SDK

Application Program Interface

November 2014
# Table of Contents

Table of Contents ................................................................................................................................. 2

Introduction............................................................................................................................................. 6
  General.................................................................................................................................................. 6
  Libraries.............................................................................................................................................. 6
  Libraries Functionality....................................................................................................................... 7
  Libraries Initialization....................................................................................................................... 8
  Distribution........................................................................................................................................ 9

Library Properties and Methods............................................................................................................. 10
  Library Slib.dll: General Functionality ............................................................................................ 10

Slib Library Functions............................................................................................................................ 10
  Calibrate............................................................................................................................................ 10
  CalibrateEx....................................................................................................................................... 12
  CleanScanner.................................................................................................................................... 12
  GetProgress...................................................................................................................................... 13
  GetScannerColor............................................................................................................................... 13
  GetScannerResolution...................................................................................................................... 13
  GetScannerSize................................................................................................................................. 14
  GetScannerType............................................................................................................................... 14
  InitScanLib....................................................................................................................................... 15
  IsPaperOn.......................................................................................................................................... 15
  IsScannerValid................................................................................................................................... 15
  NeedCalibration............................................................................................................................... 16
  ResetImage....................................................................................................................................... 16
  ScanBmpFile..................................................................................................................................... 16
  ScanBmpFileEx................................................................................................................................ 17
  SetContinueScan............................................................................................................................. 18
  SetScannerColor............................................................................................................................... 18
  SetScannerResolution...................................................................................................................... 18
  SetScannerSize................................................................................................................................ 19
  UnInitScanLib................................................................................................................................... 20
  SetCalThreshold.............................................................................................................................. 20
  GetCalThreshold.............................................................................................................................. 21
  SetDuplex......................................................................................................................................... 21
  GetDuplex....................................................................................................................................... 21
  SetInOutScan.................................................................................................................................... 21
  GetInOutScan.................................................................................................................................. 22
  SetStartModel................................................................................................................................. 22
  UseFixedModel............................................................................................................................... 23
  GetLibVersion................................................................................................................................. 23
  GetPressedButton........................................................................................................................... 24
  InitScanLibEx................................................................................................................................... 24
  GetOwnerData................................................................................................................................. 24
UnInitScanLibEx ................................................................. 25
UnloadSnapServer ............................................................ 25
TotalConnectedScanners .................................................. 26
AssignScanner ................................................................. 26
ReleaseScanner ............................................................... 26
SetCurScanner ............................................................... 27
GetCurScanner ............................................................... 27
GetFirstTwainSource ....................................................... 27
GetNextTwainSource ....................................................... 28
GetTwainScanner ............................................................ 28
SetTwainScanner ............................................................ 28
UseFixedModel ............................................................... 29
SetCentered ................................................................. 29
SetRemoteIP ................................................................. 29
IsRemoteScannerValid ..................................................... 30
UpdateRemoteScanner ..................................................... 30
IsUpdateSuccessful ....................................................... 30
GetRemoteScannerVersion ................................................ 31
ScanBmpFileAsync .......................................................... 31
GetJobList ................................................................. 31
GetJobStatus ............................................................... 32
ClearJobList ............................................................... 32
CountCSSNDimes .......................................................... 32
GetCSSNDevice ............................................................ 32
GetSnapSerial .............................................................. 33
SetExternalTriggerMode .................................................. 33
SetExternalTrigger ....................................................... 33
SetExternalOperation ..................................................... 34
IdCard Library Functions .................................................. 34
Library IdCard.dll: General Functionality .......................... 34
CountySupportAutoDetect ................................................ 34
DetectState ................................................................. 34
DetectStateEx .............................................................. 35
GetAccuracy ............................................................... 36
GetCntryNameByld ........................................................ 36
GetCountryByState ........................................................ 37
GetFace ................................................................. 37
GetFaceEx ................................................................. 38
GetFaceEx ................................................................. 38
GetFaceDuplex ............................................................ 39
GetField ................................................................. 39
GetFirstCntry ............................................................. 40
GetFirstStateByCntry ...................................................... 40
GetLastField ............................................................. 41
GetNextCntry ............................................................. 41
GetNextStateByCntry ...................................................... 41
GetSignature ............................................................. 44
GetSignatureDuplex ......................................................... 44
GetStateNameByld ........................................................ 45
SCORdll Library Functions

1.1.104 SetBrightnessForFaceImage
1.1.105 SetFieldToBeErasedFromImage

Library Barcode: General Functionality

Barcode Library Functions

1.1.106 InitBCLib
1.1.107 ProcessBC
1.1.108 Process1DBC
1.1.109 ProcAllBarcodes
1.1.110 BarcodeDetectionQuality
1.1.111 Refresh
1.1.112 GetRawText
1.1.113 GetBCField

Library SCORdll: General Functionality

SCORdll Library Functions

1.1.114 InitOcrLib
1.1.115 GetAccurateTextFromFile
1.1.116 GetTextFromFileUsingOCR

Library CPassport: General Functionality

CPassport Library Functions

1.1.117 Init
1.1.118 Process
1.1.119 GetFaceImage
1.1.120 GetPassportSignature
1.1.121 GetPassportField
Library MagLib: General Functionality
MagLib Library Functions
1.1.126 InitMagLib ............................................. 69
1.1.127 IsMagValid ............................................. 70
1.1.128 WasMagSwept ......................................... 70
1.1.129 ProcessMag ............................................ 70
1.1.130 ProcessMagStr ........................................ 71
1.1.131 GetMagRawText ....................................... 71
1.1.132 GetMagField .......................................... 71
1.1.133 ResetMagDevice .................................... 72
1.1.134 UnInitMagLib ........................................ 72

Library ClmageCtrl: General Functionality
CImageCtrl Library Functions
1.1.135 InitImageLib .......................................... 73
1.1.136 GetImgColor .......................................... 73
1.1.137 LoadRotateSave ..................................... 74
1.1.138 ConvertImgFormat .................................... 75
1.1.139 ConcatImage .......................................... 77
1.1.140 ImageDespeckle ...................................... 78
1.1.141 StampText ............................................. 78
1.1.142 StampTextEx .......................................... 79
1.1.143 getImgBufferLengthEx .............................. 80
1.1.144 AlwaysFlagAsCropped ............................... 80
1.1.145 GetActExpiryDate .................................... 80

Library DynamicFieldExtraction: General Functionality
DynamicFieldExtraction Library Functions
1.1.146 OpenConfigDialog ................................... 81
1.1.147 ExtractDynamicFields ............................... 82
1.1.148 ExtractTextFromFile .................................. 82
1.1.149 IsDynamicExportNeeded ............................ 83
1.1.150 OpenMacroDialog .................................... 83
1.1.151 PlayMacro ............................................. 84

Library IMRTD.dll: General Functionality
IMRTD Library Functions
1.1.152 ReadPassport .......................................... 84
1.1.153 ParsePassportRF ..................................... 85
1.1.154 ParsePassportRFFile ............................... 85
1.1.155 GetField .............................................. 86
1.1.156 GetField .............................................. 86
1.1.157 GetFieldName ......................................... 88
1.1.158 GetFieldData ......................................... 88
1.1.159 GetFieldDataLen .................................... 88
1.1.160 GetFieldDataDesc .................................. 89
1.1.161 GetNumFields ....................................... 89
1.1.162 PassiveAuthentication ............................ 89
Introduction

General

Card Scanning Solution’s SDK provides the capability to scan documents, manipulate the image and extract the data from the image. The SDK is a collection of 5 libraries, and each handles a different aspect of the process as follows:

- **Slib.dll:** Controls the scanner (ScanShell® 800XX, ScanShell®1000, ScanShell® 2000XXX, ScanShell® 3000XX, ScanShell® 4000, ScanShell® 5000 scanners and the SnapShell® camera) operation and saves the image in a bitmap file and in memory for further usage. The library is also capable of reading driver’s license magnetic strip data (using MagShell® 900 reader).
- **ImageCtrl.dll:** Gives the user the capability to easily manipulate images.
- **Socr.dll:** General OCR engine.
- **IdCard.dll:** Driver’s license analyzer (Text).
- **BarCode.dll:** Driver’s license analyzer (2D barcode) and general purpose PDF417 2D barcodes.
- **DynamicFieldExtraction.dll:** Enables dynamic export of the analyzed data to previously defined windows.

Libraries Relationship

The libraries Slib.dll and ImageCtrl.dll are the core libraries of the SDK. You must have these libraries in order to scan and manipulate the document image. All other libraries are optional and are used to extract different data types from the image. For example, if you only wish to scan the image and save it as, for example, a TIFF image, than you only need to use Slib.dll and ImageCtrl.dll. However, if you wish to scan a driver’s license and extract its data, then you will also need to use the libraries Socr.dll and IdCard.dll in addition to Slib.dll and ImageCtrl.dll.
The following table shows the libraries needed for different project types:

<table>
<thead>
<tr>
<th>Project Type</th>
<th>Libraries Needed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Slib.dll</td>
</tr>
<tr>
<td>Driver’s license magnetic strip reading</td>
<td>✓</td>
</tr>
<tr>
<td>General document scan</td>
<td>✓</td>
</tr>
<tr>
<td>General document scan and text extraction</td>
<td>✓</td>
</tr>
<tr>
<td>Driver’s license scan and text fields extraction</td>
<td>✓</td>
</tr>
<tr>
<td>General document scan and data extraction from 2D barcode</td>
<td>✓</td>
</tr>
<tr>
<td>Driver’s license scan and barcode data extraction</td>
<td>✓</td>
</tr>
</tbody>
</table>

**Libraries Functionality**

- **SlibEx** – Controls the scanner activity and contains the last scanned image in an internal memory. This library controls the scanning settings such as scan size, color scheme and resolution. The scanned image is loaded to an internal memory and can be saved to an external file in a bitmap format.

- **Ocr (In OCR version only)** – Extracts the textual data from the internal image.

- **IdData (In the ID version only)** – Parses and refines the textual data extracted by the OCR. The data is kept in internal variables ready to be exported to the application.

- **Cimage** – Used for internal image manipulation such as rotation, color scheme modification, dpi modification and image export to a file in various formats. This library can also be used for external image file manipulation.

- **DynamicFieldExtracton** – Provides an engine to Export the scanned fields into previously defined windows in any application.

The following figure shows the relationship between the libraries:
You can see that ImageCtrl.dll contains a buffer to the recently scanned image. This image buffer is filled up by SLib.dll (that also stores a copy of the image in a file on the hard drive). The image buffer is used to store the driver’s license image that can be processed by IdCard.dll. Alternatively, this buffer is also used for extracting general text (using SoCr.dll) from text cards. You can also save the image buffer to the disk (using ImageCtrl.dll API) in one of seven popular image formats (BMP, JPEG, TIFF, PCX, PSD, TGA, PNG).

**Libraries Initialization**

The very first step you need to do before using the libraries is to initialize each library with the license key. Failing to do so will cause a library to reject any call to its API functions.

There are two SDK license key types:

- **Temporary License Key:** This key is valid for one month only and can be obtained from the Card Scanning Solutions site at: [http://www.card-scanner.com/developers.html](http://www.card-scanner.com/developers.html). You will need to register on the site to access the SDK license section.

- **Permanent SDK License Key:** This key will be supplied to you in the product package once you make the purchase.

The scanner functions as a hardware plug. Therefore, before you initialize the libraries with a permanent license key, you must verify that the scanner is connected to the PC and that its driver is installed correctly. If
you are initializing the SDK libraries with a temporary license (for evaluation), then you do not need to have the scanner connected to the PC.

**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 installations 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 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 Slib.dll: General Functionality

Slib library is used to scan documents and save their image into external bitmap files and to the internal image buffer. The library also saves the scanned image to an external bitmap file. The library sets and retrieves the scanners properties (such as scanning size, resolution and color scheme). The library also encapsulates the magnetic reader functionality API.

Slib Library Functions

Calibrate

Format

```
short Calibrate ()
```

Return Value

- `LICENSE_INVALID` – Library was not initialized with the 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 (No calibration card was found).
- `SLIB_TRUE` – Operation succeeded.

Remarks

This function calibrates the scanner using the calibration card. The calibration creates calibration files and places them as follows:

<table>
<thead>
<tr>
<th>Scanner Name</th>
<th>File Path</th>
<th>File name</th>
</tr>
</thead>
<tbody>
<tr>
<td>ScanShell® 800</td>
<td>C:\Documents and Settings\All Users\Application Data\Card Scanning Solutions\ScanShell800</td>
<td>PixOffG6.dat, PixOff6.dat, PixGanG6.dat, PixGan6.dat, trvlusb.ini</td>
</tr>
<tr>
<td>ScanShell® 800N</td>
<td>C:\Documents and Settings\All Users\Application Data\Card Scanning Solutions\ScanShell800N</td>
<td>PixOffG6.dat, PixOff6.dat, PixGanG6.dat, PixGan6.dat, trvlusb.ini</td>
</tr>
<tr>
<td>ScanShell® 800R</td>
<td>C:\Documents and Settings\All Users\Application Data\Card Scanning Solutions\ScanShell800R</td>
<td>Calibration.dat</td>
</tr>
<tr>
<td>ScanShell® 800DX</td>
<td>C:\Documents and Settings\All Users\Application Data\Card Scanning Solutions\ScanShell800DX</td>
<td>Calibration.dat</td>
</tr>
<tr>
<td>ScanShell® 800DXN</td>
<td>C:\Documents and Settings\All Users\Application Data\Card Scanning Solutions\ScanShell800DXN</td>
<td>Calibration.dat</td>
</tr>
<tr>
<td>Model</td>
<td>Path</td>
<td>File(s)</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------------------------------------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>ScanShell® 800NR</td>
<td>C:\Documents and Settings\All Users\Application Data\Card Scanning Solutions\ScanShell800NR</td>
<td>Calibration.dat</td>
</tr>
<tr>
<td>ScanShell® 2000R</td>
<td>C:\Documents and Settings\All Users\Application Data\Card Scanning Solutions\ScanShell2000R</td>
<td>Calibration.dat</td>
</tr>
<tr>
<td>ScanShell® 2000NR</td>
<td>C:\Documents and Settings\All Users\Application Data\Card Scanning Solutions\ScanShell2000NR</td>
<td>Calibration.dat</td>
</tr>
<tr>
<td>ScanShell® 3000D</td>
<td>C:\Documents and Settings\All Users\Application Data\Card Scanning Solutions\ScanShell3000D</td>
<td>Calibration.dat</td>
</tr>
<tr>
<td>ScanShell® 3000DN</td>
<td>C:\Documents and Settings\All Users\Application Data\Card Scanning Solutions\ScanShell3000DN</td>
<td>Calibration.dat</td>
</tr>
<tr>
<td>ScanShell® 3100D</td>
<td>C:\Documents and Settings\All Users\Application Data\Card Scanning Solutions\ScanShell3100D</td>
<td>Calibration.dat</td>
</tr>
<tr>
<td>ScanShell® 3100DN</td>
<td>C:\Documents and Settings\All Users\Application Data\Card Scanning Solutions\ScanShell3100DN</td>
<td>Calibration.dat</td>
</tr>
</tbody>
</table>

**Notes:** ScanShell® 800, ScanShell® 800N and ScanShell® 1000 scanners write the calibration files on a directory that might have security restrictions in NTFS drives. This may prevent users with insufficient security level from calibrating the scanner, which results in calibration failure. To remove the security restrictions, please consult the documentation or contact Acuant support.
from this location, run the program **ResetSec.exe** once while you are logged in with Administrator privileges. This application can be obtained from:
http://www.id-scan.com/FTP/Applications/tools/ResetSec.exe

See also
NeedCalibration()

CalibrateEx

**Format**

```
short CalibrateEx ()
```

**Return Value**

- **LICENSE_INVALID** – Library was not initialized with the 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 (No calibration card was found).
- **SLIB_TRUE** – Operation succeeded.

**Remarks**

This function is similar to the **Calibrate** function but it will display the calibration progress bar dialog.

CleanScanner

**Format**

```
short CleanScanner ()
```

**Return Value**

- **LICENSE_INVALID** – Library was not initialized with the proper license.
- **SLIB_ERR_INVALID_SCANNER** – The attached scanner is invalid (neither ScanShell® 800 nor ScanShell® 800N).
- **SLIB_FALSE** – The operation failed.
- **SLIB_TRUE** – Operation succeeded.

**Remarks**

This function cleans the scanner lens by running the cleaning pad (Supplied in the scanner kit) back and forth.

**Note:**

This function applies only to ScanShell® 800 and ScanShell® 800N models.
GetProgress

Format

```c
short GetProgress ()
```

Return Value
A value in the range 0-100 that describes the current scan progress.

Remarks
Call this function from a separate thread to obtain the scan progress.

GetScannerColor

Format

```c
short GetScannerColor ()
```

Return Value
The current scanner’s color scheme (BW, GRAY or TRUECOLOR).

Remarks
Call this function to retrieve the current color scheme.

See Also
SetScannerColor()

GetScannerResolution

Format

```c
short GetScannerResolution ()
```

Return Value
The scanner’s resolution setting.

Remarks
Call this function to retrieve the resolution setting.

See also
SetScannerResolution()
GetScannerSize

Format

```c
short GetScannerSize (short * pWidth, short * pHeight)
```

Parameters

- `[out]` `pWidth` - The document width in milli-inches.
- `[out]` `pHeight` - The document height in milli-inches.

Return Value

- `LICENSE_INVALID` – Library was not initialized with the proper license.
- `SLIB_ERR_SCANNER_NOT_FOUND` – No attached scanner was found.
- `SLIB_ERR_BAD_WIDTH_PARAM` – `pWidth` value is NULL.
- `SLIB_ERR_BAD_HEIGHT_PARAM` – `pHeight` value is NULL.
- `SLIB_ERR_NONE` - Function successful.

Remarks

Retrieve the scanner size in milli inch units.

See Also

- `SetScannerSize()`

GetScannerType

Format

```c
short GetScannerType ()
```

Return Value

The attached scanner's type:

- `CSSN_NONE` – No attached scanner was found.
- `CSSN_600` – ScanShell® 600 is attached to the PC.
- `CSSN_800` – ScanShell® 800 is attached to the PC.
- `CSSN_800N` – ScanShell® 800N is attached to the PC.
- `CSSN_1000` – ScanShell® 1000 is attached to the PC.
- `CSSN_2000` – ScanShell® 2000 is attached to the PC.
- `CSSN_2000N` – ScanShell® 2000N is attached to the PC.
- `CSSN_800E` – ScanShell® 800E is attached to the PC.
- `CSSN_800EN` – ScanShell® 800EN is attached to the PC.
- `CSSN_3000` – ScanShell® 3000 is attached to the PC.
- `CSSN_4000` – ScanShell® 4000 is attached to the PC.
- `CSSN_800G` – ScanShell® 800G is attached to the PC.
- `CSSN_5000` – ScanShell® 5000 is attached to the PC.
- `CSSN_IDR` – SnapShell® is attached to the PC.
- `CSSN_800DX` – ScanShell® 800DX is attached to the PC.
- `CSSN_800DXN` – ScanShell® 800DXN is attached to the PC.
- `CSSN_FDA` – SnapShell® is attached to the PC.
CSSN_WMD – SnapShell® is attached to the PC.
CSSN_TWN – SnapShell® is attached to the PC.

Remarks
Call this function to retrieve the attached scanner type.

InitScanLib

Format

```
short InitScanLib (const char * szLicense)
```

Parameters
[in] szLicense - Null terminated string that holds the license key value.

Return Value
- LICENSE_VALID: License is valid and the library is ready to be used.
- LICENSE_INVALID: The license is invalid. All scanner operations are disabled.
- LICENSE_EXPIRED: License has expired. All scanner operations are disabled.
- SLIB_ERR_DRIVER_NOT_FOUND: The scanner driver was not found. To fix this error, re-install the scanner’s driver. All scanner operations are disabled.
- SLIB_ERR_SCANNER_NOT_FOUND: The scanner is not connected to the PC. To fix this error, make sure the scanner is connected and re-start the function. All scanner operations are disabled.

Remarks
Use this function to initialize the scanner library. This function loads the scanner driver and initializes the internal image structure. This function must be called before calling any other function in the library.

IsPaperOn

Format

```
short IsPaperOn ()
```

Return Value
- SLIB_FALSE – No document was detected in the scanner’s tray.
- SLIB_TRUE – Document is in the tray.

Remarks
This function accesses the document sensor on the scanner and returns if a document was detected.

Note:
In ScanShell® 1000, this function returns if one of the silver buttons on the scanner was pressed.
short IsScannerValid ()

Return Value
SLIB_FALSE – Scanner is attached to the PC and resends correctly.
SLIB_TRUE – No Scanner is attached to the PC.

Remarks
This function accesses the scanner to verify its connectivity to the PC.

NeedCalibration

Format

short NeedCalibration ()

Return Value
LICENSE_INVALID – Library was not initialized with the proper license.
SLIB_ERR_SCANNER_NOT_FOUND – No attached scanner was found.
SLIB_FALSE – Scanner is calibrated.
SLIB_TRUE – Scanner needs to be calibrated.

Remarks
Retrieve if the scanner needs to be calibrated. This should be tested before every scan. Non-calibrated scanners may generate images with incorrect colors.

See Also
Calibrate(),

ResetImage

Format

short ResetImage ()

Remarks
Deletes the internal image buffer.

ScanBmpFile

Format

short ScanBmpFile (const char * szFileName)
Parameters

[in] szFileName - Null terminated string that holds the destination bitmap file name.

Return Value

If the function succeeds, the return value is `SLIB_ERR_NONE`. If the function fails, the return number may be one of the following:

- `LICENSE_INVALID` – Library was not initialized with the proper license.
- `SLIB_ERR_SCANNER_NOT_FOUND` – No attached scanner was found.
- `SLIB_ERR_SCANNER_GENERAL_FAIL` - Scanner failed to start.
- `SLIB_ERR_HARDWARE_ERROR` - Scanner hardware error.
- `SLIB_ERR_PAPER_FED_ERROR` - Document was not fed enough in the tray to create image file.
- `SLIB_ERR_SCANABORT` - Scanning was aborted.
- `SLIB_ERR_NO_PAPER` - No paper in tray.
- `SLIB_ERR_PAPER_JAM` - Paper was jammed while scanner is running.
- `SLIB_ERR_FILE_IO_ERROR` - Failed to create image file on disk.
- `SLIB_ERR_PRINTER_PORT_USED` - N/A
- `SLIB_ERR_OUT_OF_MEMORY` - Not enough memory to create temporary image.

Remarks

Scans document to the internal image buffer and, at the same time, exports it to a bitmap file in the local disk. Note that the image should be scanned in True color and 300 dpi format for proper OCR recognition.

ScanBmpFileEx

Format

```c
short ScanBmpFileEx(const char * szFileName)
```

Parameters

[in] szFileName - Null terminated string that holds the destination bitmap file name.

Return Value

If the function succeeds, the return value is `SLIB_ERR_NONE`. If the function fails, the return number may be one of the following:

- `LICENSE_INVALID` – Library was not initialized with the proper license.
- `SLIB_ERR_SCANNER_NOT_FOUND` – No attached scanner was found.
- `SLIB_ERR_SCANNER_GENERAL_FAIL` - Scanner failed to start.
- `SLIB_ERR_HARDWARE_ERROR` - Scanner hardware error.
- `SLIB_ERR_PAPER_FED_ERROR` - Document was not fed enough in the tray to create image file.
- `SLIB_ERR_SCANABORT` - Scanning was aborted.
- `SLIB_ERR_NO_PAPER` - No paper in tray.
- `SLIB_ERR_PAPER_JAM` - Paper was jammed while scanner is running.
- `SLIB_ERR_FILE_IO_ERROR` - Failed to create image file on disk.
- `SLIB_ERR_PRINTER_PORT_USED` - N/A
- `SLIB_ERR_OUT_OF_MEMORY` - Not enough memory to create temporary image.

Remarks

Scan from the start.
This function works identically to ScanBmpFile, and in addition, the function displays a modal progress bar while the scan is in progress.

SetContinueScan

Format

```c
short SetContinueScan (short continueCondition)
```

Parameters

[in] continueCondition - Value of 0 stops the scan progress immediately.

Return Value

- `SLIB_ERR_SCANNER_NOT_FOUND` – No attached scanner was found.
- `SLIB_ERR_NONE` - Function successful.

Remarks

This function can be called from a separate thread to abort ongoing scanning. Once this function is called with the parameter 0, the function ScanBmpFile returns with the return value `SLIB_ERR_CANCELED_BY_USER`.

SetScannerColor

Format

```c
short SetScannerColor (short color)
```

Parameters

[in] color - The color scheme of the scanned image. Available values are:
- `BW` – Black and White (1 bit) image.
- `GRAY` – 256 shades of gray image.
- `TRUECOLOR` - 24 bit image (default).

Return Value

- `LICENSE_INVALID` – Library was not initialized with the proper license.
- `SLIB_ERR_SCANNER_NOT_FOUND` – No attached scanner was found.
- `SLIB_ERR_BAD_PARAM` – Bad parameter value.
- `SLIB_ERR_NONE`- Function successful.

Remarks

Determines the color scheme of the scan result bitmap file.

See Also

- `GetScannerColor()`

SetScannerResolution
Format

```
short SetScannerResolution (short resolution)
```

Parameters

[in] resolution - The value of the required image resolution.

Return Value

If the function succeeds, the return value is `SLIB_ERR_NONE`. If the function fails, the return number may be one of the following:

- `LICENSE_INVALID` – Library was not initialized with the proper license.
- `SLIB_ERR_SCANNER_NOT_FOUND` – No attached scanner was found.
- `SLIB_ERR_SCANNER_GENERAL_FAIL` - Scanner failed to start.
- `SLIB_ERR_CANCELED_BY_USER` - The function `SetContinueScan` was activated to cancel the scan.
- `SLIB_ERR_HARDWARE_ERROR` - Scanner hardware error.
- `SLIB_ERR_PAPER_FED_ERROR` - Document was not fed enough in the tray to create image file.
- `SLIB_ERR_SCANABORT` - Scanning was aborted.
- `SLIB_ERR_NO_PAPER` - No paper in tray.
- `SLIB_ERR_PAPER_JAM` - Paper was jammed while scanner is running.
- `SLIB_ERR_FILE_IO_ERROR` - Failed to create image file on disk.
- `SLIB_ERR_PRINTER_PORT_USED` - N/A
- `SLIB_ERR_OUT_OF_MEMORY` - Not enough memory to create temporary image.

Remarks

Sets the scanner resolution settings. Resolution value can be any integer in the range 50-600 (for 50-600 dpi). The resolution value is rounded to the upper value of the nearest 100’s boundary. That is, the input value of 180 will be rounded by the function to 200, and the input value of 201 will be rounded to 300, etc. Trying to set a value outside this range will be rejected and the previous value will be used. The scanner’s default resolution is 300 dpi.

See Also

- `GetScannerResolution()`

SetScannerSize

Format

```
short SetScannerSize (short width, short height)
```

Parameters

[in] width - The document width. The scanner’s default width is 220 (2.2”).
[in] height - The document height. The scanner’s default height is 360 (3.6”).

Return Value

- `LICENSE_INVALID` – Library was not initialized with the proper license.
- `SLIB_ERR_SCANNER_NOT_FOUND` – No attached scanner was found.
- `SLIB_ERR_NONE` - Function successful.

Remarks

Smart from the start

acuantcorp.com

6950 Bristol Parkway
Culver City, California 90230
213.862.2621
Sets the document’s scan boundaries size. A document image beyond these boundaries will not reflect in the image file. If set to -1 for both values, the scanner will work in auto scan size (this is supported with regular scanners only).

See Also
GetScannerSize()

UnInitScanLib

Format

short UnInitScanLib()

Parameters

None.

Return Value

SLIB_UNLOAD_FAILED_BAD_PARENT: Cannot unload the driver since another application is using/has loaded it.
SLIB_NOT_INITIALIZED: The driver is not found in the memory (hence it 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 InitScanLib) before initializing the SDK from the other application.

SetCalThreshold

Format

short SetCalThreshold (int val)

Parameters

[in] val - The value to effect the 'is need' calibration function, 100=very clear image, 0 = image not clear.

Remarks

Effects the 'is need' calibration function.
Accepted values: 1 - 100, where 1 is most sensitive and 100 is least sensitive.
Default value: 60.
GetCalThreshold

Format

```
Int GetCalThreshold()
```

Parameters
None.

Remarks
Returns the calibration hold value.
Accepted values: 1 - 100, where 1 is most sensitive and 100 is least sensitive.
Default value: 60

SetDuplex

Format

```
short SetDuplex(short bVal)
```

Parameters
[in] val - 1 or 0 (True\False).

Remarks
Setting this value activates the double side scan when using scanner models ScanShell® 3000D\ScanShell® 800DX\ScanShell® 800DXN.

GetDuplex

Format

```
Short GetDuplex()
```

Parameters
None.

Remarks
Returns the Duplex value. 1 or 0 (True\False).

SetInOutScan

Format

```
short SetInOutScan(short bVal)
```

Parameters
[in] val - 1 or 0 (True\False).

Remarks

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.

GetInOutScan

Format

Short GetInOutScan ()

Parameters

None.

Remarks

Returns the InOutScan value. 1 or 0 (True(False).

SetStartModel

Format

void SetStartModel (int val)

Parameters

[in] val - Scanner type.

Can be one of the following values:

- 0: No Scanner.
- 1: ScanShell® 600.
- 2: ScanShell® 800.
- 3: ScanShell® 800N.
- 4: ScanShell® 1000.
- 5: ScanShell® 2000.
- 6: ScanShell® 2000N.
- 7: ScanShell® 800E.
- 8: ScanShell® 800EN.
- 9: ScanShell® 3000D.
- 10: ScanShell® 4000.
- 11: ScanShell® 800G.
- 12: ScanShell® 5000.
- 13: SnapShell® (IDR)
- 14: ScanShell® 800DX
- 15: ScanShell® 800DXN
- 16: SnapShell® (FDA)
- 17: SnapShell® (WMD)
- 18: SnapShell® (TWN)
- 19: CSSN_PASS
- 20: CSSN_RTE8K
- 21: CSSN_TWAIN_N
- 22: CSSN_MAGTEK_STX
- 23: CSSN_CLBS
Remarks
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

Format

void UseFixedModel (int val)

Parameters

[in] val - Scanner type.

Can be one of the following values:

- 0: No Scanner.
- 1: ScanShell® 600.
- 2: ScanShell® 800.
- 3: ScanShell® 800N.
- 4: ScanShell® 1000.
- 5: ScanShell® 2000.
- 6: ScanShell® 2000N.
- 7: ScanShell® 800E.
- 8: ScanShell® 800EN.
- 9: ScanShell® 3000D.
- 10: ScanShell® 4000.
- 11: ScanShell® 800G.
- 12: ScanShell® 5000.
- 13: SnapShell® (IDR)
- 14: ScanShell® 800DX
- 15: ScanShell® 800DXN
- 16: SnapShell® (FDA)
- 17: SnapShell® (WMD)
- 18: SnapShell® (TWN)
- 19: CSSN_PASS
- 20: CSSN_RTE8K
- 21: CSSN_TWAIN_N
- 22: CSSN_MAGTEK_STX
- 23: CSSN_CLBS

Remarks
Setting this value will set the SDK to work only with the given scanner type.

GetLibVersion

Format

Void GetLibVersion(char *version)

Parameters

[in] version- Null terminated string that holds the Slib version.
Remarks
Returns the Slib version.

GetPressedButton

Format

```c
short GetPressedButton()
```

Parameters
None.

Remarks
Returns the button number that was pressed on the scanner.

InitScanLibEx

Format

```c
short InitScanLibEx(const char *lpszLicense, int ownerId, const char* lpszOwnerName)
```

Parameters
- [in] szLicense - Null terminated string that holds the license key value.
- [in] OwnerID - An ID to the parent application.
- [in] OwnerName - The name of the parent application.

Return Value
- LICENSE_VALID: License is valid and the library is ready to be used.
- LICENSE_INVALID: The license is invalid. All scanner operations are disabled.
- LICENSE_EXPIRED: License has expired. All scanner operations are disabled.
- SLIB_ERR_DRIVER_NOT_FOUND: The scanner driver was not found. To fix this error re-install the scanner's driver. All scanner operations are disabled.
- SLIB_ERR_SCANNER_NOT_FOUND: The scanner is not connected to the PC. To fix this error make sure the scanner is connected and re-start the function. All scanner operations are disabled.
- SLIB_LIBRARY_ALREADY_INITIALIZED: The library already was initialized by another application.

Remarks
Use this function to initialize the scanner library. This function loads the scanner driver and initializes the internal image structure. This function must be called before calling any other function in the library. This function allows you to develop more than one application using the SDK and to control each of the application uses of the SDK. Only one application can use the SDK at a time.

GetOwnerData

Format

```c
short GetOwnerData(char *data)
```

Notes from the Sdk
Parameters
[in] data - Null terminated string that holds the owner application name.

Remarks
Returns the current application name that initialized the SDK using the InitScanLibEx function.

UnInitScanLibEx

Format

```
short UnInitScanLibEx(int ownerId)
```

Parameters
[in] OwnerID - An ID to the parent application.

Return Value

- SLIB_UNLOAD_FAILED_BAD_PARENT: Cannot unload the driver since another application is using/ has loaded it.
- SLIB_NOT_INITIALIZED: The driver is not found in the memory (hence it 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 InitScanLibEx) before initializing the SDK from the other application.

UnloadSnapServer

Format

```
short UnloadSnapServer()
```

Parameters
None.

Return
None (Ignore all return values).

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.
TotalConnectedScanners

Format

int TotalConnectedScanners ()

Parameters

None.

Return

The number of all connected SnapShell® cameras.

Remarks

Retrieves the total number of the connected SnapShell® cameras.

AssignScanner

Format

int AssignScanner (int scannerIndex)

Parameters

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

Return

The USB port number that the SnapShell® is connected to.

Remarks

Use a loop to call this function to get all the SnapShell® devices and save it in a local array to call each SnapShell® device later in your application.

ReleaseScanner

Format

int ReleaseScanner (int scannerIndex)

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.
SetCurScanner

Format

```
int SetCurScanner (int scannerIndex)
```

Parameters

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

Return

None.

Remarks

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

GetCurScanner

Format

```
int GetCurScanner ()
```

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 (char *szSource, int bufLen)
```

Parameters

- [In-out] szSource 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 is connected to your system. If this function returns SLIB_ERR_NO_NEXT_VALUE, then no more twain scanners are detected and if the return value is
SLIB_TRUE, then there are more twain scanners found and you can call GetNextTwainSource function to get all the twain scanners list.

**GetNextTwainSource**

*Format*

\[
\text{GetNextTwainSource (char *szSource, int bufLen)}
\]

*Parameters*

[In-out] szSource 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 and if the return value is SLIB_TRUE, then there are more twain scanners found and you can call this function again to get all the twain scanners list.

**GetTwainScanner**

*Format*

\[
\text{GetTwainScanner (char *szSource, int bufLen)}
\]

*Parameters*

[In-out] szSource 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*

\[
\text{SetTwainScanner (const char *szStr)}
\]

*Parameters*

[In] szStr as string.

*Return*

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

*Remarks*
Use this function to set the current twain scanner that is connected to your system. Use this function before initializing the library.

**UseFixedModel**

**Format**

```
UseFixedModel (int val)
```

**Parameters**

[in] - val device index.

**Return**

None.

**Remarks**

Make the Slib in order to use fix model device.

**SetCentered**

**Format**

```
SetCentered (short val)
```

**Parameters**

[in] - align mode: 1 = center, 0 = not center.

**Return**

None.

**Remarks**

Set the alignment of the captured image to be centered or not.

**SetRemoteIP**

**Format**

```
SetRemoteIP (char* pRemoteIP, int nPort, int nModel)
```

**Parameters**

[in] - pRemoteIP - A null terminating string containing the IP Address of the remote scanner.
[in] - nPort - An integer containing the port which the Remote scanner listens to.
[in] - nModel - The model of the scanner that the Remote scanner should use.

**Return**

None.
Remarks
Slib enables the usage of a scanner that is connected to a different machine. This machine should have IpScan installed and running.

Once IpScan 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 Slib API to activate it.

The following code demonstrates the usage of the IpScan from SLIB API:

```
UseFixedModel(CSSN_IP);
SetRemoteIp ("192.168.2.103", 8888, CSSN_NONE);
InitScanLibEx(LICENSE, 0, "IdScan");
```

From now on, each call to Slib API will be referring to the remote Client and not to your local machine.

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**

**Format**

```
Int IsRemoteScannerValid ()
```

**Return**
The function returns the scanner type connected to IpScan or -1 when the scanner is not connected.

**Remarks**
Use this function to test if the scanner that is connected to IpScan is valid. This function does not test the status of the communication with IpScan, for that use IsScannerValid().

**UpdateRemoteScanner**

**Format**

```
bool UpdateRemoteScanner(char* pFolder)
```

**Return**
The function is relevant to IpScan only. It will return true if the files were transferred successfully to the remote controller.

**Remarks**
UpdateRemoteScanner updates the remote scanner controller with the version that exists in pFolder. Once it is called, Slib will send the files to the remote controller and the remote controller will be upgraded.

**IsUpdateSuccessful**

**Format**

```
Int IsUpdateSuccessful ()
```


Return
The function is true if the update was successful.

Remarks
Call this function after a successful call to UpdateRemoteScanner.

GetRemoteScannerVersion

Format

```
Int GetRemoteScannerVersion ()
```

Return
The function returns the version number of the remote scanner, or -1 if SLIB is not connected to a remote scanner.

ScanBmpFileAsync

Format

```
Int ScanBmpFileAsync (const char *szFileName,int nDataPortNumber=-1,int nThumbImageResolution=-1,int nThumbImageJpgQuality =-1);
```

Return
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 SLIB. 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 image.

GetJobList

Format

```
Int short GetJobList(int nSize,int* pJobs);
```

GetJobList fills the array pJobs with the job numbers that are still in the queue. The function returns the number of Jobs that are still in the queue.

pJobs – A pointer to an array in which the job numbers will be stored.
nSize – The size of the array supplied to the function.
GetJobStatus

Format

short GetJobStatus(int nJob);

Return
The function returns the status of the job specified by nJob.
The status can be one of the following:

#define JOB_PENDING 0
#define JOB_DONE 1
#define JOB_THUMB_READY 2
#define JOB_NOT_FOUND 3
#define JOB_TIMEOUT 4
#define JOB_FAILED 5

Once a job has a status as follows: JOB_DONE, JOB_TIMEOUT or JOB_FAILED, and is queried, it will be deleted from the job list.

ClearJobList

Format

short ClearJobList();

Return
The return value is 0. The function clears the jobs that are still in the queue.

CountCSSNDevices

Format

short CountCSSNDevices();

Return
The function returns the number of CSSN devices connected to the machine or to the remote machine.

GetCSSNDevice

Format

void GetCSSNDevice (int nIndex, char* pStr, int nBufLen);
Return

The function copies the name of the CSSN devices connected to the machine or to the remote machine into pStr.

nIndex - The index of the device.
pStr - Pointer to a null terminated string that will contain the device name.
BufLen - 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.

GetSnapSerial

Format

```c
int GetSnapSerial(char* pSerial, int nBufLen)
```

Return

This one returns the serial number of the SnapShell® scanner.

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.

SetExternalTriggerMode

Format

```c
int SetExternalTriggerMode(bool bVal);
```

Description

This one sets the mode of the 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

```c
int SetExternalTrigger();
```
Description

Call this function 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

```c
int SetExternalOperation();
```

Description

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.

IdCard Library Functions

Library IdCard.dll: General Functionality

IdCard library is used to analyze and extract data from the recently scanned driver’s license image that is stored in the internal image buffer. The library also supports the extraction of the face and signature images from the general license image file and saves them to separate image files.

CountySupportAutoDetect

Format

```c
short CountySupportAutoDetect (short countryId)
```

Parameters

[in] countryId - Constant value of the country.

Return Value

ID_TRUE: The country supports state auto detection.
ID_FALSE: The country does not support state auto detection.

Remarks

The state auto-detection feature is not implemented on all the supported countries. Use this function to determine which countries can use the DetectState function.

DetectState

Format

```c
short DetectState(const char *reserved)
```
Parameters

[in] reserved - Empty string (""").

Return Value

LICENSE_INVALID: The license is invalid. All scanner operations are disabled.
ID_ERR_USA_TEMPLATES_NOT_FOUND: The template database file for the USA states (UsaIds.bin) is missing. The file should be located in the SDK directory.
INVALID_INTERNAL_IMAGE: No internal image is loaded. This value returns when attempting to use this function without scanning an image first.
ID_ERR_STATE_NOT_SUPPORTED: The license image does not match any state template.
ID_ERR_STATE_NOT_RECOGNIZED: The license image does not match any state template.

If none of the above error values is returned, the function returns the state ID value.

Remarks

Use this function to automatically detect the state type according to the image. If the function returns with none of the above error values, then the return value is the state ID. This value can be assigned to the input parameter IdState in the function ProcessState for data extraction.

Note:
This function analyzes the image in the internal buffer and determines the issuing state.

DetectStateEx

Format

short DetectStateEx(const char *reserved, short *angle)

Parameters

[in] reserved - Empty string (""").
[out] angle - The number of times that the image was rotated.

Return Value

LICENSE_INVALID: The license is invalid. All scanner operations are disabled.
ID_ERR_USA_TEMPLATES_NOT_FOUND: The template database file for the USA states (UsaIds.bin) is missing. The file should be located in the SDK directory.
INVALID_INTERNAL_IMAGE: No internal image is loaded. This value returns when attempting to use this function without scanning an image first.
ID_ERR_STATE_NOT_SUPPORTED: The license image does not match any state template.
ID_ERR_STATE_NOT_RECOGNIZED: The license image does not match any state template.

If none of the above error values is returned, the function returns the state ID value.

Remarks

Use this function to automatically rotate the internal image and then to detect the state type. If the function returns with none of the above error values, then the return value is the state ID. The function also loads the parameter angle with the amount of clockwise 90 degrees hops that took to align the image horizontally.
This value can be one of the following:
ANGLE_0: The image was not rotated
ANGLE_90: The image was rotated once by 90 degrees clockwise.
GetAccuracy

Format

short GetAccuracy()

Return Value
A value in the range of 0-100 that reports the analysis accuracy.

Remarks
The accuracy value is obtained by comparing the expected data length and the retrieved data length in specific fields. For example, when processing the ISSUE DATE field, the expected data should be in the format of “mm-dd-yy” (6 chars). If the retrieved data is “mm- -yy”, then the detection value (or accuracy) is 4/6 = 66%. Averaging this value with the rest of the fields yields the overall detection accuracy.

GetCntryNameById

Format

short GetCntryNameById(short countryID, char *szCountryName)

Parameters
[in] countryID - Constant value of the country.
[out] szCountryName - The country name.

Return Value
ID_ERR_NO_MATCH: The input country ID value does not match any country value.
ID_TRUE: Found country constant that matches and the input parameter countryID. The country string is copied to szCountryName.
Country id number: Matching country ID value.

Remarks
This function converts between the country ID numeric value (constant short) and its textual name (string). The conversion can be from constant to string or from string to constant. The function examines the value of the countryID. If this parameter is equal to -1, then the function takes the state name from the parameter szCountryName and returns the country ID. If the string does not match any of the country names, the function returns ID_ERR_NO_MATCH. If the parameter countryID is not -1, then the function loads the string szCountryName with the state textual name and the function returns ID_TRUE. If the countryID does not match any of the country constants, then the function returns ID_ERR_NO_MATCH. The string comparisons are all case sensitive.
GetCountryByState

Format

short GetCountryByState(int stateId)

Parameters

[in] stateId - Constant value that represents the state.

Return Value

ID_ERR_NO_MATCH: The input state ID value does not match any known state value.
Country id number: The country ID value that contains the state.

Remarks
Use this function to find what country the state belongs to. If the input value does not match any known state, then the function returns ID_ERR_NO_MATCH.

GetFace

Format

short GetFace(const char *szSourceFile, const char * szDestFile, short stateID)

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.

Return Value

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
Use this function to extract the face image from the document image. 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 szSourceFile 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. The destination file name must be with a BMP extension.
GetFaceEx

Format

GetFaceEx(const char *szSourceFile, const char *szDestFile, short stateID, short ImageType)

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 Value

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_DESTINATION: 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 it 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.

GetFaceEx

Format

GetFaceAndSignatureOnly(const char *SourceFile, const char *FaceDestFile, const char *SignatureDestFile, short stateID, short ImageType)

Parameters

[in] szSourceFile - Full name of the source driver's license image file.
[in] szFaceDestFile - Full name of the destination face image.
[in] szSignatureDestFile - Full name of the destination signature image.
[in] stateID - The state ID.
[in] ImageType - The face image type to extract.

Return Value

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_DESTINATION: Copying face image file to destination file failed.
**Remarks**

This function is similar to the GetFace function but it 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.

**GetFaceDuplex**

**Format**

```c
short GetFaceDuplex(const char *szSourceFile, const char *szBackSourceFile, const char* szDestFile, short stateID)
```

**Parameters**

- `[in]` `szSourceFile` - Full name of the source driver's license image file.
- `[in]` `szBackSourceFile` - Full name of the source driver's license image file second side.
- `[in]` `szDestFile` - Full name of the destination face image.
- `[in]` `stateID` - The state ID.

**Return Value**

- **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_DESTINATION**: 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 for Duplex scan support that gives you the option to get the face image without knowing where it is located on the card (front or back side). Duplex scanning is available only with ScanShell® 800DX, 800DXN, 3000D, 5000.

Use this function to extract the face image from the document image. 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 `szSourceFile` is empty, 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. The destination file name must be with a BMP extension.

**GetField**

**Format**

```c
short GetField(short fieldIndex, char *szBuf)
```
Parameters
[in] fieldIndex - Constant value of the field.
[out] szBuf - Destination buffer to receive the field value.

Return Value
ID_ERR_BAD_PARAM: Bad field index.
ID_TRUE: Function completed successfully.

Remarks
Use this function to retrieve a field string. This function is called after the call to retrieve the fields data extracted by the ProcessState function.

GetFirstCntry

Format
short GetFirstCntry()

Return Value
The value of the first country in the countries list.

Remarks
Use this function to get the first country in the countries list. Combining this function with the function GetNextCntry allows you to obtain the constant values of all supported countries. The countries constant values in the region ARE NOT always consecutive and should be obtained using the function GetNextCntry.

See Also
GetNextCntry()

GetFirstStateByCntry

Format
short GetFirstStateByCntry(short country)

Parameters
[in] country - Constant value of the country.

Return Value
ID_ERR_NO.Match: Bad country constant.
ID_TRUE: Function completed successfully.

Remarks
Use this function to retrieve the first state constant in the country. The states constant values in the country ARE NOT always consecutive and should be obtained using the function GetNextStateByCntry.

See Also
GetNextStateByCntry()

GetLastField

Format

short GetLastField ()

Return Value
The last field index.

Remarks
Use this function to obtain the last field index. This function can be used to iterate all the field indexes and obtain their values.

GetNextCntry

Format

short GetNextCntry ()

Return Value
- ID_ERR_COUNTRY_NOT_INIT: Returned if the GetFirstCntry function was not called before.
- ID_ERR_NO_MATCH: Returned if the list has an internal error.
- ID_ERR_NO_NEXT_COUNTRY - Returned if this country is the last country in the list.
- Returns the next country constant.

Remarks
Use this function to obtain the next country in the country list. To obtain the countries list call GetFirstCntry once to obtain the first country. Then continuously call GetNextCntry until the value ID_ERR_NO_NEXT_COUNTRY is returned.

See Also
GetFirstCntry()

GetNextStateByCntry

Format

short GetNextStateByCntry(short country)
Parameters

[in] country - Constant value of the country.

Return Value

- **ID_ERR_NO_MATCH**: Bad country constant.
- **ID_ERR_COUNTRY_NOT_INIT**: Returned if the GetFirstCntr function was not called before.
- **ID_ERR_NO_NEXT_STATE** - Returned if this state is the last state of the country state list.
- Returns the next state constant.

**Remarks**

Use this function to obtain the next state in the list.

**See Also**

GetFirstStateByCntry().

---

**GetSignature**

**Format**

```c
short GetSignature(const char *szSourceFile, const char *szDestFile, short stateID)
```

**Parameters**

- `[in]` `szSourceFile` - Full name of the source driver's license image file.
- `[in]` `szDestFile` - Full name of the destination signature image.
- `[in]` `stateID` - The state ID.

**Return Value**

- `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_DESTINATION`: Copying face image file to destination file failed.
- `ID_ERR_FACE_IMAGE_NOT_FOUND`: Extraction of the signature image failed - could not locate the signature rectangle.
- `ID_TRUE`: Function completed successfully.

**Remarks**

Use this function to extract the signature image from the document image. 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 `szSourceFile` 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. The destination file name must be with a BMP extension.

---

**GetSignatureDuplex**

**Format**

```c
short GetSignatureDuplex(const char *szSourceFile, const char *szBackSourceFile, const char* szDestFile, short stateID)
```

**Parameters**

- `[in]` `szSourceFile` - Full name of the source driver's license image file.
- `[in]` `szBackSourceFile` - Full name of the source driver's license image file second side.
- `[in]` `szDestFile` - Full name of the destination signature image.
- `[in]` `stateID` - The state ID.
Return Value

- **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_DESTINATION**: 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 for Duplex scan support that gives you the option to get the face image without knowing where it is located on the card (front or back side). Duplex scanning is available only with ScanShell® scanners 800DX\800DXN, 3000D, 5000.

Use this function to extract the face image from the document image. 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 `szSourceFile` 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. The destination file name must be with a BMP extension.

GetStateNameById

Format

```c
short GetStateNameById(short stateID, char *szStateName)
```

Parameters

- `[in]` `stateID` - Constant value of the state.
- `[out]` `szStateName` - The state name.

Return Value

- **ID_ERR_NO_MATCH**: The input state ID value does not match any state value.
- **ID_TRUE**: Found state constant that matches and the input parameter `stateID`. The country string is copied to `szName`.

State id number: Matching country ID value.

Remarks

This function converts between the state ID numeric value (constant short) and its textual name (string). The conversion can be from constant to string or from string to constant. The function examines the value of `stateID`. If this parameter is equal to -1, then the function takes the state name from the parameter `szStateName` and returns the state ID. If the string does not match any of the state names, the function returns **ID_ERR_NO_MATCH**. If the parameter `stateID` is not -1, then the function loads the string `szStateName` with the state textual name and the function returns **ID_TRUE**. If the `stateID` does not match any of the state constants, then the function returns **ID_ERR_NO_MATCH**.

The string comparisons are all case sensitive.
GetStateShort

Format

```c
short GetStateShort(int stateID, char *szStateName)
```

Parameters

- [in] stateID - Constant value of the state.
- [out] szStateName - The state name.

Return Value

- **ID_ERR_NO_MATCH**: The input state ID value does not match any state value.
- **ID_TRUE**: Found state constant that matches.

Remarks

This function takes the stateID constant and loads szStateName with the state abbreviated name. For example, if the input value is 0 (the constant value of the state Alabama), then the function loads szStateName with the string “AL”.

InitIdLib

Format

```c
short InitIdLib(const char *szLicense)
```

Parameters

- [in] szLicense - The library license key.

Return Value

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

Remarks

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

ProcessState
Format

short ProcessState(const char *szReserved, short stateID, unsigned long reserved)

[in] stateID - Constant value of the state.

Return Value

If the function succeeds, the return value is ID_TRUE.
If the function fails, the return value is one of the following:

- LICENSE_INVALID – Library was not initialized with the proper license.
- SLIB_ERR_SCANNER_NOT_FOUND – No attached scanner was found.
- LICENSE_INVALID - The library was not initialized with the proper license.
- SLIB_ERR_INVALID_SCANNER - No scanner was found attached to the PC.
- ID_ERR_STATE_NOT_SUPPORTED - The requested state ID is not supported.
- INVALID_INTERNAL_IMAGE - No internal image is loaded. This value returns when attempting to use this function without scanning an image first.

Remarks

Use this function to process the internal image acquired in the last scan. The function de-skews and cleans the image and then passes to the OCR for analysis. The resultant textual data is kept in internal structure ready for retrieval by the GetField function. Processing the image does not modify the image content.

Successful image processing depends on the following:

a. The image must be scanned in 24 bit (true color) and 300 dpi.
b. The image must be aligned horizontally with tolerance of no more than ±10 degrees.

GetCntryNameById

Format

short GetCntryNameById(short countyID, char *szCountryName)

Parameters

[in] szCountryName - Country name as retrieved from the SDK.

Return

1) ID_ERR_NO_MATCH: The parameter CountryName does not contain known country name.
2) Country ID number.

Remarks

Use this helper function to get the country ID number.
SetRegion

Format

```
short SetRegion(int region)
```

Parameters

[in] `region` - The current region ID.

Return Value

- **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 DetectStateEx (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 A.

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

SetRegionDetectionSequence

Format

```
short SetRegionDetectionSequence(int region0, int region1, int region2,
                                 int region3, int region4, int region5, int region6)
```

Parameters

[in] `RegionId0 - 6` - The region ID’s.

Return Value

- **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 in this function.

**Example**: Call this function with 0 & 1 for USA & Canada and now whenever you call the function DetectStateEx with USA or Canadian cards, it will find the card.
GetFirstRegion

Format

```
short GetFirstRegion()
```

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 GetNextRegion().

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 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 functions GetFirstRegion and GetNextRegion.
Not all the regions support auto detection. However, in the following versions more and more regions will support this feature.

GetNextRegion

Format

```
short GetNextRegion()
```

Return
ID_ERR_NO_MATCH: The function GetFirstRegion 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 1, 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 GetFirstRegion - this is called once.
Call GetNextRegion 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.

GetRegionNameById

Format

```
short GetRegionNameById(short id, char *name)
```

Smart from the start

acuantcorp.com

6960 Bristol Parkway
Culver City, California 90230

310.961.2621
Parameters

[in] id - The region numeric value.
[out] name - A string that accepts the region name.

Return

- ID_ERR_NO_MATCH – The parameter id is not a valid region enum.
- ID_TRUE – Region found. The name 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 GetFirstRegion and GetNextRegion, builds the region name list.

GetRegionNameById

Format

short GetRegionNameById(short id, char *name)

Parameters

[in] name - The region Name.
[out] id - An integer that accept the region ID.

Return

- ID_ERR_NO_MATCH – The parameter name is not a valid region name.
- ID_TRUE – Region found. The id returns loaded with the region ID.

Remarks

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

AutoDetectSupport

Format

short AutoDetectSupport(int region)

Parameters

[in] region - The region enum value.

Return Value

- ID_ERR_NO_MATCH – The input parameter Region value does not contain a valid region enum.
- **ID_FALSE** - The parameter Region represents a region that does not support auto detection.
- **ID_TRUE** - The parameter Region 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 `DetectStateEx` after scanning the document. If so, `DetectStateEx` will return a proper state enum, and this value can be used as an input parameter to the function `ProcessState` that retrieves the data from the image.

If `AutoDetectSupport` returns **ID_FALSE**, then you should skip the call to `DetectStateEx` and call `ProcessState` directly while setting the state ID parameter manually.

### GetRegionByCountry

**Format**

```c
short GetRegionByCountry(int countryId)
```

**Parameters**

[in] `countryId` - Constant value of a country.

**Return Value**

**ID_ERR_NO_MATCH** - The parameter CountryId 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 A.

### GetFirstCountryInRegion

**Format**

```c
short GetFirstCountryInRegion(int region)
```

**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.

Otherwise, the function returns the first country ID in the region.
Remarks
Use this function (combined with GetNextCountryInRegion) to obtain the list of countries in the region.

GetNextCountryInRegion

Format

short GetNextCountryInRegion(int region)

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 GetFirstCountryInRegion) to obtain the list of countries in the region. You should call GetFirstCountryInRegion once, than call GetNextCountryInRegion 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:

1. Retrieve the full region list using GetFirstRegion and GetNextRegion functions.
2. For each region, retrieve the full countries list using GetFirstCountryInRegion and GetNextCountryInRegion functions.
3. For each Country, retrieve the full state list using GetFirstStateByCntry and GetNextStateByCntry functions.

GetFaceImgBuffer

Format

short GetFaceImgBuffer(LPCSTR sSource, BSTR *pBuf, LPCSTR sFileType, int nStateID)

Parameters
[in] Source - Full name of the source driver's license image file.
[in] sFileType - Set the buffer image format. May be one of the following strings (case insensitive):
“BMP”
“JPG”
GetSignatureImgBuffer

Format

short GetSignatureImgBuffer(LPCSTR sSource, BSTR *pBuf, LPCSTR sFileType, int nStateID)

Parameters

[in] **Source** - Full name of the source driver's license image file.
[in] **sFileType** - Set the buffer image format. May be one of the following strings (case insensitive):
  “BMP”
  "JPG"
  “PNG”
  “TIFF”
  “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.

Smart from the start
DetectProcessAndCompare

Format

```c
short DetectProcessAndCompare(const char *ImageA, const char *ImageB,
int stateID, int& ImageAassignment, int&angleA, int& angleB, DWORD
reserved);
```

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 **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 CidData and Barcode classes (Using ScanShell® 3000D\ScanShell® 800DX only).

DetectProcessAndCompare2

Format

```c
short DetectProcessAndCompare2(const char *ImageA, const char *ImageB,
int stateID, int& ImageAassignment, int&angleA, int& angleB,
defaultBarcode int, DWORD reserved);
```

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 **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 CidData and Barcode classes (Using ScanShell® 3000D\ScanShell® 800DX only).

DetectProcessDuplex

Format

short DetectProcessDuplex (const char *ImageA, const char *ImageB, int stateID, int & ImageAassignement, int &angleA, int & angleB, DWORD 
Preserved);

Parameters

[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 assignement.
[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).

ProcMRZ

Format

short ProcMRZ (const char *FileName, int rotationAngle);

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
Void ResetIDFields ()

Parameters
None.

Return
None.

Remarks
Use this function to clear the fields data from the IDData class.

GetIntMRZChecksumVerified

Format
Int GetFieldMRZChecksumVerified (short index)

Parameters
[in] Index - 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).

SetDatesFormat

Format
void SetDatesFormat(short val)

Parameters
[in] 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)

Remarks
Use this function to extract the date's fields with the format you need. This property will take effect on passport date fields as well.
GetFaceAndSignatureImagesOnly

Format

void GetFaceAndSignatureImagesOnly(const char *SourceFile, const char *FaceDestFile, const char *SignatureDestFile, short stateID, short ImageType)

Parameters

[in] SourceFile - Full name of the source driver's license image file.
[in] FaceDestFile - Full name of the destination face image.
[in] SignatureDestFile - 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_DESTINATION: 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 GetFaceEx 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.

1.1.104 SetBrightnessForFaceImage

Format

void 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.

1.1.105 SetFieldToBeErasedFromImage

Format

void SetFieldToBeErasedFromImage(int nFieldIndex,bool bErase)

Parameters

[in] nFieldIndex - The index of the field to be erased.
[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.

Library Barcode: General Functionality
BarCode library functionality has a similar functionality to the idData library. It extracts the data from 2D, PDF417 type bar code images. The library fetches the internal image (the last scanned image), processes its graphic information and activates its internal image analyzer. The resultant text is kept in internal data structure ready to be retrieved by the application.

Notice: The library processes only the last image that was recently scanned. The library does not support external image file processing.

NOTE: The image MUST be scanned in 600dpi, gray scale (256 shades of gray) format.

Barcode Library Functions

1.1.106 InitBCLib

Format

short InitBCLib(const char *szLicense)

Parameters

[in] szLicense - The library license key.

Return Value

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

Remarks

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

1.1.107 ProcessBC

Format

short ProcessBC (const char *reserved0, long reserved1)

Parameters

[in] reserved0 - Empty string ("").
Return Value

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

If the function fails, the return value is one of the following:

- **LICENSE_INVALID** – Library was not initialized with the proper license.
- **SLIB_ERR_SCANNER_NOT_FOUND** – No attached scanner was found.
- **LICENSE_INVALID** – The library was not initialized with the proper license.
- **SLIB_ERR_INVALID_SCANNER** – No scanner was found attached to the PC.
- **ID_ERR_STATE_NOT_SUPPORTED** – The requested state ID is not supported.
- **INVALID_INTERNAL_IMAGE** – No internal image is loaded. This value returns when attempting to use this function without scanning an image first.
- **BC_ERR_NO_BC_FOUND** – No barcode pattern (PDF417) was found on the image.

Remarks

Use this function to process the internal image and extract 2D barcode information acquired in the last scan. The function deskews and cleans the image and then passes to the image analyzer for data extraction. The resultant textual data is kept in internal structure ready for retrieval by either `RefreshData()` or `GetRawText()` functions. Processing the image does not modify the image content.

Successful image processing depends on the following:

- The image must be scanned in 600 dpi – Gray scale color scheme.
- The image must be aligned in such way that the bar code image is vertical with tolerance of no more than ±10 degrees.

**Note**: This function is now obsolete and is replaced by `ProcAllBarcodes()`. It is supported only for backward compatibility.

### 1.1.108 Process1DBC

**Format**

```c
short Process1DBC (const char *reserved0, long reserved1 )
```

**Parameters**

- **reserved0** – Empty string ("").
- **reserved1** – The value 0.

**Return Value**

- If the function succeeds, the return value is **BC_ERR_NONE**.
- If the function fails, the return value is one of the following:
  - **LICENSE_INVALID** – Library was not initialized with the proper license.
  - **SLIB_ERR_SCANNER_NOT_FOUND** – No attached scanner was found.
  - **LICENSE_INVALID** – The library was not initialized with the proper license.
  - **SLIB_ERR_INVALID_SCANNER** – No scanner was found attached to the PC.
  - **ID_ERR_STATE_NOT_SUPPORTED** – The requested state ID is not supported.
  - **INVALID_INTERNAL_IMAGE** – No internal image is loaded. This value returns when attempting to use this function without scanning an image first.
  - **BC_ERR_NO_BC_FOUND** – No barcode pattern (PDF417) was found on the image.
Remarks

Use this function to process the internal image and extract 1D barcode information acquired in the last scan. The function deskews and cleans the image and then passes to the image analyzer for data extraction. The resultant textual data is kept in internal structure ready for retrieval by either RefreshData() or GetRawText() functions. Processing the image does not modify the image content.

Successful image processing depends on the following:

c. The image must be scanned in 600 dpi – Gray scale color scheme.
d. The image must be aligned in such way that the barcode image is vertical with tolerance of no more than ±10 degrees.

Note: This function is now obsolete and is replaced by ProcAllBarcodes(). It is supported only for backward compatibility.

1.1.109 ProcAllBarcodes

Format

short ProcAllBarcodes (const char *reserved0, long reserved1 )

Parameters

[in] reserved0 - Empty string (""").
[in] reserved1 - The value 0.

Return Value

If the function succeeds, the return value is one of the following:

- BC_ERR_NO_BC_FOUND (0) – No barcode found on the card.
- BC_2D_BC_FOUND (0x2) – 2D barcode found on the card.
- BC_1D_BC_FOUND (0x1) – 1D barcode found on the card.

Notice that if the image contains 1D and 2D barcodes on it, the return value will be the OR operation of BC_1D_BC_FOUND and BC_2D_BC_FOUND – i.e., the value 3.

If the function fails, the return value is one of the following:

- LICENSE_INVALID – Library was not initialized with the proper license.
- SLIB_ERR_SCANNER_NOT_FOUND – No attached scanner was found.
- LICENSE_INVALID - The library was not initialized with the proper license.
- SLIB_ERR_INVALID_SCANNER - No scanner was found attached to the PC.
- ID_ERR_STATE_NOT_SUPPORTED - The requested state ID is not supported.
- INVALID_INTERNAL_IMAGE - No internal image is loaded. This value returns when attempting to use this function without scanning an image first.
- BC_ERR_NO_BC_FOUND - No barcode pattern (PDF417) was found on the image.

Remarks

Use this function to process the internal image and extract all the barcodes on the image (1D and 2D). The function deskews and cleans the image and then passes to the image analyzer for data extraction.
The resultant textual data is kept in internal structure ready for retrieval by either RefreshData() or GetRawText() functions. Processing the image does not modify the image content.

Successful image processing depends on the following:
   e. The image must be scanned in 600 dpi – Gray scale color scheme.
   f. The image must be aligned in such way that the bar code image is vertical with tolerance of no more than ±10 degrees.

1.1.1.10 BarcodeDetectionQuality

Format

int BarcodeDetectionQuality( )

Parameters
None.

Return Value
If the function succeeds, the return value is one of the following:
   • BC_DETECTION_NO_BARCODE_FOUND (0) – No barcode found on the card.
   • BC_DETECTION_WITH_NO_ERRORS (1) – The barcode(s) was detected successfully with no errors.
   • BC_DETECTION_WITH_TOO_MANY_ERRORS (4) – A barcode was found on the card but was not retrieved successfully due to poor image quality (too many checksum errors).
   • BC_DETECTION_WITH_CHECKSUM_ERRORS (8) – A barcode was found on the card and was read successfully with some checksum errors. This means that although most of the fields were extracted successfully, some of the fields contain errors.

Remarks
Use this function to decide if the retrieved information was detected accurately.

1.1.1.11 Refresh

Format

Void Refresh ( )

Parameters
None.

Remarks
This function takes the raw data (obtained by ProcessBC()) and parses it according to the AAMVA standard. The result data fields (name, address, license number, etc.) can be retrieved later on using the function GetBCField().

1.1.1 RefreshFields
Format

```cpp
bool RefreshFields()
```

Parameters

None.

Return Value

If the function returns a non-zero value, the data was retrieved successfully.
If the function returns a zero value, the data was retrieved un-successfully.

Remarks

This function takes the raw data (obtained by `ProcessBC()`) and parses it according to the AAMVA standard. The result data fields (name, address, license number, etc.) can be retrieved later on using the function `GetBCField()`. 

*Note this function is the same as “Refresh” the only difference is that it has a return value.*

1.1.112 GetRawText

Format

```cpp
void GetRawText(char *buffer)
```

Parameters

- `[out]` *buffer* – A pointer to a 2048 characters buffer.

Return Value

None.

Remarks

This function copies the raw data that was extracted from the barcode analyzer without further parsing. The buffer should be able to allocate up to 2048 characters. This function is useful to obtain data from general-purpose documents that use the PDF417 standard to export data.

1.1.113 GetBCField

Format

```cpp
void GetBCField(short fieldIndex, char *buffer)
```

Parameters

- `[in]` *index* – The field index.
- `[out]` *buffer* – A pointer to a 60 character buffer that will be loaded with the data.

Return Value

`BC_ERR_BAD_PARAM` – Unknown field index number.
BC_ERR_NONE – Field extracted successfully.

Remarks
This function can be called repetitively (each time with a different field index) to extract the value of a specific field into the buffer. Call this function after calling Refresh(). The field indexes are described in Appendix A.

Library SOCRdll: General Functionality

SOCRdll provides basic text extraction from an image file. The image file format must have a resolution of 300 dpi. The image may be in either color or black and white color scheme.

SOCRdll Library Functions

1.1.114  InitOcrLib

Format

short InitOcrLib(const char *license, unsigned int reserved)

Parameters
[in] license - The library license key.

Return Value
LICENSE_VALID: License is valid and the library is ready to be used.
LICENSE_INVALID: The license is invalid. All scanner operations are disabled.
LICENSE_EXPIRED: License has expired. All scanner operations are disabled.
LICENSE_DOES_NOT_MATCH_LIBRARY: The license is invalid for this library. All library operations are disabled.
GENERAL_ERR_PLUG_NOT_FOUND: This error returns if either:
• The license is valid but no scanner is attached to the PC.
• The license is temporary but it has expired.

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

1.1.115  GetAcurateTextFromFile

Format

short GetAcurateTextFromFile(const char *fName, char *str, int len)

Parameters
[in] fName – Full path name of the original image.
[in] int – Buffer size.
Return Value

If the function succeeds, the return the value is **TOCR_OK (=0)**.
If the function fails, it returns one of the following values:

- **LICENSE_INVALID** – Library was not initialized with the proper license.
- **TOCRJOBERROR** – The OCR engine was not able to accomplish the detection process correctly.

Remarks

Use this function to extract text bulks from an image. The text size is limited to 4K (4096) characters.

1.1.116  GetTextFromFileUsingOCR

**Format**

```c
short GetTextFromFile(const char *fName, char *str, int len, BYTE type, long *pAutoRotation, const char *CustStr)
```

**Parameters**

- [in] fName – Full path name of the original image.
- [in] str - Buffer.
- [in] int - Buffer size.
- [in] type - Instruct the OCR what type of data is written in the image. This value increases the detection accuracy and speeds 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] CustStr – Char set

Return Value

If the function succeeds, the return the value is **TOCR_SUCCESS**.
If the function fails, it returns one of the following values:

- **LICENSE_INVALID** – Library was not initialized with the proper license.
- **TOCRJOBERROR** – The OCR engine was not able to accomplish the detection process correctly.

Remarks

Use this function to extract text bulks from an image. The text size is limited to 4K (4096) characters.

Library CPassport: General Functionality

CPassport analyses and retrieves data from a standard passport image. The passport image is taken using the ScanShell® 1000 scanner in either color or gray color scheme, it is then analyzed by the library and the result data is stored in the library properties. The image may be a full image of the page (3"x5") or only the lower portion of the page (1"x5").
CPassport Library Functions

1.1.117 Init

Format

short Init(const char *license)

Parameters

[in] license – The library license key.

Return Value

LICENSE_VALID: License is valid and the library is ready to be used.
LICENSE_INVALID: The license is invalid. All scanner operations are disabled.
LICENSE_EXPIRED: License has expired. All scanner operations are disabled.
LICENSE_DOES_NOT_MATCH_LIBRARY: The license is invalid for this library. All library operations are disabled.
GENERAL_ERR_PLUGIN_NOT_FOUND: This error returns if the attached scanner is not one of the following scanners:
  • ScanShell® 600
  • ScanShell® 800
  • ScanShell® 1000
MAG_ERR_NO_READER_FOUND: The magnetic reader device could not be found on any of the PC ports.

Remarks
This function initializes the library. This function must be called before any other function in the library can be used.

1.1.118 Process

Format

short Process(const char *inFile, unsigned long reserved)

Parameters

[in] inFile – The image file. If this is null then the buffer image will be used.

Return Value

If the function succeeds, the return value is:
PASS_ERR_NONE
If the function fails, the return value is one of the following:
LICENSE_INVALID - The library was not initialized with the proper license.
INVALID_INTERNAL_IMAGE – No internal image is loaded. This value returns when attempting to use this function without scanning an image first.

Remarks
Call this function to process the recently scanned passport image. When scanning the passport page using the ScanShell® 1000, the opened page should be aligned to the top right corner which yields a rotated internal image. Before processing the image, it needs to be rotated by 180 degrees (using the function LoadRotateSave).
The image should have the following properties:

**Color scheme:** Select one of the following:
- 24 bit (True color)
- 256 Gray shades

**Image size:** Select one of the following:
- 3” x 5”: This scans the full page of the passport.
- 1” x 5”: This scans only the lower portion of the page.

Once the function returns **PASS_ERR_NONE**, the library properties will be loaded with the analyzed text. Otherwise, the library property fields will be empty.

The raw data is scanned for format detection. If a specific format is detected, the data is parsed further and loads the library properties.

### 1.1.119 GetFacImage

**Format**

```
short GetFacImage(const char *DestFile)
```

**Parameters**

| In | DestFile | Null terminated string that holds the full name of the destination image file that will contain the face image from the passport. |

**Return Value**

- **PASS_ERR_NONE** - If the function succeeds, this is the return value.
- **LICENSE_INVALID** – Library was not initialized with the proper license.
- **PASS_ERR_CANNOT_DELETE_DESTINATION_IMAGE** - Returned when a file with the same name as the destination file already exists and cannot be overwritten.
- **PASS_ERR_CANNOT_COPY_TO_DESTINATION** - Returned when the destination file cannot be opened for write on the disk.
- **PASS_ERR_FACE_IMAGE_NOT_FOUND** - Could not retrieve the face image from the passport image.

**Remarks**

Use this function to extract the image rectangle of the person’s face from the source passport image. Remember that the original scanned image must be rotated by 180 degrees (so it will be aligned correctly) before this function is called. This function works properly only for 3"x5" images.

### 1.1.120 GetPassportSignature

**Format**

```
short GetPassportSignature(const char *src, const char *DestFile)
```
Parameters

[in] src - Full name of the source driver's license image file.
[In] DestFile - Null terminated string that holds the full name of the destination image file that will contain the signature image from the passport.

Return Value

PASS_ERR_NONE - If the function succeeds, this is the return value.
If the function fails, one of the following values is returned:
LICENSE_INVALID – Library was not initialized with the proper license.
PASS_ERR_CANNOT_DELETE_DESTINATION_IMAGE - Returned when a file with the same name as the destination file already exists and cannot be overwritten.
PASS_ERR_CANNOT_COPY_TO_DESTINATION - Returned when the destination file cannot be opened for write on the disk.
PASS_ERR_FACE_IMAGE_NOT_FOUND - Could not retrieve the signature image from the passport image

Remarks

Use this function to extract the image rectangle of the person's signature from the source passport image. Remember that the original scanned image must be rotated by 180 degrees (so it will be aligned correctly) before this function is called. This function works properly only for 3"x5" images.

1.1.121  GetPassportField

Format

short GetPassportField (short index, char *buf)

Parameters

[in] index - Constant value of the field.
[ out] buf - Destination buffer to receive the field value.

Return Value

PASS_ERR_BAD_PARAM: Bad field index.
PASS_ERR_NONE: Function completed successfully.

Remarks

Use this function to retrieve a field string. This function is called after the call to retrieve the fields data extracted by the ProcessPassport function.

1.1.122  ResetPassportData

Format

Void ResetPassportData ()

Parameters

None.

Return

None.
Remarks
Use this function to clear data from all the fields in the passport class.

1.1.123  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).

1.1.124  GetPassportFaceAndSignatureImageOnly

Format

short GetPassportFaceAndSignatureImageOnly(const char *src, const char *FaceDestFile, const char *SignatureDestFile)

Parameters
[in] src - Full name of the source driver's license image file.
[in] FaceDestFile - Full name of the destination face image.
[in] SignatureDestFile - Full name of the destination signature image.

Return Value
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_DESTINATION: 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 the GetPassportFaceEx function but it will extract the face and signature images.

1.1.125  IsRfidReaderExists

Format

long IsRfidReaderExists (bool* blsRfidReaderExists)
Parameters

[out] bIsRfidReaderExists - boolean pointer.

Return

Call status.

Remarks

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

Library MagLib: General Functionality

MagLib controls the magnetic reader, and it collects and analyzes its data once a card is swiped. The library scans COM1-COM16 for the existence of the magnetic reader and initializes it. Once a magnetic card is swiped, the data is parsed and it refreshes the relevant properties of the library. The library automatically detects the data format and parses it. The following driver license formats are supported:

- AAMVA
- Old DMV (California)
- Old DMV (Louisiana).

MagLib Library Functions

1.1.126 InitMagLib

Format

```
short InitMagLib (const char *license)
```

Parameters

[in] license - The library license key.

Return Value

- LICENSE_VALID: License is valid and the library is ready to be used.
- LICENSE_INVALID: The license is invalid. All scanner operations are disabled.
- LICENSE_EXPIRED: License has expired. All scanner operations are disabled.
- LICENSE_DOES_NOT_MATCH_LIBRARY: The license is invalid for this library. All library operations are disabled.
- GENERAL_ERR_PLUG_NOT_FOUND: This error returns if the attached scanner is not one of the following scanners:
  - ScanShell® 600
  - ScanShell® 800
  - ScanShell® 1000
- MAG_ERR_NO_READER_FOUND: The magnetic reader device could not be found on any of the PC ports.

Remarks

Smart from the start
This function scans COM1-COM16 and searches for the magnetic reader device. Once found, the reader is initialized and the library loads and is initialized.

1.1.127  IsMagValid

Format

short IsMagValid ()

Return Value

- MAG_ERR_NO_READER_FOUND: The reader is not connected to the PC.
- MAG_ERR_NONE: The reader is connected to the PC and functioning correctly.

Remarks
Detects if the Magnetic Reader hardware is connected and functioning. The reader is searched in the COM port found in the InitMagLib function.

1.1.128  WasMagSwept

Format

short WasMagSwept ()

Return Value

- MAG_ERR_NO_READER_FOUND: The reader is not connected to the PC.
- SERIAL_NOT_INIT: Serial port is not initialized.
- SERIAL_PORT_NOT_OPEN: Serial port could not be opened.
- SERIAL_PORT_CONFIG_FAIL: COM Port configuration failed.
- SERIAL_COM_TIMEOUT_FAIL: COM Port timeout failure.
- MAG_ERR_CARD_NOT_DETECTED: No new card swipe was detected from the last call to this function.
- MAG_ERR_CARD_DETECTED: A recent card swipe was detected and the data is available for process.

Remarks
Call this function periodically to find out if a new card swipe was performed. If no new swipe was performed, the function returns MAG_ERR_CARD_NOT_DETECTED. If the reader detects a new swipe it returns MAG_ERR_CARD_DETECTED. If the system is in error condition (due to bad initialization or disconnection of the reader from the PC), the function returns one of the other values.

1.1.129  ProcessMag

Format

short ProcessMag ()

Return Value

- 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 case no further processing is done.

Remarks
Call this function to process the recently swiped card 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. When the function returns, the internal raw data buffer is cleared.

1.1.130 ProcessMagStr

Format

```c
short ProcessMagStr (const char *str)
```

Parameters

[in] str - The buffer that will hold the raw data.

Return Value

- 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 case no further processing is done.

Remarks

Call this function to process the raw data in the `rawData` input string. The raw data is scanned for format detection. If a specific format is detected, the data is parsed further and loads the library properties. When the function returns, the internal raw data buffer is cleared.

1.1.131 GetMagRawText

Format

```c
short GetMagRawText(char *buf)
```

Parameters

[in] str - Null terminated string that receives the raw data.

Return Value

- MAG_ERR_NONE: Data retrieved successfully.
- MAG_ERR_CARD_NOT_DETECTED: Buffer is empty.

Remarks

Call this function to get the data as retrieved from the magnetic reader device without further processing.

1.1.132 GetMagField

Format

```c
```
**short GetMagField(short index, char *val)**

**Parameters**
- [in] index – Constant value of the field.
- [out] val – Destination buffer to receive the field value.

**Return Value**
- ID_ERR_BAD_PARAM: Bad field index.
- ID_ERR_NONE: Function completed successfully.

**Remarks**
Use this function to retrieve a field string. This function is called after the call to retrieve the fields data extracted by the Process function.

1.1.133 **ResetMagDevice**

**Format**
```
void ResetMagDevice()
```

**Parameters**
None.

**Return Value**
None.

**Remarks**
Use this function to reset all data and settings in the MagShell® 900 device.

1.1.134 **UnInitMagLib**

**Format**
```
void UnInitMagLib()
```

**Parameters**
None.

**Return Value**
None.

**Remarks**
Release the USB port and unutilize the magnetic library.

**Library CImageCtrl: General Functionality**
CImageCtrl library is a collection of graphic functions, capable of manipulating an image object. The image object may be loaded from an external file or the image object stored in the SLib library (which is the image of the last scanned document).

The library functions are capable of doing the following:
- **Image Rotation**: Rotating an image by 90, 180 or 270 degrees.
- **Resolution Modification**: Modifying the resolution to any value.
- **Image Color Conversion**: Converting the image to 24 bit (true color), 256 colors (gray or color) or Black and White (1 bit).
- **Concatenate two image files to a single image**: Attaching two images (horizontally or vertically) to form a single image file of both ID card sides.

The image can be exported (saved) to an external image file in any one of seven popular image formats such as BMP, JPG, TIFF, PCX, PNG, TGA and PSD. Alternatively, the image object can be exported to the clipboard and from there, be imported to other applications.

### CImageCtrl Library Functions

#### 1.1.135 InitImageLib

**Format**

```
short InitImageLib(const char *license)
```

**Parameters**

[in] license - The library license key.

**Return Value**

- **LICENSE_VALID**: License is valid and the library is ready to be used.
- **LICENSE_INVALID**: The license is invalid. All scanner operations are disabled.
- **LICENSE_EXPIRED**: License has expired. All scanner operations are disabled.
- **LICENSE_DOES_NOT_MATCH_LIBRARY**: The license is invalid for this library. All library operations are disabled.
- **GENERAL_ERR_PLUGIN_NOT_FOUND**: This error returns if either:
  - The license is valid but no scanner is attached to the PC.
  - The license is temporary but it has expired.

**Remarks**

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

#### 1.1.136 GetImgColor

**Format**

```
short GetImgColor(const char *fileName)
```
Parameters

[in] fileName – Image file name or empty string if evaluating the internal image.

Return Value

- **IM_ERR_FILE_OPEN**: Cannot open input image file.
- **INVALID_INTERNAL_IMAGE**: Internal image is invalid and cannot be analyzed.
- **IMAGE_BW**: The image has Black and White colors (1 bit image).
- **IMAGE_GRAY_256**: The image has 256 colors of gray (8 bit image).
- **IMAGE_COLOR_256**: The image has 256 colors (8 bit image).
- **IMAGE_COLOR_TRUE**: The image has 16 million colors (24 bit image).

Remarks

Use this function to obtain the image color scheme.

1.1.137 **LoadRotateSave**

Format

```c
short LoadRotateSave(const char *src, short angle, short destType, const char *dest)
```

Parameters

- **[in] src** – Full path name of the original image file. If this string is empty, the rotation is performed on the internal image.
- **[in] angle** – The angle to rotate the original image. This value can be one of the following values:
  - **ANGLE_0**: 0 degrees rotation
  - **ANGLE_90**: 90 degrees rotation
  - **ANGLE_180**: 180 degrees rotation
  - **ANGLE_270**: 270 degrees rotation
- **[in] destType** – The destination of the rotated image. This parameter may be one of two values:
  - **SAVE_TO_FILE**: Save the image to a file. The file name should be given in the DestImage parameter.
  - **SAVE_TO_CLIPBOARD**: Copy the rotated image the image to the clipboard.
  - **SAVE_TO_IR**: Saves the IR image. (If available)
  - **SAVE_TO_UV**: Saves the UV Image. (If available)
- **[in] dest** – Full path name of the destination file. This parameter is ignored if the parameter destType is set to **SAVE_TO_CLIPBOARD**. If this value is an empty string, no save will be performed.

Return Value

If the function succeeds, the return value is **IMG_ERR_SUCCESS**.
If the function fails, it returns one of the following values:

- **LICENSE_INVALID** – Library was not initialized with the proper license.
- **IM_ERR_BAD_ANGLE_0** – Bad rotation parameter.
- **IM_ERR_BAD_DESTINATION** – Bad destination parameter (the destination parameter is neither file nor clipboard).
**Remarks**

Use this function to rotate an image by 0, 90, 180 or 270 degrees and save it to a file in any one of seven formats. The manipulated image may be loaded from an external file (if src string holds a string value equal to the source image file name), or performed on the internal image buffer (if src string is empty). When using a file as the image source, it is important to use the proper file extension to indicate the image format. Proper extension types are: BMP, JPG, TIFF, PCX, PNG, TGA and PSD. If an image has an unrecognizable extension due to an error (e.g. TIFF instead of TIF), the function refers to the file as BITMAP.

After the image is rotated, it can be exported to either the clipboard or to an external image file. The destination file name may be one of the seven file formats indicated above. If an image has an unrecognizable extension due to an error (e.g. TIFF instead of TIF) the function exports to the file in a BITMAP format. The destination file name may be the same as the source file name. In such a case the new file, resulting with a rotated image, will overwrite the original file. If no destination image file name is given (empty string), no save is done.

Do not be misled by the name of this function. This function’s flexibility actually allows you implicitly to do the following:

- Use the following function call to convert an image file from one type to another:
  \[ \text{RotateImage} \left( \text{“xxx.bmp”}, \text{ANGLE}_0, \text{SAVE_TO_FILE}, \text{“xxx.jpg”} \right) \]

- Use the following function call to copy an image file to the clipboard:
  \[ \text{RotateImage} \left( \text{“xxx.bmp”}, \text{ANGLE}_0, \text{SAVE_TO_CLIPBOARD}, \text{“”} \right) \]

- Use the following function call to rotate the internal image:
  \[ \text{RotateImage} \left( \text{“”}, \text{ANGLE}_0, \text{SAVE_TO_FILE}, \text{“”} \right) \]

- Use the following function call to save the internal image to a file:
  \[ \text{RotateImage} \left( \text{“”}, \text{ANGLE}_0, \text{SAVE_TO_FILE}, \text{“xxx.bmp”} \right) \]

---

**ConvertImgFormat**

**Format**

```c
short ConvertImgFormat(const char *src, short toColor, short toDpi, const char *dst)
```

**Parameters**

- **src** - Full path name of the original image file. If this string is empty, the rotation is performed on the internal image.
- **toColor** - One of five values:
  - **LICENSE_INVALID** – Library was not initialized with the proper license.
• **IMAGE_SAME_COLOR** – No modification in the image color scheme.
• **IMAGE_BW** – Convert to black and white color scheme.
• **IMAGE_GRAY_256** – Convert to 256 gray scale color scheme.
• **IMAGE_COLOR_256** – Convert to 256-color scheme.
• **IMAGE_COLOR_TRUE** – Convert to true color scheme.

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

  [in] dst - Full path name of the destination file. If this value is an empty string, no save will be performed.

**Return Value**

If the function succeeds, the return value is **IMG_ERR_SUCCESS**.
If the function fails, it returns one of the following values:

• **IMG_ERR_BAD_COLOR** – Bad toColor parameter value.
• **IMG_ERR_BAD_DPI** – Bad toDpi parameter value.
• **IMG_ERR_FILE_OPEN** – Cannot open input file. This value is returned if the src string is not empty but it points to a missing or invalid source image file.
• **INVALID_INTERNAL_IMAGE** – This value is returned if the src string is empty but no document was scanned so there is no internal image in the memory.
• **IMG_ERR_FILE_SAVE_TO_FILE** – Cannot save destination file.
• **IMG_ERR_FILE_SAVE_TO_FILE** – Cannot save destination file due to invalid destination file or disk save error.

**Remarks**

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

Image reformat can be done either on the image color scheme or the image dpi, or both. Notice that changing the image format may lose the image color information (e.g., when converting from 24 bit true color to 256 gray scale). Modifying an image format from 256 gray scales to 24 bit true color will (obviously) not add color to the image but it will save the image with the proper RGB format (no color map) instead of using 256 gray scale palette.

After the image is reformatted it can be exported to external image file. The destination file name may be one of the seven file formats indicated above. If the destination file name has an unrecognizable extension, the function exports to the file in a BITMAP format (the default format). If no destination image file name is given (empty string), no save is done.

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

<table>
<thead>
<tr>
<th>Destination Image Extension</th>
<th>True color (BW2bit)</th>
<th>256 colors (8 bit)</th>
<th>Gray scale (8 bit)</th>
<th>Black and white (1 bit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMP</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>TIF</td>
<td>√</td>
<td></td>
<td>1</td>
<td>√</td>
</tr>
<tr>
<td>JPG</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCX</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TGA</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PNG</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSD</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1.1.139  ConcatImage

**Format**

```c
short ConcatImage(const char *src0, short src0Angle, const char *src1, short src1Angle, short combType, short destType, const char *dest)
```

**Parameters**

- `[in] src0` - Full path name of the first image.
- `[in] src0Angle` - The angle to rotate src0 before the combination.
- `[in] src1` - Full path name of the second image.
- `[in] src1Angle` - The angle to rotate src1 before the combination.
- `[in] combType` - The location of the images in the resulting image file:
  - `IMAGE_COMB_HORIZONTAL` - src0 is located to the left of src1.
  - `IMAGE_COMB_VERTICAL` - src0 is located above src1.
- `[in] destType` - The destination of the rotated image. This parameter may be one of two values:
  - `SAVE_TO_FILE` - Saves the image to a file. The file name should be given in dest parameter.
  - `SAVE_TO_CLIPBOARD` - Copy the rotated image to the clipboard.
- `[in] dest` - Full path name of the destination file. This parameter is ignored if the parameter combType is set to `SAVE_TO_CLIPBOARD`.

**Return Value**

If the function succeeds, the return value is `IMG_ERR_SUCCESS`.
If the function fails, it returns one of the following values:

- `LICENSE_INVALID` - Library was not initialized with the proper license.
- `IMG_ERR_BAD_ANGLE_0` - Bad rotation parameter for Image 0.
- `IMG_ERR_BAD_ANGLE_1` - Bad rotation parameter for Image 1.
- `IMG_ERR_FILE_OPEN_FIRST` - Cannot open src0 file.
- `IMG_ERR_FILE_OPEN_SECOND` - Cannot open src1 file.
- `IMG_ERR_BAD_DESTINATION` - Bad destination parameter (the destination is neither file nor clipboard).
- `IMG_ERR_COMB_TYPE` - Bad combType value.
- `IMG_ERR_FILE_SAVE_TO_CLIPBOARD` - Cannot save image to clipboard due to an error.
- `IMG_ERR_FILE_SAVE_TO_FILE` - Cannot save destination file due to a bad destination path or disk error.

**Remarks**

Use this function to combine two image files into a single image file. The function works in the following sequence:

1. Imports src0 to an image object 0.
2. Rotates image object0 by src0Angle.
3. Imports src1 to an image object 1.
4. Rotates image object1 by src1Angle.
5. Combines Image0 and Image1 one on top of each other (if combType is equal to `IMAGE_COMB_VERTICAL`) or one to the left of the other (if combType is equal to `IMAGE_COMB_HORIZONTAL`).
6. Saves the result image to an external file or to the clipboard.

**Notice:**
This function can work only on image files and not on the internal image.
The source files must not be of the same type (i.e., one may be a TIF type and the second can be a JPG type).

1.1.140 **ImageDespeckle**

**Format**

`short ImageDespeckle(const char *Image, short maxBlobArea)`

**Parameters**
- [in] `Image` - Image file name or an empty string if referring to the internal image.
- [in] `maxBlobArea` - Integer value that defines the blobs that will be deleted - all blobs with an area under this area will be deleted. Typically 30 will do the job.

**Return Value**
If the function succeeds, the return value is `IMG_ERR_SUCCESS`.
If the function fails, it returns one of the following values:

- `LICENSE_INVALID` – Library was not initialized with the proper license.
- `IMG_ERR_FILE_OPEN` - Cannot open input file. This value is returned if the Image string is not empty but it points to a missing or invalid source image file.
- `INVALID_INTERNAL_IMAGE` – This value is returned if the Image string is empty but no document was scanned so there is no internal image in the memory.
- `IMG_ERR_FILE_SAVE_TO_FILE` - Cannot save the file.

**Remarks**
Use this function to clear the "dirt" in black and white images.

1.1.141 **StampText**

**Format**

`StampText(const char* src, const char* str, int TextHeight, int location, COLORREF textColor, const char* dst)`

**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 texts (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 Value

- **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**: Prints the text on the upper portion of the image.
- **IMAGE_MIDDLE**: Prints the text on the middle portion of the image.
- **IMAGE_BOTTOM**: Prints the text on the bottom portion of the image.

1.1.142  **StampTextEx**

Format

```
StampTextEx(const char *Image, const char *txt, COLORREF color, int opacity, int verticalPos, int horizontalPos, const char *fontName, int fontSize, bool bold, bool bUnderline)
```

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 according to the location you specify. The vertical alignment of the text is set by the parameter verticalPos as follows:

- **IMAGE_TOP**: Prints the text on the upper portion of the image.
- **IMAGE_MIDDLE**: Prints the text on the middle portion of the image.
- **IMAGE_BOTTOM**: Prints the text on the bottom portion of the image.

Set by the parameter horizontalPos as follows:

- **IMAGE_LEFT**: Prints the text on the left portion of the image.
- **IMAGE_MID_HOR**: Prints the text on the middle portion of the image.
- **IMAGE_RIGHT**: Prints the text on the right portion of the image.
1.1.143  getImgBufferLengthEx

Format

GetImgBufferLengthEx(const char *FileType, int nImageType)

Parameters

FileType: The file extension in string format (JPG, BMP...).
nImageType: The image type (Front, Back, Face, Signature...).

Return Value

Integer value that represents the image buffer length.

Remarks

Use this function to get the size of the given image in order to define your buffer.
nImageType can be one of these values:
IMAGE_TYPE_FRONT 0
IMAGE_TYPE_BACK 1
IMAGE_TYPE_FACE 2
IMAGE_TYPE_SIGNATURE 3

1.1.144  AlwaysFlagAsCropped

Format

AlwaysFlagAsCropped(BOOL bCrop)

Parameters

bCrop: Controls the images cropping process. TRUE = images will not be cropped.

Return Value

Void.

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.

1.1.145  GetActExpiryDate

Format

GetActExpiryDate(LPCTSTR format, BSTR* pExpiry, BOOL* pNever)
Parameters

**format**: Controls the desired format of the returned expiry date value. Use any acceptable system string format or specify NULL or empty string to get the default local settings time/date format.

**pExpiry**: The Activation’s expiry date output.

**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

Void.

Remarks

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

Library DyamicFieldExtraction: General Functionality

DyamicFieldExtraction lets you export the scanned fields to any window you select.

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.

DyamicFieldExtraction Library Functions

1.1.146 OpenConfigDialog

Format

```c
short OpenConfigDialog(int nComponentType, const char sDefaultCfgFilename)
```

Parameters

- **nComponentType** - Indicates what is the component/components you are about to configure. i.e. - ID Card, Passport etc..
  Possible values are - DRIVER_LICENSE, BAR_CODE, PASSPORT, RAW_BARCODE, RAW_OCR, 
  BUISNESS_CARD, CHEQUE, MED.
  You can combine 1 or more values using the bitwise "or".
- **sDefaultCfgFilename** - A null terminated string specifying a filename to open when launching the dialog.

Return Value

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

Remarks

Use this function to setup the windows you would like the fields to be exported to. When done, save the configuration file for later usage.
1.1.147 ExtractDynamicFields

Format

```c
short ExtractDynamicFields (const char sCfgFilename, bool bUseWideString ,int nComponentType = 0,int nFieldIndex=-1)
```

Parameters

- **sCfgFilename**: A null terminated string specifying the configuration file to use in order to extract the scanned values.
- **bUseWideString**: If 0, then the fields are exported as Unicode strings, otherwise the fields will be exported as wide strings.
- **nComponentType**: Specifies what will be the extracted component. Possible choices are: DRIVER_LICENSE, BAR_CODE, PASSPORT, RAW_BARCODE, RAW_OCR, BUSINESS_CARD, CHEQUE, MED. Values can be combined with the | operator. If this parameter is 0, all possible sections will be extracted.
- **nFieldIndex**: Specifies the field to be extracted. If the value of the parameter is -1, then all fields in the specified section will be extracted.

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

1.1.148 ExtractTextFromFile

Format

```c
short ExtractTextFromFile (const char *sCfgFilename,const char *fName, unsigned char type , long *pAutoRotation = NULL, const char *CustStr=NULL , bool bUseWideString )
```

Parameters

- **sCfgFilename**: A null terminated string specifying the configuration file to use in order to extract the possible values.
- **fName**: Full path name of the original image.
- **type**: - Instructs the OCR what type of data is written in the image. This value increases the detection accuracy and speeds 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.
- **CustStr**: - Char set
- **bUseWideString**: If 0, then the fields are exported as Unicode strings, otherwise the fields will be exported as wide strings.

Return Value

Start from the start
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 to extract text bulks from an image. The text size is limited to 4K (4096) characters. When the scan is done, this function will take the values in memory and will post it to the configured windows.

### 1.1.149 IsDynamicExportNeeded

**Format**

```
DYAMICFIELDEXTRACION_API int IsDynamicExportNeeded(const char *ConfigFileName)
```

**Parameters**

- **[in] ConfigFileName** A null terminated string specifying the configuration file to use in order to extract the possible values.

**Return Value**

The 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, BUSINESSE_CARD, CHEQUE, MED. The values can be read using the ‘&’ operator.

### 1.1.150 OpenMacroDialog

**Format**

```
short OpenMacroDialog (const char *sDefaultMacroFilename=NULL )
```

**Parameters**

- **[In] sDefaultMacroFilename** A null terminated 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.
1.1.151 PlayMacro

Format

short PlayMacro (const char * sMacroFilename)

Parameters

[In] sMacroFilename A null terminated string specifying the macro file to be played.

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.

Library IMRTD.dll: General Functionality

The IMRTD library lets you read an electronic passport or an electronic ID according to ICAO3203 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 data using ReadPassport or ParsePassportRF and then you can access each field using GetField or GetFieldData for binary data. Extended Access control and Active Authentication are not implemented yet.

IMRTD Library Functions

1.1.152 ReadPassport

Format

int ReadPassport(const char * pMRZFirstLine, const char * pMRZSecondLine, const char * pMRZThirdLine, const char * pDumpFolder);
Parameters

[in] pMRZFirstLine - A null terminated string containing the first line of the MRZ as it appears on the passport.
[in] pMRZSecondLine - A null terminated string containing the second line of the MRZ as it appears on the passport.
[in] pMRZThirdLine - A null terminated 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] pDumpFolder - A null terminated 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 that the MRZ info will be entered in order to get access to the passport. You can use the OCR passport library to scan and read the MRZ info automatically.

1.1.153 ParsePassportRF

Format

```c
int ParsePassportRf(unsigned char* pBuf, int nSize)
```

Parameters

[in] pBuf - A null terminated 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 already read from the passport and will put each field in its corresponding place.

1.1.154 ParsePassportRFFile
### ParsePassportRfFile

**Parameters**

- **pBuf** - A null terminated string pointing to the file to be parsed.
- **bReset** - A Boolean - if set to true, 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.

#### 1.1.155 GetField

**Format**

```c
void GetField(int nIndex, char *szBuf);
```

**Parameters**

- **nIndex** - An integer specifying the index of the field to be retrieved.
- **szBuf** - A buffer to contain the field data.

**Return Value**

Void.

**Remarks**

The function will return a string containing the value of the field.

Use this function after calling ReadPassport or ReadPassportRF or ReadPassportRFFile.

#### 1.1.156 GetField

```c
void GetField(std::string sName, char *szBuf);
```
Parameters

[in] std::string A string containing the name of the field required.
[out] szBuf A buffer to contain the field data.

It can be one of the following:

- COUNTRY CODE
- FIRST NAME
- MIDDLE NAME
- LAST NAME
- PASSPORT_NUMBER
- PASSPORT_CHECK_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
- CONTENT_SPECIFIC_CONSTRUVD_DATA
- NUMBER_OF_OTHER_NAMES
- ISSUING_AuthorITY
- ISSUE_DATE YYYYMMDD
- NAME_OF_OTHER_PERSON
- ENDORSEMENT
- TAX_EXIT_REQUIREMENTS
- IMAGE_OF_FRONT_DOCUMENT
- IMAGE_OF_REAR_DOCUMENT
- DATE_TIME_OF_DOC_PERSONALIZATION
- SERIAL_NUM_OF_PERSONALIZATION_SYSTEM

Return Value

Void.

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

1.1.157  GetFieldName

Format

```c
void GetFieldName(int nIndex, char *szBuf);
```

Parameters

- [in] `nIndex` An integer specifying the index of the field to be retrieved.
- [out] `szBuf` A buffer to contain the field name.

Return Value

[out] Void.

Remarks

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

1.1.158  GetFieldData

Format

```c
char* GetFieldData(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] `char*`

Remarks

The function will return a buffer to the data of the field.

1.1.159  GetFieldDataLen

Format

```c
int GetFieldDataLen(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

Smart from the start
The size of the data in bytes.

**Remarks**
The function will return the size of the data in bytes. Use it together with GetFieldData.

### 1.1.160 GetFieldDataDesc

**Format**

```c
void GetFieldDesc(int nIndex, char *szBuf);
```

**Parameters**

- **[in]** `nIndex` An integer specifying the index of the field to be retrieved.
- Right now only FACE_IMAGE (16) is supported.
- **[out]** `szBuf` A buffer to contain the field description.

**Return Value**
Void.

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

### 1.1.161 GetNumFields

**Format**

```c
int GetNumFields();
```

**Parameters**

**Return Value**
[out] int – Number of available fields.

**Remarks**
This one returns the number of fields available, empty or not.

### 1.1.162 PassiveAuthentication

**Format**

```c
bool PassiveAuthentication(const char* sCerFile, const char* pDumpFolder);
```

**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 it will return false.

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

1.1.163 IsCountryCertificateApproved

Format

```c
bool IsCountryCertificateApproved();
```

Parameters

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

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

1.1.164 IsChipCertificateApproved

Format

```c
bool IsChipCertificateApproved();
```

Parameters

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 passive authentication.

1.1.165 IsPassiveAuthenticationFlowSucceeded

Format

```c
bool IsPassiveAuthenticationFlowSucceeded();
```

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.

1.1.166 GetPassiveAuthenticationLastError.

Format

```c
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.

Appendix A: Definitions

1.1.167 Field definition for Slib library

```c
#define SLIB_TRUE 1
#define SLIB_FALSE 0
#define SLIB_ERR_NONE 1
#define SLIB_ERR_INVALID_SCANNER -1

// scanning failure definition
#define SLIB_ERR_SCANABORT -7
#define SLIB_ERR_NO_PAPER -8
#define SLIB_ERR_PAPER_JAM -9
#define SLIB_ERR_FILE_IO_ERROR -10
#define SLIB_ERR_PRINTER_PORT_USED -11
#define SLIB_ERR_OUT_OF_MEMORY -12
```
#define SLIB_ERR_IMAGE_CONVERSION -16
#define SLIB_ERR_BAD_WIDTH_PARAM -2
#define SLIB_ERR_BAD_HEIGHT_PARAM -3
#define SLIB_ERR_BAD_PARAM -2
#define SLIB_LIBRARY_ALREADY_INITIALIZED -13
#define SLIB_ERR_DRIVER_NOT_FOUND -14
#define SLIB_ERR_SCANNER_BUSSY -15
#define SLIB_UNLOAD_FAILED_BAD_PARENT -17
#define SLIB_NOT_INITILIZED -18
#define SLIB_LIBRARY_ALREADY_USED_BY_OTHER_APP -19
#define SLIB_CONFLICT_WITH_INOUTSCAN_PARAM -20
#define SLIB_CONFLICT_WITH_SCAN_SIZE_PARAM -21
#define SLIB_NO_SUPPORT_MULTIPLE_DEVICES -22
#define SLIB_ERR_CAM_ALREADY_ASSIGNED -23
#define SLIB_ERR_NO_FREE_CAM_FOUND -24
#define SLIB_ERR_CAM_NOT_FOUND -25
#define SLIB_ERR_CAM_NOT_ASSIGNED_TO_THIS_APP -26
#define SLIB_ERR_IP_SCAN_VERSION_TOO_OLD -27
#define SLIB_ERR_ASYNC_SCANS_IN_QUEUE -28
#define GENERAL_ERR_PLUG_NOT_FOUND -200
#define ERR_SCANNER_ALREADY_IN_USE -201
#define SLIB_ERR_SCANNER_ALREADY_IN_USE -202
#define SLIB_ERR_CANNOT_OPEN_TWAIN_SOURCE -203
#define SLIB_ERR_NO_TWAIN_INSTALLED -204
#define SLIB_ERR_NO_NEXT_VALUE -205

1.1.168 Field definition for IdCard library

#define FIELD_NAME 0
#define FIELD_NAME_TYPE 1
#define FIELD_ADDRESS 2
#define FIELD_CITY 3
#define FIELD_STATE 4
#define FIELD_ZIP 5
#define FIELD_DOB 6
#define FIELD_EXPIRES 7
#define FIELD_ISSUE 8
#define FIELD_LICENSE_MAIN 9
#define FIELD_LICENSE_SEC 10
#define FIELD_CLASS 11
#define FIELD_ID_NUMBER 12
#define FIELD_EYES 13
#define FIELD_HAIR 14
#define FIELD_HEIGHT 15
#define FIELD_SEX 16
#define FIELD_WEIGHT 17
#define FIELD_DUPLICATE 18
#define FIELD_COUNTY 19
#define FIELD_FEE 21
#define FIELD_RESTRICTION 22
#define FIELD_TYPE 23
#define FIELD_END 24
#define FIELD_SIG_NUM 25
#define FIELD_ORIGINAL 26
#define FIELD_SSNUMBER 27
#define FIELD_AUDIT 28
#define FIELD_NAME_F 29
#define FIELD_NAME_M 30
#define FIELD_NAME_L 31
#define FIELD_NAME_S 32
#define FIELD_ADDRESS2 33
#define FIELD_ADDRESS3 34
#define FIELD_ADDRESS4 35
#define FIELD_TEXT1 36
#define FIELD_TEXT2 37
#define FIELD_TEXT3 38
#define FIELD_ADDRESS5 39
#define FIELD_COUNTRY 41
#define FIELD_ADDRESS6 42
#define FIELD_DOC_TYPE 43
#define FIELD_COUNTRY_SHORT 44
#define FIELD_ISSUE_LOCAL 55
#define FIELD_ID_NATIONALITY 56
#define FIELD_PLACE_OF_BIRTH 57
#define FIELD_PLACE_OF_ISSUE 58
#define FIELD_MOTHER_NAME 59
#define FIELD_FATHER_NAME 60
#define FIELD_RAW_DATA 61
#define FIELD_CARD_TYPE 62
1.1.169 Field definition for Barcode library

#define BCF_ADDRESS2 54
#define BCF_CITY2 55
#define BCF_STATE2 56
#define BCF_ZIP2 57
#define BCF_EMUL_FULL_NAME 100
#define BCF_EMUL_FIRST_NAME 101
#define BCF_EMUL_MIDDLE_NAME 102
#define BCF_EMUL_LAST_NAME 103
#define BCF_EMUL_NAME_SUFFI 104
#define BCF_EMUL DOB 105
#define BCF_EMUL_ISSUE 106
#define BCF_EMUL_EXP 107
#define BCF_EMUL_ADDRESS 108
#define BCF_EMUL_CITY 109
#define BCF_EMUL_STATE 110
#define BCF_EMUL ZIP 111
#define BCF_EMUL LICENSE 112
#define BCF_EMUL SSN 113
#define BCF_EMUL END 114
#define BCF_EMUL EYES 115
#define BCF_EMUL HAIR 116
#define BCF_EMUL HEIGHT 117
#define BCF_EMUL WEIGHT 118
#define BCF_EMUL_RANK 119
#define BCF_EMUL GENEVA_CODE 120
#define BCF_EMUL_SECURITY_CODE 121
#define BCF_EMUL CHAMPUS_EFFECTIVE_DATE 122
#define BCF_EMUL CHAMPUS_EXPIRATION_DATE 123
#define BCF_EMUL SPONSER_PERSON_PID 124
#define BCF_EMUL SPONSER FULL_NAME 125
#define BCF_EMUL SPONSER FIRST_NAME 126
#define BCF_EMUL SPONSER_MIDDL 127
#define BCF_EMUL SPONSER_LAST_NAME 128
#define BCF_EMUL SPONSER_NAME SUFFIX 129

1.1.170 Field definition for CPassport library

#define FIELD_STATE 4
#define FIELD_DOB 6
#define FIELD_EXPIRES 7
#define FIELD_ISSUE 8
#define FIELD_ID_NUMBER 12
#define FIELD SEX 16
#define FIELD_NAME F 29
#define FIELD_NAME M 30
#define FIELD_NAME L 31
#define FIELD_ADDRESS2 33
1.1.171 Library CPassport constants

\[\text{Public Const PASS\_ERR\_NONE} = 1\]
\[\text{Public Const PASS\_ERR\_BAD\_PARAM} = -40\]
\[\text{Public Const PASS\_ERR\_CANNOT\_DELETE\_DESTINATION\_IMAGE} = -41\]
\[\text{Public Const PASS\_ERR\_CANNOT\_COPY\_TO\_DESTINATION} = -42\]
\[\text{Public Const PASS\_ERR\_FACE\_IMAGE\_NOT\_FOUND} = -43\]

1.1.172 Field definition for MagLib library

\[\text{#define FIELD\_ADDRESS} = 2\]
\[\text{#define FIELD\_CITY} = 3\]
\[\text{#define FIELD\_STATE} = 4\]
\[\text{#define FIELD\_ZIP} = 5\]
\[\text{#define FIELD\_DOB} = 6\]
\[\text{#define FIELD\_EXPIRES} = 7\]
\[\text{#define FIELD\_ISSUE} = 8\]
\[\text{#define FIELD\_LICENSE\_MAIN} = 9\]
\[\text{#define FIELD\_CLASS} = 11\]
\[\text{#define FIELD\_EYES} = 13\]
\[\text{#define FIELD\_HAIR} = 14\]
\[\text{#define FIELD\_HEIGHT} = 15\]
\[\text{#define FIELD\_SEX} = 16\]
\[\text{#define FIELD\_WEIGHT} = 17\]
\[\text{#define FIELD\_RESTRICTION} = 22\]
\[\text{#define FIELD\_END} = 24\]
\[\text{#define FIELD\_NAME\_F} = 29\]
\[\text{#define FIELD\_NAME\_M} = 30\]
\[\text{#define FIELD\_NAME\_L} = 31\]
\[\text{#define FIELD\_NAME\_S} = 32\]
1.1.173  **Library CImageCtrl constants**

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

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

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

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

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

' image color conversion
Public Const IMAGE_SAME_COLOR = 0
Public Const IMAGE_BW = 1
Public Const IMAGE_GRAY_256 = 2
Public Const IMAGE_COLOR_256 = 3
Public Const IMAGE_COLOR_TRUE = 4
Public Const IMAGE_SAVE_ENHANCED_IMAGE = 5

1.1.174  **Library SOCRdll constants**

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

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

1.1.175 Library MRTD constants

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

#define RFID_FIELD_FIRST 0
#define COUNTRY_CODE 0
#define FIRST_NAME 1
#define MIDDLE_NAME 2
#define LAST_NAME 3
#define PASSPORT_NUMBER 4
#define PASSPORT_CHECK_DIGIT 5
#define NATIONALITY 6
#define DOB 7
#define DOB_CHECK_DIGIT 8
#define SEX 9
#define EXPIRES 10
#define EXPIRES_CHECK_DIGIT 11
#define PERSONAL_NUMBER 12
#define PERSONAL_NUMBER_CHECK_DIGIT 13
#define COMPOSITE_CHECK_DIGIT 14
#define MRZ 15
// Dg 2
#define FACE_IMAGE 16
// Dg 11
#define POB 17
#define FULL_NAME 18
#define OTHER_NAME 19
#define PERSONAL_NUMBER_DG11 20
#define FULLDOB 21
#define PERMANENT_ADDRESS 22
#define TELEPHONE_NUMBER 23
#define PROFESSION 24
#define TITLE 25
#define PERSONAL_SUMMARY 26
#define PROOF_OF_CITIZENSHIP 27
#define OTHER_VALID_TD_NUMBERS 28
#define CUSTODY_INFORMATION 29
#define CONTENT_SPECIFIC_CONSTRUCTED_DATA 30
#define NUMBER_OF_OTHER_NAMES 31
// DG 12
#define ISSUING_AUTHORITY 32
#define ISSUE_DATE 33
#define NAME_OF_OTHER_PERSON 34
#define ENDORSEMENT 35
#define TAX_EXIT_REQUIREMENTS 36
#define IMAGE_OF_FRONT_DOCUMENT 37
#define IMAGE_OF_REAR_DOCUMENT 38
Appendix B – Supported States for Detection

The following table shows the supported states by IdCard library. This list will be updated in every new version release of IdCard library.

<table>
<thead>
<tr>
<th>Region Name</th>
<th>Region ID</th>
<th>Country Name</th>
<th>Country ID</th>
<th>Document\State Name</th>
<th>Document\State ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA</td>
<td>0</td>
<td>USA</td>
<td>0</td>
<td>ALABAMA</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ALASKA</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ARIZONA</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ARKANSAS</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CALIFORNIA</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>COLORADO</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CONNECTICUT</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DELAWARE</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FLORIDA</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>GEORGIA</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>HAWAII</td>
<td>54</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>IDAHO</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ILLINOIS</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>INDIANA</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>IOWA</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>KANSAS</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>KENTUCKY</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>LOUISIANA</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MAINE</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MARYLAND</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MASSACHUSETTS</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MICHIGAN</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MINNESOTA</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MISSISSIPPI</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MISSOURI</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MONTANA</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>NEBRASKA</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>NEVADA</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>NEW HAMPSHIRE</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>NEW JERSEY</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>NEW MEXICO</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>NEW YORK</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>NORTH CAROLINA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>NORTH DAKOTA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>OHIO</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>OKLAHOMA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>OREGON</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>PENNSYLVANIA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>RHODE ISLAND</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>SOUTH CAROLINA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>SOUTH DAKOTA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>TENNESSEE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>TEXAS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>91</td>
<td>US VIRGINISLANDS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>UTAH</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>VERMONT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>VIRGINIA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>WASHINGTON</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>WASHINGTON DC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>WEST VIRGINIA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>WISCONSIN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>WYOMING</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>81</td>
<td>GREEN CARD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>82</td>
<td>ARMY CARD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>83</td>
<td>SSN CARD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>84</td>
<td>NYPD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>85</td>
<td>GUAM</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>86</td>
<td>MEXICO USA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>88</td>
<td>TRIBAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>89</td>
<td>FIPS ID</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>92</td>
<td>PASSPORT CARD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CANADA</td>
<td>CANADA</td>
</tr>
<tr>
<td>71</td>
<td>ALBERTA</td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>BRITISH COLUMBIA</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>ONTARIO</td>
<td></td>
</tr>
<tr>
<td>880</td>
<td>PRINCE EDWARD</td>
<td></td>
</tr>
<tr>
<td>73</td>
<td>MANITOBA</td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>NEW</td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>BRUNSWICK</td>
<td></td>
</tr>
<tr>
<td>76</td>
<td>NEW FOUNDLAND</td>
<td></td>
</tr>
<tr>
<td>76</td>
<td>NW TERITORIES</td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>NOVASCOTIA</td>
<td></td>
</tr>
<tr>
<td>1079</td>
<td>QUEBEC</td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>SASKATCHEWAN</td>
<td></td>
</tr>
<tr>
<td>79</td>
<td>CANADA CITIZEN ID</td>
<td></td>
</tr>
<tr>
<td>Country</td>
<td>Code</td>
<td>Country</td>
</tr>
<tr>
<td>----------------------</td>
<td>------</td>
<td>----------------------</td>
</tr>
<tr>
<td>Antigua</td>
<td>108</td>
<td>Antigua</td>
</tr>
<tr>
<td>Argentina</td>
<td>93</td>
<td>Argentina</td>
</tr>
<tr>
<td>Aruba</td>
<td>118</td>
<td>Aruba</td>
</tr>
<tr>
<td>Bahamas</td>
<td>21</td>
<td>Bahamas</td>
</tr>
<tr>
<td>Barbados</td>
<td>133</td>
<td>Barbados</td>
</tr>
<tr>
<td>Belize</td>
<td>106</td>
<td>Belize</td>
</tr>
<tr>
<td>Bermuda</td>
<td>13</td>
<td>Bermuda</td>
</tr>
<tr>
<td>Bolivia</td>
<td>60</td>
<td>Bolivia</td>
</tr>
<tr>
<td>Brazil</td>
<td>8</td>
<td>Brazil</td>
</tr>
<tr>
<td>Cayman Islands</td>
<td>134</td>
<td>Cayman Islands</td>
</tr>
<tr>
<td>Chile</td>
<td>4</td>
<td>Chile</td>
</tr>
<tr>
<td>Colombia</td>
<td>79</td>
<td>Colombia</td>
</tr>
<tr>
<td>Costa Rica</td>
<td>28</td>
<td>Costa Rica</td>
</tr>
<tr>
<td>Cuba</td>
<td>135</td>
<td>Cuba</td>
</tr>
<tr>
<td>Curacao</td>
<td>112</td>
<td>Curacao</td>
</tr>
<tr>
<td>Dominican Republic</td>
<td>73</td>
<td>Dominican Republic</td>
</tr>
<tr>
<td>Ecuador</td>
<td>67</td>
<td>Ecuador</td>
</tr>
<tr>
<td>El Salvador</td>
<td>34</td>
<td>El Salvador</td>
</tr>
<tr>
<td>French Guiana</td>
<td>136</td>
<td>French Guiana</td>
</tr>
<tr>
<td>Greenland</td>
<td>137</td>
<td>Greenland</td>
</tr>
<tr>
<td>Grenada</td>
<td>138</td>
<td>Grenada</td>
</tr>
<tr>
<td>Guyana</td>
<td>139</td>
<td>Guyana</td>
</tr>
<tr>
<td>Guatemala</td>
<td>33</td>
<td>Guatemala</td>
</tr>
<tr>
<td>Haiti</td>
<td>74</td>
<td>Haiti</td>
</tr>
<tr>
<td>Honduras</td>
<td>69</td>
<td>Honduras</td>
</tr>
<tr>
<td>Jamaica</td>
<td>140</td>
<td>Jamaica</td>
</tr>
<tr>
<td>Paraguay</td>
<td>150</td>
<td>Paraguay</td>
</tr>
<tr>
<td>Mexico</td>
<td>6</td>
<td>Mexico</td>
</tr>
<tr>
<td>Nicaragua</td>
<td>32</td>
<td>Nicaragua</td>
</tr>
<tr>
<td>Panama</td>
<td>36</td>
<td>Panama</td>
</tr>
<tr>
<td>Peru</td>
<td>29</td>
<td>Peru</td>
</tr>
<tr>
<td>Puerto Rico</td>
<td>30</td>
<td>Puerto Rico</td>
</tr>
<tr>
<td>Saint Christopher Nevis</td>
<td>94</td>
<td>Saint Christopher Nevis</td>
</tr>
<tr>
<td>Saint Kitts Nevis</td>
<td>160</td>
<td>Saint Kitts Nevis</td>
</tr>
<tr>
<td>Saint Lucia</td>
<td>170</td>
<td>Saint Lucia</td>
</tr>
<tr>
<td>Saint Vincent Grenadines</td>
<td>180</td>
<td>Saint Vincent Grenadines</td>
</tr>
<tr>
<td>Suriname</td>
<td>190</td>
<td>Suriname</td>
</tr>
<tr>
<td>Trinidad</td>
<td>120</td>
<td>Trinidad</td>
</tr>
<tr>
<td>Turks Caicos</td>
<td>103</td>
<td>Turks Caicos</td>
</tr>
<tr>
<td>Uruguay</td>
<td>200</td>
<td>Uruguay</td>
</tr>
<tr>
<td>Venezuela</td>
<td>80</td>
<td>Venezuela</td>
</tr>
<tr>
<td>Virgin Islands</td>
<td>115</td>
<td>Virgin Islands</td>
</tr>
</tbody>
</table>

**Europe** 3
<table>
<thead>
<tr>
<th>Country</th>
<th>Code</th>
<th>Country</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALBANIA</td>
<td>95</td>
<td>ALBANIA</td>
<td>1010</td>
</tr>
<tr>
<td>ANDORRA</td>
<td>88</td>
<td>ANDORRA</td>
<td>940</td>
</tr>
<tr>
<td>ARMENIA</td>
<td>128</td>
<td>ARMENIA</td>
<td>1390</td>
</tr>
<tr>
<td>AUSTRIA</td>
<td>57</td>
<td>AUSTRIA</td>
<td>640</td>
</tr>
<tr>
<td>AZERBAIJAN</td>
<td>113</td>
<td>AZERBAIJAN</td>
<td>1240</td>
</tr>
<tr>
<td>BELARUS</td>
<td>125</td>
<td>BELARUS</td>
<td>1360</td>
</tr>
<tr>
<td>BELGIUM</td>
<td>38</td>
<td>BELGIUM</td>
<td>420</td>
</tr>
<tr>
<td>BOSNIA</td>
<td>48</td>
<td>BOSNIA</td>
<td>530</td>
</tr>
<tr>
<td>BULGARIA</td>
<td>45</td>
<td>BULGARIA</td>
<td>490</td>
</tr>
<tr>
<td>CROATIA</td>
<td>41</td>
<td>CROATIA</td>
<td>450</td>
</tr>
<tr>
<td>CYPRUS</td>
<td>76</td>
<td>CYPRUS</td>
<td>800</td>
</tr>
<tr>
<td>CZECH</td>
<td>47</td>
<td>CZECH</td>
<td>520</td>
</tr>
<tr>
<td>DENMARK</td>
<td>72</td>
<td>DENMARK</td>
<td>760</td>
</tr>
<tr>
<td>ESTONIA</td>
<td>71</td>
<td>ESTONIA</td>
<td>750</td>
</tr>
<tr>
<td>FINLAND</td>
<td>64</td>
<td>FINLAND</td>
<td>690</td>
</tr>
<tr>
<td>FRANCE</td>
<td>5</td>
<td>FRANCE</td>
<td>90</td>
</tr>
<tr>
<td>EUROPE_GENERAL_CARDS</td>
<td>46</td>
<td>EUROPE_GENERAL_CARDS</td>
<td>510</td>
</tr>
<tr>
<td>GERMANY</td>
<td>10</td>
<td>GERMAN_ID</td>
<td>140</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GERMAN_LIC</td>
<td>141</td>
</tr>
<tr>
<td>EUR_GEORGIA</td>
<td>126</td>
<td>EUR_GEORGIA</td>
<td>1370</td>
</tr>
<tr>
<td>GREECE</td>
<td>121</td>
<td>GREECE</td>
<td>1320</td>
</tr>
<tr>
<td>GUERNSEY</td>
<td>89</td>
<td>GUERNSEY</td>
<td>950</td>
</tr>
<tr>
<td>HOLAND</td>
<td>17</td>
<td>HOLAND</td>
<td>210</td>
</tr>
<tr>
<td>HUNGARY</td>
<td>49</td>
<td>HUNGARY</td>
<td>540</td>
</tr>
<tr>
<td>ISLAND</td>
<td>78</td>
<td>ISLAND</td>
<td>820</td>
</tr>
<tr>
<td>IRELAND</td>
<td>54</td>
<td>IRELAND</td>
<td>600</td>
</tr>
<tr>
<td>ISLE_OF_MAN</td>
<td>90</td>
<td>ISLE_OF_MAN</td>
<td>960</td>
</tr>
<tr>
<td>ISRAEL</td>
<td>9</td>
<td>ISRAEL</td>
<td>120</td>
</tr>
<tr>
<td>ITALY</td>
<td>23</td>
<td>ITALY</td>
<td>270</td>
</tr>
<tr>
<td>KOSOVO</td>
<td>51</td>
<td>KOSOVO</td>
<td>560</td>
</tr>
<tr>
<td>LATVIA</td>
<td>91</td>
<td>LATVIA</td>
<td>970</td>
</tr>
<tr>
<td>LIECHTENSTEIN</td>
<td>63</td>
<td>LIECHTENSTEIN</td>
<td>680</td>
</tr>
<tr>
<td>LITHUANIA</td>
<td>19</td>
<td>LITHUANIA</td>
<td>230</td>
</tr>
<tr>
<td>LUX</td>
<td>18</td>
<td>LUX</td>
<td>220</td>
</tr>
<tr>
<td>MACEDONIA</td>
<td>100</td>
<td>MACEDONIA</td>
<td>1060</td>
</tr>
<tr>
<td>MALTA</td>
<td>92</td>
<td>MALTA</td>
<td>980</td>
</tr>
<tr>
<td>MOLDOVA</td>
<td>105</td>
<td>MOLDOVA</td>
<td>1110</td>
</tr>
<tr>
<td>MONACO</td>
<td>129</td>
<td>MONACO</td>
<td>1400</td>
</tr>
<tr>
<td>MONTENEGRO</td>
<td>97</td>
<td>MONTENEGRO</td>
<td>1030</td>
</tr>
<tr>
<td>NORWAY</td>
<td>15</td>
<td>NORWAY</td>
<td>190</td>
</tr>
<tr>
<td>POLAND</td>
<td>27</td>
<td>POLAND</td>
<td>310</td>
</tr>
<tr>
<td>PORTUGAL</td>
<td>31</td>
<td>PORTUGAL</td>
<td>350</td>
</tr>
<tr>
<td>ROMANIA</td>
<td>12</td>
<td>ROMANIA</td>
<td>160</td>
</tr>
<tr>
<td>RUSSIA</td>
<td>58</td>
<td>RUSSIA</td>
<td>650</td>
</tr>
<tr>
<td>SAN_MARINO</td>
<td>130</td>
<td>SAN_MARINO</td>
<td>1410</td>
</tr>
<tr>
<td>SERBIA</td>
<td>59</td>
<td>SERBIA</td>
<td>660</td>
</tr>
<tr>
<td>Country</td>
<td>Code</td>
<td>Country</td>
<td>Code</td>
</tr>
<tr>
<td>------------------</td>
<td>------</td>
<td>------------------</td>
<td>------</td>
</tr>
<tr>
<td>SLOVAKIA</td>
<td>50</td>
<td>SLOVAKIA</td>
<td>550</td>
</tr>
<tr>
<td>SLOVENIA</td>
<td>53</td>
<td>SLOVENIA</td>
<td>580</td>
</tr>
<tr>
<td>SPAIN</td>
<td>11</td>
<td>SPAIN</td>
<td>150</td>
</tr>
<tr>
<td>SWEDEN</td>
<td>22</td>
<td>SWEDEN</td>
<td>260</td>
</tr>
<tr>
<td>SWISS</td>
<td>20</td>
<td>SWISS</td>
<td>240</td>
</tr>
<tr>
<td>TURKEY</td>
<td>25</td>
<td>TURKEY</td>
<td>290</td>
</tr>
<tr>
<td>UKRAINE</td>
<td>131</td>
<td>UKRAINE</td>
<td>1420</td>
</tr>
<tr>
<td>UNITED KINGDOM</td>
<td>7</td>
<td>UNITED KINGDOM AND IRELAND</td>
<td>110</td>
</tr>
<tr>
<td>UZBEKISTAN</td>
<td>127</td>
<td>UZBEKISTAN</td>
<td>1380</td>
</tr>
<tr>
<td>VATICAN</td>
<td>132</td>
<td>VATICAN</td>
<td>1430</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Region</th>
<th>Numeric Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUSTRALIA</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>NSW</td>
<td>50</td>
</tr>
<tr>
<td>ACT</td>
<td>51</td>
</tr>
<tr>
<td>QLD</td>
<td>52</td>
</tr>
<tr>
<td>TAS</td>
<td>55</td>
</tr>
<tr>
<td>VIC</td>
<td>53</td>
</tr>
<tr>
<td>WST</td>
<td>56</td>
</tr>
<tr>
<td>SA</td>
<td>57</td>
</tr>
<tr>
<td>NT</td>
<td>58</td>
</tr>
<tr>
<td>COOK ISLANDS</td>
<td>59</td>
</tr>
<tr>
<td>FIJI</td>
<td>61</td>
</tr>
<tr>
<td>KEYPASS</td>
<td>500</td>
</tr>
<tr>
<td>NEWZEALAND (AU)</td>
<td>2290</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Region</th>
<th>Numeric Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASIA</td>
<td>5</td>
</tr>
<tr>
<td>AFGHANISTAN</td>
<td>970</td>
</tr>
<tr>
<td>BAHRAIN</td>
<td>56</td>
</tr>
<tr>
<td>BANGLADESH</td>
<td>680</td>
</tr>
<tr>
<td>BHUTAN</td>
<td>690</td>
</tr>
<tr>
<td>BRUNEI</td>
<td>68</td>
</tr>
<tr>
<td>CAMBODIA</td>
<td>700</td>
</tr>
<tr>
<td>CHINA</td>
<td>43</td>
</tr>
<tr>
<td>EAST_TIMOR</td>
<td>710</td>
</tr>
<tr>
<td>ISRAEL_DOCS</td>
<td>107</td>
</tr>
<tr>
<td>INDIA</td>
<td>81</td>
</tr>
<tr>
<td>INDONESIA</td>
<td>37</td>
</tr>
<tr>
<td>IRAN</td>
<td>720</td>
</tr>
<tr>
<td>IRAQ</td>
<td>111</td>
</tr>
<tr>
<td>JAPAN</td>
<td>730</td>
</tr>
<tr>
<td>JORDAN</td>
<td>740</td>
</tr>
<tr>
<td>KAZAKHSTAN</td>
<td>750</td>
</tr>
<tr>
<td>KUWAIT</td>
<td>116</td>
</tr>
<tr>
<td>KYRGYZSTAN</td>
<td>760</td>
</tr>
<tr>
<td>LAOS</td>
<td>770</td>
</tr>
<tr>
<td>Country</td>
<td>Code</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------</td>
</tr>
<tr>
<td>LEBANON</td>
<td>780</td>
</tr>
<tr>
<td>MALAYSIA</td>
<td>2</td>
</tr>
<tr>
<td>MALDIVES</td>
<td>790</td>
</tr>
<tr>
<td>MONGOLIA</td>
<td>800</td>
</tr>
<tr>
<td>MYANMAR</td>
<td>810</td>
</tr>
<tr>
<td>NEPAL</td>
<td>820</td>
</tr>
<tr>
<td>New ZEALAND</td>
<td>16</td>
</tr>
<tr>
<td>NORTH_KOREA</td>
<td>830</td>
</tr>
<tr>
<td>OMAN</td>
<td>85</td>
</tr>
<tr>
<td>PAKISTAN</td>
<td>840</td>
</tr>
<tr>
<td>PHILIPPINES</td>
<td>102</td>
</tr>
<tr>
<td>QATAR</td>
<td>86</td>
</tr>
<tr>
<td>SAUDI_ARABIA</td>
<td>87</td>
</tr>
<tr>
<td>SINGAPORE</td>
<td>14</td>
</tr>
<tr>
<td>SOUTH_KOREA</td>
<td>850</td>
</tr>
<tr>
<td>SRI_LANKA</td>
<td>860</td>
</tr>
<tr>
<td>SYRIA</td>
<td>870</td>
</tr>
<tr>
<td>TAJIKISTAN</td>
<td>880</td>
</tr>
<tr>
<td>THAILAND</td>
<td>104</td>
</tr>
<tr>
<td>TURKMENISTAN</td>
<td>890</td>
</tr>
<tr>
<td>UAE</td>
<td>55</td>
</tr>
<tr>
<td>VIETNAM</td>
<td>900</td>
</tr>
<tr>
<td>YEMEN</td>
<td>910</td>
</tr>
</tbody>
</table>

**AFRICA** 6

<table>
<thead>
<tr>
<th>Country</th>
<th>Code</th>
<th>Country</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALGERIA</td>
<td>210</td>
<td>ALGERIA</td>
<td>1580</td>
</tr>
<tr>
<td>ANGOLA</td>
<td>220</td>
<td>ANGOLA</td>
<td>1590</td>
</tr>
<tr>
<td>BENIN</td>
<td>230</td>
<td>BENIN</td>
<td>1600</td>
</tr>
<tr>
<td>BOTSWANA</td>
<td>240</td>
<td>BOTSWANA</td>
<td>1610</td>
</tr>
<tr>
<td>BURKINA_FASO</td>
<td>250</td>
<td>BURKINA_FASO</td>
<td>1620</td>
</tr>
<tr>
<td>BURUNDI</td>
<td>260</td>
<td>BURUNDI</td>
<td>1630</td>
</tr>
<tr>
<td>CAMEROON</td>
<td>124</td>
<td>CAMEROON</td>
<td>1640</td>
</tr>
<tr>
<td>CAPE_VERDE</td>
<td>270</td>
<td>CAPE_VERDE</td>
<td>1650</td>
</tr>
<tr>
<td>CENTRAL_AFRICAN_REPUBLIC</td>
<td>280</td>
<td>CENTRAL_AFRICAN_REPUBLIC</td>
<td>1660</td>
</tr>
<tr>
<td>CHAD</td>
<td>290</td>
<td>CHAD</td>
<td>1670</td>
</tr>
<tr>
<td>COMOROS</td>
<td>300</td>
<td>COMOROS</td>
<td>1670</td>
</tr>
<tr>
<td>REPUBLIC_OF_THE_CONGO</td>
<td>310</td>
<td>REPUBLIC_OF_THE_CONGO</td>
<td>1680</td>
</tr>
<tr>
<td>DEMOCRATIC_REPUBLIC_OF_THE_CONGO</td>
<td>320</td>
<td>DEMOCRATIC_REPUBLIC_OF_THE_CONGO</td>
<td>1690</td>
</tr>
<tr>
<td>DJIBOUTI</td>
<td>330</td>
<td>DJIBOUTI</td>
<td>1700</td>
</tr>
<tr>
<td>EGYPT</td>
<td>340</td>
<td>EGYPT</td>
<td>1710</td>
</tr>
<tr>
<td>EQUATORIAL_GUINEA</td>
<td>350</td>
<td>EQUATORIAL_GUINEA</td>
<td>1720</td>
</tr>
<tr>
<td>ERITREA</td>
<td>360</td>
<td>ERITREA</td>
<td>1730</td>
</tr>
<tr>
<td>ETHIOPIA</td>
<td>370</td>
<td>ETHIOPIA</td>
<td>1740</td>
</tr>
<tr>
<td>Country</td>
<td>Code</td>
<td>Country</td>
<td>Code</td>
</tr>
<tr>
<td>------------------</td>
<td>------</td>
<td>------------------</td>
<td>------</td>
</tr>
<tr>
<td>IVORY COAST</td>
<td>114</td>
<td>IVORY COAST</td>
<td>1250</td>
</tr>
<tr>
<td>GABON</td>
<td>380</td>
<td>GABON</td>
<td>1750</td>
</tr>
<tr>
<td>THEGAMBAIA</td>
<td>390</td>
<td>THEGAMBAIA</td>
<td>1760</td>
</tr>
<tr>
<td>GHANA</td>
<td>400</td>
<td>GHANA</td>
<td>1770</td>
</tr>
<tr>
<td>GUINEA</td>
<td>410</td>
<td>GUINEA</td>
<td>1780</td>
</tr>
<tr>
<td>GUINEA_BISSAU</td>
<td>420</td>
<td>GUINEA_BISSAU</td>
<td>1790</td>
</tr>
<tr>
<td>KENYA</td>
<td>98</td>
<td>KENYA</td>
<td>1040</td>
</tr>
<tr>
<td>LESOTHO</td>
<td>430</td>
<td>LESOTHO</td>
<td>1800</td>
</tr>
<tr>
<td>LIBERIA</td>
<td>440</td>
<td>LIBERIA</td>
<td>1810</td>
</tr>
<tr>
<td>LIBYA</td>
<td>450</td>
<td>LIBYA</td>
<td>1820</td>
</tr>
<tr>
<td>MADAGASCAR</td>
<td>460</td>
<td>MADAGASCAR</td>
<td>1830</td>
</tr>
<tr>
<td>MALAWI</td>
<td>470</td>
<td>MALAWI</td>
<td>1840</td>
</tr>
<tr>
<td>MALI</td>
<td>480</td>
<td>MALI</td>
<td>1850</td>
</tr>
<tr>
<td>MAURITANIA</td>
<td>123</td>
<td>MAURITANIA</td>
<td>1340</td>
</tr>
<tr>
<td>MAURITIU</td>
<td>490</td>
<td>MAURITIU</td>
<td>1860</td>
</tr>
<tr>
<td>MOROCCO</td>
<td>101</td>
<td>MOROCCO</td>
<td>1070</td>
</tr>
<tr>
<td>MOZAMBIQUE</td>
<td>500</td>
<td>MOZAMBIQUE</td>
<td>1870</td>
</tr>
<tr>
<td>NAMIBIA</td>
<td>82</td>
<td>NAMIBIA</td>
<td>860</td>
</tr>
<tr>
<td>NIGER</td>
<td>510</td>
<td>NIGER</td>
<td>1880</td>
</tr>
<tr>
<td>NIGERIA</td>
<td>99</td>
<td>NIGERIA</td>
<td>1050</td>
</tr>
<tr>
<td>RWANDA</td>
<td>520</td>
<td>RWANDA</td>
<td>1890</td>
</tr>
<tr>
<td>SAO_TOME_AND_PRINCIPÉ</td>
<td>530</td>
<td>SAO_TOME_AND_PRINCIPÉ</td>
<td>1900</td>
</tr>
<tr>
<td>SENEGAL</td>
<td>540</td>
<td>SENEGAL</td>
<td>1910</td>
</tr>
<tr>
<td>SEYCHELLES</td>
<td>550</td>
<td>SEYCHELLES</td>
<td>1920</td>
</tr>
<tr>
<td>SIERRA_LEONE</td>
<td>560</td>
<td>SIERRA_LEONE</td>
<td>1930</td>
</tr>
<tr>
<td>SOMALIA</td>
<td>570</td>
<td>SOMALIA</td>
<td>1940</td>
</tr>
<tr>
<td>SOUTH_AFRICA</td>
<td>35</td>
<td>SOUTH_AFRICA</td>
<td>390</td>
</tr>
<tr>
<td>SUDAN</td>
<td>580</td>
<td>SUDAN</td>
<td>1950</td>
</tr>
<tr>
<td>SWAZILAND</td>
<td>590</td>
<td>SWAZILAND</td>
<td>1960</td>
</tr>
<tr>
<td>TANZANIA</td>
<td>600</td>
<td>TANZANIA</td>
<td>1970</td>
</tr>
<tr>
<td>TOGO</td>
<td>610</td>
<td>TOGO</td>
<td>1980</td>
</tr>
<tr>
<td>TUNISIA</td>
<td>620</td>
<td>TUNISIA</td>
<td>1990</td>
</tr>
<tr>
<td>UGANDA</td>
<td>630</td>
<td>UGANDA</td>
<td>2000</td>
</tr>
<tr>
<td>WESTERN_SAHARA</td>
<td>640</td>
<td>WESTERN_SAHARA</td>
<td>2010</td>
</tr>
<tr>
<td>ZAIRE</td>
<td>650</td>
<td>ZAIRE</td>
<td>2020</td>
</tr>
<tr>
<td>ZAMBIA</td>
<td>83</td>
<td>ZAMBIA</td>
<td>870</td>
</tr>
<tr>
<td>ZIMBABWE</td>
<td>660</td>
<td>ZIMBABWE</td>
<td>2030</td>
</tr>
</tbody>
</table>

**GENERAL_DOC 7**

<table>
<thead>
<tr>
<th>UNIVERSITY_USA</th>
<th>24</th>
<th>Student Id (UMASS, Boston Un., Emerson Clg., Harvard Un., Northeastern Un., Suffolk Un.)</th>
<th>280</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPLOYMENT_CARDS</td>
<td>26</td>
<td>EMPLOYMENT_CARDS</td>
<td>300</td>
</tr>
</tbody>
</table>
### Silent installation

#### 1.1.1. Wise Installer SDK (Used until November 2014):

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)

---

<table>
<thead>
<tr>
<th>SERVICE_CARDS</th>
<th>ENTERTAINMENT_CARDS</th>
<th>USAPILOTS_CARDS</th>
<th>ACCESS_CARDS</th>
<th>OCB_CARDS</th>
<th>SPAIN_POLICE_CARDS</th>
<th>EHIC_CARDS</th>
<th>SCSIUSAC_CARDS</th>
<th>USAA_CARDS</th>
<th>AMPORT_CARDS</th>
<th>PH_CARDS_CARDS</th>
<th>IRELAND_FIREARM_CARDS</th>
<th>TUNISIA_ELECTION_CARDS</th>
<th>BEAUCES_CARDS</th>
<th>INTERPOL_CARDS</th>
<th>T_MOBILE_CARDS</th>
<th>EASYPAY_CARDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>39</td>
<td>40</td>
<td>42</td>
<td>44</td>
<td>52</td>
<td>61</td>
<td>65</td>
<td>70</td>
<td>75</td>
<td>77</td>
<td>84</td>
<td>96</td>
<td>109</td>
<td>110</td>
<td>117</td>
<td>119</td>
<td>122</td>
</tr>
<tr>
<td>SERVICE_CARDS</td>
<td>ENTERTAINMENT_CARDS</td>
<td>USAPILOTS_CARDS</td>
<td>ACCESS_CARDS</td>
<td>OCB_CARDS</td>
<td>SPAIN_POLICE_CARDS</td>
<td>EHIC_CARDS</td>
<td>SCSIUSAC_CARDS</td>
<td>USAA_CARDS</td>
<td>AMPORT_CARDS</td>
<td>PH_CARDS_CARDS</td>
<td>IRELAND_FIREARM_CARDS</td>
<td>TUNISIA_ELECTION_CARDS</td>
<td>BEAUCES_CARDS</td>
<td>INTERPOL_CARDS</td>
<td>T_MOBILE_CARDS</td>
<td>EASYPAY_CARDS</td>
</tr>
<tr>
<td>39</td>
<td>40</td>
<td>42</td>
<td>44</td>
<td>52</td>
<td>61</td>
<td>65</td>
<td>70</td>
<td>75</td>
<td>77</td>
<td>84</td>
<td>96</td>
<td>109</td>
<td>110</td>
<td>117</td>
<td>119</td>
<td>122</td>
</tr>
</tbody>
</table>
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: IPUran
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=

1.1.2.   Install Shield Installer SDK (Used from February 2014):
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 in silent mode 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.