Remarks
Call this function after a successful call to passive authentication.

IsChipCertificateApproved.

Format

```
int IsChipCertificateApproved();
```

Parameters

Return value
[out] boolean – The function will return true if the chip certificate fits the checked passport.

Remarks
Call this function after a successful call to passive authentication.

IsPassiveAuthenticaionFlowSucceded.

Format

```
bool IsPassiveAuthenticaionFlowSucceded ();
```

Parameters

Return value
[out] bool – The function will return true if the flow of the authentication completed successfully

Remarks
Call this function after a successful call to passive authentication.

GetPassiveAuthenticationLastError.

Format

```
int GetPassiveAuthenticationLastError ();
```

Parameters

Return value
[out] bool – The function will return the latest error after a call to Passive Authentication, or zero if there is no error.

Error List:
EXE_FAILED_1=-2
EXE_FAILED_2=-3
EXE_FAILED_3=-4
EXE_FAILED_4=-5
EXE_FAILED_5=-6
READ_HASH_FAILED=-7
NOT_AUTHENTICATED=-8
TAIL_SOD_FAILED=-9
PEM_FILE_EXISTS=-10

Remarks
Call this function after a successful call to passive authentication. Used mainly for debugging.

WriteFieldData.

Format

```
int WriteFieldData (int index, string sFileName)
```

Parameters

[in] **nIndex –** An integer specifying the index of the field to be written.
Right now only FACE_IMAGE (16) is supported.
[in] sFilename – A string specifying the file name to write the data to.

Return value
[out] bool – The function will return true if the chip certificate fits the checked passport.

Remarks
Call this function after a successful call to read a passport. This function will write a field data to the disk.

IsRfIdReaderExists

Format

```
long IsRfIdReaderExists (bool* bIsRfIdReaderExists)
```

Parameters

[out] **bIsRfIdReaderExists -** boolean pointer.

Return

Call status.

Remarks

Use this function to detect presence of RF-Id reader on the passport camera.

Constants list

Return values:

NO_READER -1
NO_PASSPORT_ON_READER -2
MRZ_WRONG -3
FAILED_DG1_PROCESS -4
MRDT_ERR_NO_LICENSE -5
READ_SUCCESS 0

Fields indexes:

```
RFID_FIELD_FIRST 0
COUNTRY_CODE 0
FIRST_NAME 1
MIDDLE_NAME 2
LAST_NAME 3
PASSPORT_NUMBER 4
PASSPORT_CHEKC_DIGIT 5
NATIONALITY 6
DOB 7
DOB_CHECK_DIGIT 8
SEX 9
EXPIRES 10
EXPIRES_CHECK_DIGIT 11
PERSONAL_NUMBER 12
PERSONAL_NUMBER_CHECK_DIGIT 13
COMPOSITE_CHECK_DIGIT 14
MRZ 15
/ Dg 2
FACE_IMAGE 16
// Dg 11
POB 17
FULL_NAME 18
OTHER_NAME 19
PERSONAL_NUMBER_DG11 20
FULL_DOB 21
PERMANET_ADDRESS 22
TELEPHONE_NUMBER 23
PROFESSION 24
TITLE 25
PERSONAL_SUMMARY 26
PROOF_OF_CITIZENSHIP 27
OTHER_VALID_TD_NUMBERS 28
CUSTODY_INFORMATION 29
```

```
CONTENET_SPECIFIC_CONSTRUVTED_DATA 30
NUMBER_OF_OTHER_NAMES 31
// DG 12
ISSUING_AUTHORITY 32
ISSUE_DATE 33
NAME_OF_OTHER_PERSON 34
ENDORSMENT 35
TAX_EXIT_REQUIREMENTS 36
IMAGE_OF_FRONT_DOCUMENT 37
IMAGE_OF_REAR_DOCUMENT 38
DATE_TIME_OF_DOC_PERSONALIZATION 39
SERIAL_NUM_OF_PERSONALIZATION_SYSTEM 40
```

Silent installation

Wise Installer SDK (Used until February 2015):

The SDK setup can be run silently by running the SDK setup from a command line ex: “C:\SDK_Setup.exe \S”.
Running the SDK setup with the “\S” command will make the SDK setup running with its defaults settings in silent mode.
Other option to run the SDK silent in silent mode but with the capability to control its behavior and not use its defaults settings is to add the command line option “\M=” followed by a text file that contain the variables and values that you want to set in the SDK setup. Example of such a command line is: “C:\SDK_Setup.exe \S\M=C:\MySilentValues.txt”.

The silent values text file content can contain any of the following variables according to the values that specified in the following section.

[MAINDIR] (SDK destination folder)
Valid path for the destination folder to install the SDK in it.
Empty: setup will use the default path ex: C:\Program Files\Card scanning Solutions\SDK

[APP_COMPONENTS] (SDK components selection)
A: Driver License, ID & Passports
B: Business Card
C: Check
D: Medical Cards
E: Live Update SDK

[COMPONENTS] (Drivers selection)
A: ScanShell800R
B: ScanShell800NR
C: ScanShell800Dx
D: ScanShell800DxN
E: ScanShell1000A\NA\B\NB
F: ScanShell2000R
G: ScanShell2000NR
H: ScanShell3100\D\DN
I: Fujitsu F-60 (Add-in to driver and using security dongle)
J: SnapShell Camera
K: Twain Scanner (using security dongle)
L: RTE8000
M: MagShell900
N: Digimarc data verification
O: MagTek Excella STX
P: 3M AT9000
Q: IPScan
R: Citrix – TWAIN
S: ScanShell900DX
T: SnapShell ePassport Camera

[DONGLE_COMPONENTS] (Dongles selection)
A: Blue\Green GeniusDog Dongle
B: Purple HASP Dongle
C: I don’t know (install both drivers)

[SIGNISHELL_COMPONENTS] (Signishell pads driver selection)

A: Signishell verification support

[INSTALL_TYPE] (SDK installation type selection)
A: Express Install
B: Custom Install

[CUSTOM_COMPONENTS] (SDK component selection. This option needed if INSTALL_TYPE=B)
A: USA
B: Canada
C: South America
D: Europe
E: Australia
F: Asia
G: General Documents
H: Africa
I: Passports

[INSTALL_VCREDIST] (Prerequisite installation of VCRedist runtime files)
A: Not install the VCRedist.
Empty: Will install it

[RF_COMPONENTS] (RAFID driver selection)
A: Install RF ID Driver
Empty: Will not install

[INSTALL_SHORTCUT] (Install or not shortcuts)
0: Will not install any shortcuts.
Empty: Will install it

[UNINSTALL_OLD_SDK]
1: Uninstall current SDK from the given path.
Empty: Will install the SDK without uninstalling the previous SDK installation.

The values in the text file listed here in the brackets (ex: [APP_COMPONENTS list]) and the possible values can be one or more of the letters under it. If you want to set the variable to more than one option you can combine more letters without spaces (ex: A or ABF).

Sample of the silent values text file:

```
APP_COMPONENTS=AE
MAINDIR=
COMPONENTS=AJ
DONGLE_COMPONENTS=
INSTALL_TYPE=A
CUSTOM_COMPONENTS=ABCDEFGHI
INSTALL_VCREDIST=
RF_COMPONENTS=
UNINSTALL_OLD_SDK=
INSTALL_SHORTCUT=
```

Install Shield Installer SDK (Used from February 2015):
Using Install Shield in silent mode is very simple.
First you run the SDK setup with this command line: <path>setup.exe /r /f1"<path to file.iss> (ex: “C:\sdk_setup.exe /r /f1”C:\SilentValues.iss”).
This will record every step and everything that you do during the SDK installation process including all your selections and will save it in the iss file you specified in the command line.

After the iss file is created you can easily modified it as a text file in order to see its content or change some settings for your needs.

To run the SDK setup ion silent more you will need to run it with this command line: <path>setup.exe /s /f1"<path to file.iss> (ex: “C:\sdk_setup.exe /s /f1”C:\SilentValues.iss”).
This will run the SDK setup using the recorded iss file that you recorded before.