



# Swelio Library

Belgian Electronic ID card access library



# Table of Contents

<b>Symbol Reference</b>	<b>1</b>
<b>Functions</b>	<b>1</b>
ActivateCard Function	10
ActivateCardEx Function	11
AddFileToContainer Function	11
AddRemoveMessageFilter Function	11
AllocateBuffer Function	12
AllocateDefaultHWNDAs Function	12
AllocateDefaultHWNDW Function	13
AllocateHWNDAs Function	13
AllocateHWNDW Function	13
AllocateLayeredWindowA Function	14
AllocateLayeredWindowW Function	14
AllocateWindowClassA Function	14
AllocateWindowClassW Function	15
AlphaBlendBitmap Function	15
AlphaBlendNative Function	15
BringWindowToFront Function	15
CardDecryptFileA Function	16
CardDecryptFileW Function	16
CardEncryptFileA Function	16
CardEncryptFileW Function	17
CardSignCMS Function	17
CardSignCMSEx Function	18
CardSignCadesT Function	18
CardSignCadesTEx Function	19
CertSignCMS Function	19
CertSignCMSData Function	20
CertSignCMSEx Function	20
CertSignCadesT Function	21
CertSignCadesTData Function	21
CertSignCadesTEx Function	22
CheckMD5 Function	22
CheckSHA1 Function	23
CheckSHA256 Function	23
ClearFileAttributesA Function	24
ClearFileAttributesW Function	24
ClearUnusedMemory Function	25

CloneFont Function	25
ContainerCertificate Function	25
ContainerEidCertificate Function	25
ContainerPickCertificate Function	26
CopyNativeBitmap Function	26
CreateCardBuffer Function	27
CreateNativeBitmap Function	27
CreateUnicodeFileA Function	27
CreateUnicodeFileW Function	27
CreateWindowsFont Function	28
CurrentIPAddressA Function	28
CurrentIPAddressW Function	28
DeactivateCard Function	28
DeactivateCardEx Function	29
DeallocateBuffer Function	29
DeallocateHWND A Function	29
DeallocateHWNDA Function	29
DeallocateHWNDA Function	30
DecryptFileAESA Function	30
DecryptFileAESW Function	31
DeleteCardBuffer Function	31
DeleteToRecycleBinA Function	31
DeleteToRecycleBinW Function	32
DestroyFont Function	32
DestroyImageBuffer Function	32
DirectoryExistsA Function	33
DirectoryExistsW Function	33
DisplayCertificate Function	34
DpiY Function	34
DrawAlphaText Function	34
DrawAlphaTextRect Function	34
DrawLayeredWindow Function	35
DrawNativeBitmap Function	35
DrawTextDirect Function	35
DrawTextDirectEx Function	36
DrawTextGlow Function	36
DrawTextLine Function	36
DrawTextOutline Function	36
DrawTextRect Function	37
EmToPixels Function	37
EmptyRecycleBin Function	37
EncodeCertificate Function	37
EncodePhoto Function	38

EncryptFileAESA Function	38
EncryptFileAESW Function	39
FileCloseA Function	39
FileCloseW Function	39
FileCopyA Function	40
FileCopyW Function	40
FileCreateRewriteA Function	41
FileCreateRewriteW Function	41
FileDeleteA Function	41
FileDeleteW Function	42
FileExistsA Function	42
FileExistsW Function	42
FileExtensionIsA Function	43
FileExtensionIsW Function	43
FileGetSizeA Function	43
FileGetSizeW Function	44
FileIsExeA Function	44
FileIsExeW Function	45
FileIsIconA Function	45
FileIsIconW Function	45
FileIsImageA Function	46
FileIsImageW Function	46
FileIsLink Function	46
FileOrFolderExistsA Function	47
FileOrFolderExistsW Function	47
FileRenameA Function	47
FileRenameW Function	48
FileWriteA Function	48
FileWriteCharA Function	48
FileWriteCharW Function	49
FileWriteNewLineA Function	49
FileWriteNewLineW Function	49
FileWriteW Function	50
FreeContainer Function	50
FullPathA Function	50
FullPathW Function	51
GenerateAuthenticationSignatureA Function	51
GenerateAuthenticationSignatureExA Function	51
GenerateAuthenticationSignatureExW Function	52
GenerateAuthenticationSignatureW Function	53
GenerateBMPA Function	53
GenerateBMPW Function	54

GenerateNonRepudiationSignatureA Function	54
GenerateNonRepudiationSignatureExA Function	55
GenerateNonRepudiationSignatureExW Function	55
GenerateNonRepudiationSignatureW Function	56
GeneratePNGA Function	56
GeneratePNGW Function	57
GenerateQRCodeA Function	57
GenerateQRCodeExA Function	57
GenerateQRCodeExW Function	58
GenerateQRCodeW Function	58
GetAllFiles Function	59
GetCardBufferA Function	59
GetCardBufferSize Function	60
GetCardBufferW Function	60
GetCardSerialNumber Function	60
GetCardVersion Function	61
GetContainerError Function	61
GetContainerErrorsCount Function	62
GetEncodedCertificateSize Function	62
GetEncodedPhotoSize Function	63
GetFileMD5A Function	63
GetFileMD5W Function	63
GetFileSHA1A Function	64
GetFileSHA1W Function	64
GetFileSHA256A Function	65
GetFileSHA256W Function	65
GetFileSHA384A Function	66
GetFileSHA384W Function	66
GetFileSHA512A Function	66
GetFileSHA512W Function	66
GetFilesCountA Function	66
GetFilesCountW Function	67
GetHBitmapA Function	67
GetHBitmapW Function	68
GetISOCODEA Function	68
GetISOCODEW Function	69
GetMD5 Function	69
GetPNGA Function	69
GetPNGW Function	70
GetReaderIndexA Function	70
GetReaderIndexW Function	71
GetReaderNameA Function	71

---

GetReaderNameLenA Function	72
GetReaderNameLenW Function	72
GetReaderNameW Function	72
GetReadersCount Function	73
GetSHA1 Function	73
GetSHA256 Function	73
GetSHA384 Function	74
GetSHA512 Function	74
GetSelectedReaderIndex Function	74
GetStartupA Function	75
GetStartupW Function	75
GetSupportSIS Function	75
GetTextLineSize Function	76
GetTextSize Function	76
GetTextSizeEx Function	76
HibernateWindows Function	76
IgnoreHardwareEvents Function	77
IgnoreServiceEvents Function	77
InitializeContainer Function	77
IsAnimatedGIFA Function	78
IsAnimatedGIFW Function	78
IsCardActivated Function	78
IsCardActivatedEx Function	79
IsCardPresent Function	79
IsCardPresentEx Function	79
IsCardStillInserted Function	80
IsCardStillInsertedEx Function	80
IsCitrixSession Function	80
IsConnectedToInternet Function	80
IsDirectoryA Function	81
IsDirectoryW Function	81
IsEIDCard Function	81
IsEIDCardEx Function	82
IsEngineActive Function	82
IsFemaleA Function	82
IsFemaleW Function	83
IsMaleA Function	83
IsMaleW Function	84
IsMediaCenter Function	84
IsMetroActive Function	84
IsMultiTouchReady Function	84
IsNativeWin64 Function	85

---

IsRemoteSession Function	85
IsSISCard Function	85
IsSISCardEx Function	85
IsTabletPC Function	86
IsUnicodeFileA Function	86
IsUnicodeFileW Function	86
IsValidFileNameA Function	87
IsValidFileNameW Function	87
IsValidPathNameA Function	88
IsValidPathNameW Function	88
IsWindows10 Function	88
IsWindows7 Function	89
IsWindows8 Function	89
IsWindowsVista Function	89
IsWindowsXP Function	89
IsWindowsXPSP2 Function	90
IsWow64 Function	90
LayeredWndProcA Function	90
LayeredWndProcW Function	90
LoadAddressA Function	90
LoadAddressW Function	91
LoadBitmapJPG Function	91
LoadBitmapPNG Function	91
LoadCertificateA Function	91
LoadCertificateW Function	92
LoadIdentityA Function	92
LoadIdentityW Function	92
LoadPNGResource Function	93
LoadPhotoA Function	93
LoadPhotoW Function	93
MakeCompatibleBitmap Function	94
MakeSoundFromFileA Function	94
MakeSoundFromFileW Function	94
MakeSoundFromResourceA Function	95
MakeSoundFromResourceW Function	95
NonRepudiationSignatureAlgoA Function	95
NonRepudiationSignatureAlgoW Function	96
PointsToPixels Function	96
PortAvailable Function	96
ReadAddressA Function	96
ReadAddressExA Function	97
ReadAddressExW Function	97



ReadAddressW Function	97
ReadAuthenticationCertificate Function	98
ReadAuthenticationCertificateEx Function	98
ReadBufferFromFileA Function	99
ReadBufferFromFileW Function	99
ReadCaCertificate Function	99
ReadCaCertificateEx Function	100
ReadIdentityA Function	100
ReadIdentityExA Function	101
ReadIdentityExW Function	101
ReadIdentityW Function	101
ReadNonRepudiationCertificate Function	102
ReadNonRepudiationCertificateEx Function	102
ReadPhoto Function	102
ReadPhotoAsBitmap Function	103
ReadPhotoAsBitmapEx Function	103
ReadPhotoEx Function	104
ReadRootCaCertificate Function	104
ReadRootCaCertificateEx Function	104
ReadRrnCertificate Function	105
ReadRrnCertificateEx Function	105
ReadSISCardA Function	106
ReadSISCardExA Function	106
ReadSISCardExW Function	106
ReadSISCardW Function	107
RecycleBinEmpty Function	107
ReloadReadersList Function	107
RemoveCallback Function	108
RemoveStartupA Function	108
RemoveStartupW Function	108
RestoreWindowSubclassA Function	109
RestoreWindowSubclassW Function	109
SaveAddressA Function	109
SaveAddressW Function	110
SaveAuthenticationCertificateA Function	110
SaveAuthenticationCertificateExW Function	110
SaveAuthenticationCertificateW Function	111
SaveCaCertificateA Function	111
SaveCaCertificateExW Function	111
SaveCaCertificateW Function	112
SaveCardToToXMLStreamExA Function	112
SaveCardToToXMLStreamExW Function	112

SaveCardToXmlA Function	113
SaveCardToXmlExA Function	113
SaveCardToXmlExW Function	114
SaveCardToXmlW Function	114
SaveContainer Function	114
SaveIdentityA Function	115
SaveIdentityW Function	115
SaveNonRepudiationCertificateA Function	116
SaveNonRepudiationCertificateExW Function	116
SaveNonRepudiationCertificateW Function	116
SavePersonCsvToStreamA Function	117
SavePersonCsvToStreamW Function	117
SavePersonToCsvA Function	117
SavePersonToCsvExA Function	118
SavePersonToCsvExW Function	118
SavePersonToCsvW Function	119
SavePhotoA Function	119
SavePhotoAsBitmapA Function	119
SavePhotoAsBitmapExA Function	120
SavePhotoAsBitmapExW Function	120
SavePhotoAsBitmapW Function	121
SavePhotoAsJpegA Function	121
SavePhotoAsJpegExA Function	121
SavePhotoAsJpegExW Function	122
SavePhotoAsJpegW Function	122
SavePhotoW Function	122
SaveRootCaCertificateA Function	123
SaveRootCaCertificateExW Function	123
SaveRootCaCertificateW Function	124
SaveRrnCertificateA Function	124
SaveRrnCertificateExW Function	124
SaveRrnCertificateW Function	125
SelectReader Function	125
SelectReaderByNameA Function	125
SelectReaderByNameW Function	126
SendAPDU Function	126
ServerAccessible Function	126
SetCallback Function	127
SetContainerCanonization Function	127
SetContainerTimeServer Function	128
SetMWCompatibility Function	128
SetStartupA Function	128

SetStartupW Function	129
SetSupportSIS Function	129
ShellCopyFileA Function	129
ShellCopyFileW Function	130
ShutdownWindows Function	130
SignPdfFile Function	131
SignPdfFileEx Function	131
StartEngine Function	131
StopEngine Function	132
StretchNativeBitmap Function	132
StripFileNameA Function	132
StripFileNameW Function	133
SuspendWindows Function	133
TurnMonitorOff Function	133
TurnMonitorOn Function	133
UpdateWindowPosition Function	134
VerifyContainer Function	134
VerifyPinA Function	134
VerifyPinExA Function	135
VerifyPinExW Function	135
VerifyPinW Function	136
VerifySignature Function	136
WriteBufferToFileA Function	136
WriteBufferToFileW Function	137
fpreset Function	137
<b>Structs, Records, Enums</b>	<b>137</b>
structErrorInformation Structure	139
tagCardEventType Enumeration	139
tagEidAddressA Structure	139
tagEidAddressW Structure	140
tagEidCertificate Structure	140
tagEidIdentityA Structure	140
tagEidIdentityExA Structure	142
tagEidIdentityExW Structure	143
tagEidIdentityW Structure	145
tagEidPicture Structure	146
tagSISRecordA Structure	147
tagSISRecordW Structure	148
CardEventType Enumeration	148
EidAddressA Structure	149
EidAddressW Structure	149
EidCertificate Structure	149

EidIdentityA Structure	150
EidIdentityExA Structure	151
EidIdentityExW Structure	153
EidIdentityW Structure	154
EidPicture Structure	156
ErrorInformation Structure	156
PEidAddressA Structure	156
PEidAddressW Structure	157
PEidCertificate Structure	157
PEidIdentityA Structure	157
PEidIdentityExA Structure	159
PEidIdentityExW Structure	160
PEidIdentityW Structure	162
PErrorInformation Structure	163
PSISRecordA Structure	164
PSISRecordW Structure	164
PeidPicture Structure	165
SISRecordA Structure	166
SISRecordW Structure	166
<b>Files</b>	<b>167</b>
Algorithms.h	167
CardEvents.h	168
CardStructures.h	168
Encryption.h	169
FileOperations.h	170
Graphics.h	171
NationalityConverter.h	172
Network.h	172
Swelio.h	173
System.h	178
SystemInfo.h	179
quicol.h	179

## Index

## a

# 1 Symbol Reference

## 1.1 Functions

The following table lists functions in this documentation.

### Functions

	Name	Description
≡	ActivateCard (see page 10)	Established communication between the card and the reader
≡	ActivateCardEx (see page 11)	Established communication between the card and the reader
≡	AddFileToContainer (see page 11)	Add existing file to the container
≡	AddRemoveMessageFilter (see page 11)	Adds or removes a message from the User Interface Privilege Isolation (UIPI) message filter.
≡	AllocateBuffer (see page 12)	Allocates the buffer in memory
≡	AllocateDefaultHWND (see page 12)	This function creates the invisible tool window
≡	AllocateDefaultHWN (see page 13)	This function creates the invisible tool window
≡	AllocateHWND (see page 13)	This function creates the invisible tool window using the provided window procedure
≡	AllocateHWN (see page 13)	This function creates the invisible tool window using the provided window procedure
≡	AllocateLayeredWindowA (see page 14)	This function creates the layered window using the provided window class name
≡	AllocateLayeredWindowW (see page 14)	This function creates the layered window using the provided window class name
≡	AllocateWindowClassA (see page 14)	This function creates the standard window using the provided window class name
≡	AllocateWindowClassW (see page 15)	This function creates the standard window using the provided window class name
≡	AlphaBlendBitmap (see page 15)	This is function AlphaBlendBitmap.
≡	AlphaBlendNative (see page 15)	This is function AlphaBlendNative.
≡	BringWindowToFront (see page 15)	This function brings the specified window to the top of the z-order.
≡	CardDecryptFileA (see page 16)	Decrypt file using Belgian Id card
≡	CardDecryptFileW (see page 16)	Decrypt file using Belgian Id card
≡	CardEncryptFileA (see page 16)	Encrypt file using Belgian Id card
≡	CardEncryptFileW (see page 17)	Encrypt file using Belgian Id card
≡	CardSignCMS (see page 17)	Sign data with eID card according to CMS standard
≡	CardSignCMSEx (see page 18)	Sign data with eID card according to CMS standard
≡	CardSignCadesT (see page 18)	Sign data with eID card according to CADES-T standard
≡	CardSignCadesTEx (see page 19)	Sign data with eID card according to CADES-T standard
≡	CertSignCMS (see page 19)	Sign data with the certificate file according to CMS standard
≡	CertSignCMSData (see page 20)	Sign data with the certificate according to CMS standard
≡	CertSignCMSEx (see page 20)	Sign data with the certificate file according to CMS standard

⇒	CertSignCadesT (see page 21)	Sign data with the certificate file according to CADES-T standard
⇒	CertSignCadesTData (see page 21)	Sign data with the certificate according to CADES-T standard
⇒	CertSignCadesTEx (see page 22)	Sign data with the certificate file according to CADES-T standard
⇒	CheckMD5 (see page 22)	Checks the MD5 hash value of the memory buffer
⇒	CheckSHA1 (see page 23)	Checks the SHA1 hash value of the memory buffer
⇒	CheckSHA256 (see page 23)	Checks the SHA256 hash value of the memory buffer
⇒	ClearFileAttributesA (see page 24)	This function sets the file attributes to normal.
⇒	ClearFileAttributesW (see page 24)	This function sets the file attributes to normal.
⇒	ClearUnusedMemory (see page 25)	Clears unused memory and minimized the application memory usage
⇒	CloneFont (see page 25)	This is function CloneFont.
⇒	ContainerCertificate (see page 25)	Assign certificate for signing ASIC container
⇒	ContainerEidCertificate (see page 25)	Select EID card certificate to sign ASIC container
⇒	ContainerPickCertificate (see page 26)	Pick certificate to sign ASIC container
⇒	CopyNativeBitmap (see page 26)	This is function CopyNativeBitmap.
⇒	CreateCardBuffer (see page 27)	Creates XML buffer
⇒	CreateNativeBitmap (see page 27)	This is function CreateNativeBitmap.
⇒	CreateUnicodeFileA (see page 27)	Creates UNICODE file
⇒	CreateUnicodeFileW (see page 27)	Creates UNICODE file
⇒	CreateWindowsFont (see page 28)	This is function CreateWindowsFont.
⇒	CurrentIPAddressA (see page 28)	Returns the IP address
⇒	CurrentIPAddressW (see page 28)	Returns the IP address
⇒	DeactivateCard (see page 28)	Terminates a connection between a smart card and a reader
⇒	DeactivateCardEx (see page 29)	Terminates a connection between a smart card and a reader
⇒	DeallocateBuffer (see page 29)	Deallocates the memory buffer
⇒	DeallocateHWND (see page 29)	This function destroys the specified window.
⇒	DeallocateHWNDW (see page 30)	This function destroys the specified window.
⇒	DecryptFileAESA (see page 30)	Decrypts file using AES algorithm.
⇒	DecryptFileAESW (see page 31)	Decrypts file using AES algorithm.
⇒	DeleteCardBuffer (see page 31)	Deletes XML buffer
⇒	DeleteToRecycleBinA (see page 31)	Deletes file to the Windows Recycle Bin
⇒	DeleteToRecycleBinW (see page 32)	Deletes file to Windows Recycle Bin
⇒	DestroyFont (see page 32)	This is function DestroyFont.
⇒	DestroyImageBuffer (see page 32)	Destroys the memory buffer
⇒	DirectoryExistsA (see page 33)	Determines whether a specified directory exists.
⇒	DirectoryExistsW (see page 33)	Determines whether a specified directory exists.
⇒	DisplayCertificate (see page 34)	Displays the dialog window with certificate information
⇒	DpiY (see page 34)	This is function DpiY.

◆	DrawAlphaText (see page 34)	This is function DrawAlphaText.
◆	DrawAlphaTextRect (see page 34)	This is function DrawAlphaTextRect.
◆	DrawLayeredWindow (see page 35)	Repaints the surface of the layered window
◆	DrawNativeBitmap (see page 35)	This is function DrawNativeBitmap.
◆	DrawTextDirect (see page 35)	This is function DrawTextDirect.
◆	DrawTextDirectEx (see page 36)	This is function DrawTextDirectEx.
◆	DrawTextGlow (see page 36)	This is function DrawTextGlow.
◆	DrawTextLine (see page 36)	This is function DrawTextLine.
◆	DrawTextOutline (see page 36)	This is function DrawTextOutline.
◆	DrawTextRect (see page 37)	This is function DrawTextRect.
◆	EmToPixels (see page 37)	This is function EmToPixels.
◆	EmptyRecycleBin (see page 37)	Empties the recycle bin
◆	EncodeCertificate (see page 37)	Performs Base64 encoding of the certificate
◆	EncodePhoto (see page 38)	Performs Base64 encoding of the photo
◆	EncryptFileAESA (see page 38)	Encrypts file using AES algorithm.
◆	EncryptFileAESW (see page 39)	Encrypts file using AES algorithm.
◆	FileCloseA (see page 39)	Concludes input/output (I/O) to a file opened using the FileCreateRewrite function.
◆	FileCloseW (see page 39)	Concludes input/output (I/O) to a file opened using the FileCreateRewrite function.
◆	FileCopyA (see page 40)	The CopyFile function copies an existing file to a new file.
◆	FileCopyW (see page 40)	The CopyFile function copies an existing file to a new file.
◆	FileCreateRewriteA (see page 41)	Creates new or overwrites existing file
◆	FileCreateRewriteW (see page 41)	Creates new or overwrites existing file
◆	FileDeleteA (see page 41)	Deletes a file from disk.
◆	FileDeleteW (see page 42)	Deletes a file from disk.
◆	FileExistsA (see page 42)	Tests whether a specified file exists.
◆	FileExistsW (see page 42)	Tests whether a specified file exists.
◆	FileExtensionIsA (see page 43)	Checks the file extension
◆	FileExtensionIsW (see page 43)	Checks the file extension
◆	FileGetSizeA (see page 43)	Retrieves the size of a specified file.
◆	FileGetSizeW (see page 44)	Retrieves the size of a specified file.
◆	FileIsExeA (see page 44)	Checks if the file is a Windows executable
◆	FileIsExeW (see page 45)	Checks if the file is a Windows executable
◆	FileIsIconA (see page 45)	Checks if the file is a Windows icon (.ico) file
◆	FileIsIconW (see page 45)	Checks if the file is a Windows icon (.ico) file
◆	FileIsImageA (see page 46)	Checks if the file is an image file
◆	FileIsImageW (see page 46)	Checks if the file is an image file
◆	FileIsLink (see page 46)	Checks to see if the file specified by file name is a Microsoft Windows shortcut (.Lnk) file (and is neither a file nor a folder).
◆	FileOrFolderExistsA (see page 47)	Checks if the file or folder with the given name exists
◆	FileOrFolderExistsW (see page 47)	Checks if the file or folder with the given name exists
◆	FileRenameA (see page 47)	Renames the file
◆	FileRenameW (see page 48)	Renames the file
◆	FileWriteA (see page 48)	Writes string to the file
◆	FileWriteCharA (see page 48)	Writes one character to the file

✦	FileWriteCharW (see page 49)	Writes one character to the file
✦	FileWriteNewLineA (see page 49)	Writes new line sequence to the file
✦	FileWriteNewLineW (see page 49)	Writes new line sequence to the file
✦	FileWriteW (see page 50)	Writes string to the file
✦	FreeContainer (see page 50)	Deallocates ASIC container
✦	FullPathA (see page 50)	Gets the full path to the file based on file name
✦	FullPathW (see page 51)	Gets the full path to the file based on file name
✦	GenerateAuthenticationSignatureA (see page 51)	Generate authentication signature
✦	GenerateAuthenticationSignatureExA (see page 51)	Generate authentication signature
✦	GenerateAuthenticationSignatureExW (see page 52)	Generate authentication signature
✦	GenerateAuthenticationSignatureW (see page 53)	Generate authentication signature
✦	GenerateBMPA (see page 53)	Generates Windows Bitmap file with QR Code image
✦	GenerateBMPW (see page 54)	Generates Windows Bitmap file with QR Code image
✦	GenerateNonRepudiationSignatureA (see page 54)	Generate non repudiation signature
✦	GenerateNonRepudiationSignatureExA (see page 55)	Generate non repudiation signature
✦	GenerateNonRepudiationSignatureExW (see page 55)	Generate non repudiation signature
✦	GenerateNonRepudiationSignatureW (see page 56)	Generate non repudiation signature
✦	GeneratePNGA (see page 56)	Generates PNG file with QR Code image
✦	GeneratePNGW (see page 57)	Generates PNG file with QR Code image
✦	GenerateQRCodeA (see page 57)	Read eID card and save the identity information and address to PNG QR Code file
✦	GenerateQRCodeExA (see page 57)	Read eID card and save the identity information and address to PNG QR Code file
✦	GenerateQRCodeExW (see page 58)	Read eID card and save the identity information and address to PNG QR Code file
✦	GenerateQRCodeW (see page 58)	Read eID card and save the identity information and address to PNG QR Code file
✦	GetAllFiles (see page 59)	Returns the names of files in a specified directory.
✦	GetCardBufferA (see page 59)	Gets XML or CSV information from the memory buffer
✦	GetCardBufferSize (see page 60)	This function returns the size of the buffer needed to hold the information from the eID card in the XML or CSV format
✦	GetCardBufferW (see page 60)	Gets XML or CSV information from the memory buffer
✦	GetCardSerialNumber (see page 60)	Get the serial number of EID card
✦	GetCardVersion (see page 61)	Get the applet version number for card in the reader with specified number
✦	GetContainerError (see page 61)	Gets the value of the container validation error message based on the message index
✦	GetContainerErrorsCount (see page 62)	Get the number of error messages available in the container validation report
✦	GetEncodedCertificateSize (see page 62)	Returns the size of the Base64 encoded certificate
✦	GetEncodedPhotoSize (see page 63)	Calculates buffer size for Base64 encoded photo
✦	GetFileMD5A (see page 63)	Gets the MD5 hash value for the file
✦	GetFileMD5W (see page 63)	Gets the MD5 hash value for the file



GetFileSHA1A (see page 64)	Gets the SHA1 hash value for the file
GetFileSHA1W (see page 64)	Gets the SHA1 hash value for the file
GetFileSHA256A (see page 65)	Gets the SHA256 hash value for the file
GetFileSHA256W (see page 65)	Gets the SHA256 hash value for the file
GetFileSHA384A (see page 66)	This is function GetFileSHA384A.
GetFileSHA384W (see page 66)	This is function GetFileSHA384W.
GetFileSHA512A (see page 66)	This is function GetFileSHA512A.
GetFileSHA512W (see page 66)	This is function GetFileSHA512W.
GetFilesCountA (see page 66)	Calculates the number of files in the given folder
GetFilesCountW (see page 67)	Calculates the number of files in the given folder
GetHBitmapA (see page 67)	Generates Windows Bitmap in memory with QR Code image
GetHBitmapW (see page 68)	Generates Windows Bitmap in memory with QR Code image
GetISOCodeA (see page 68)	Returns the country ISO code based on the nationality string
GetISOCodeW (see page 69)	Returns the country ISO code based on the nationality string
GetMD5 (see page 69)	Gets the MD5 hash value for the content of the memory buffer
GetPNGA (see page 69)	Writes PNG image to the memory buffer.
GetPNGW (see page 70)	Writes PNG image to the memory buffer.
GetReaderIndexA (see page 70)	Returns the zero-based reader index with specified name
GetReaderIndexW (see page 71)	Returns the zero-based reader index with specified name
GetReaderNameA (see page 71)	Returns the name of the card reader
GetReaderNameLenA (see page 72)	Returns the length of the reader name
GetReaderNameLenW (see page 72)	Returns the length of the reader name
GetReaderNameW (see page 72)	Returns the name of the card reader
GetReadersCount (see page 73)	Get number of card readers connected to PC
GetSHA1 (see page 73)	Gets the SHA1 hash value for the content of the memory buffer
GetSHA256 (see page 73)	Gets the SHA256 hash value for the content of the memory buffer
GetSHA384 (see page 74)	This is function GetSHA384.
GetSHA512 (see page 74)	This is function GetSHA512.
GetSelectedReaderIndex (see page 74)	Returns the index of the active smart card reader
GetStartupA (see page 75)	Checks if the application is registered to run when Windows starts
GetStartupW (see page 75)	Checks if the application is registered to run when Windows starts
GetSupportSIS (see page 75)	Checks if the SIS cards are supported by the engine
GetTextLineSize (see page 76)	This is function GetTextLineSize.
GetTextSize (see page 76)	This is function GetTextSize.
GetTextSizeEx (see page 76)	This is function GetTextSizeEx.
HibernateWindows (see page 76)	Hibernates Windows
IgnoreHardwareEvents (see page 77)	Ignore USB reader insert / remove events
IgnoreServiceEvents (see page 77)	Ignore smartcard service stop events when reporting readers list change
InitializeContainer (see page 77)	Initializes ASIC container
IsAnimatedGIFA (see page 78)	Checks if the file is an animated GIF image file
IsAnimatedGIFW (see page 78)	Checks if the file is an animated GIF image file
IsCardActivated (see page 78)	Checks the connection between a smart card and a reader
IsCardActivatedEx (see page 79)	Checks the connection between a smart card and a reader
IsCardPresent (see page 79)	Checks if the card is present in the card reader
IsCardPresentEx (see page 79)	Checks if the card is present in the card reader
IsCardStillInserted (see page 80)	Checks if the card is still inserted in the card reader
IsCardStillInsertedEx (see page 80)	Checks if the card is still inserted in the card reader

IsCitrixSession (see page 80)	Checks if application is running in Citrix session
IsConnectedToInternet (see page 80)	Checks if PC is connected to Internet
IsDirectoryA (see page 81)	Verifies that a path is a valid directory.
IsDirectoryW (see page 81)	Verifies that a path is a valid directory.
IsEIDCard (see page 81)	Check if Belgian EID card is inserted into card reader
IsEIDCardEx (see page 82)	Check if Belgian EID card is inserted into card reader
IsEngineActive (see page 82)	Checks if the Swelio Engine is activated
IsFemaleA (see page 82)	Checks if the card owner is female
IsFemaleW (see page 83)	Checks if the card owner is female
IsMaleA (see page 83)	Checks if the card owner is male
IsMaleW (see page 84)	Checks if the card owner is male
IsMediaCenter (see page 84)	Checks if the Media Center version of Windows is installed
IsMetroActive (see page 84)	Checks if metro interface is active
IsMultiTouchReady (see page 84)	Checks if the system is multi touch ready
IsNativeWin64 (see page 85)	Checks if the application is native 64 bit executable
IsRemoteSession (see page 85)	Checks if application is running in RDP session
IsSISCard (see page 85)	Check if Belgian SIS card is inserted into card reader
IsSISCardEx (see page 85)	Check if Belgian SIS card is inserted into card reader
IsTabletPC (see page 86)	Checks if the application is running on the Tablet PC
IsUnicodeFileA (see page 86)	Checks if the file is UNICODE file
IsUnicodeFileW (see page 86)	Checks if the file is UNICODE file
IsValidFileNameA (see page 87)	Checks if provided string is a valid file name
IsValidFileNameW (see page 87)	Checks if provided string is a valid file name
IsValidPathNameA (see page 88)	Checks if provided string is a valid file path
IsValidPathNameW (see page 88)	Checks if provided string is a valid file path
IsWindows10 (see page 88)	Checks if PC is running Windows 10 or better
IsWindows7 (see page 89)	Checks if PC is running Windows 7 or better
IsWindows8 (see page 89)	Checks if PC is Running Windows 8 or better
IsWindowsVista (see page 89)	Checks if PC is running Windows Vista or better
IsWindowsXP (see page 89)	Checks if PC is running Windows XP
IsWindowsXPSP2 (see page 90)	Checks if PC is running Windows XP with Service Pack 2 installed
IsWow64 (see page 90)	Checks if the 32 bit application runs on 64 bit Windows
LayeredWndProcA (see page 90)	The default window procedure for the layered window
LayeredWndProcW (see page 90)	The default window procedure for the layered window
LoadAddressA (see page 90)	This is function LoadAddressA.
LoadAddressW (see page 91)	This is function LoadAddressW.
LoadBitmapJPG (see page 91)	This is function LoadBitmapJPG.
LoadBitmapPNG (see page 91)	This is function LoadBitmapPNG.
LoadCertificateA (see page 91)	Reads the certificate from a file
LoadCertificateW (see page 92)	Reads the certificate from a file
LoadIdentityA (see page 92)	Reads the raw identity information from a file
LoadIdentityW (see page 92)	Reads the raw identity information from a file
LoadPNGResource (see page 93)	This is function LoadPNGResource.
LoadPhotoA (see page 93)	Loads photo from a file
LoadPhotoW (see page 93)	Loads photo from a file
MakeCompatibleBitmap (see page 94)	This is function MakeCompatibleBitmap.
MakeSoundFromFileA (see page 94)	Plays the wave sound from the file
MakeSoundFromFileW (see page 94)	Plays the wave sound from the file

◆	MakeSoundFromResourceA (see page 95)	Plays the wave sound from the resource
◆	MakeSoundFromResourceW (see page 95)	Plays the wave sound from the resource
◆	NonRepudiationSignatureAlgoA (see page 95)	This is function NonRepudiationSignatureAlgoA.
◆	NonRepudiationSignatureAlgoW (see page 96)	This is function NonRepudiationSignatureAlgoW.
◆	PointsToPixels (see page 96)	This is function PointsToPixels.
◆	PortAvailable (see page 96)	Checks if the port with specified number is available
◆	ReadAddressA (see page 96)	Read address information from Belgian eID card
◆	ReadAddressExA (see page 97)	Read address information from Belgian eID card
◆	ReadAddressExW (see page 97)	Read address information from Belgian eID card
◆	ReadAddressW (see page 97)	Read address information from Belgian eID card
◆	ReadAuthenticationCertificate (see page 98)	Read Authentication Certificate to memory
◆	ReadAuthenticationCertificateEx (see page 98)	Read Authentication Certificate to memory
◆	ReadBufferFromFileA (see page 99)	Reads the content of the file to the memory buffer
◆	ReadBufferFromFileW (see page 99)	Reads the content of the file to the memory buffer
◆	ReadCaCertificate (see page 99)	Read Ca Certificate to memory
◆	ReadCaCertificateEx (see page 100)	Read Ca Certificate to memory
◆	ReadIdentityA (see page 100)	Read identity information from Belgian eID card
◆	ReadIdentityExA (see page 101)	Read identity information from Belgian eID card
◆	ReadIdentityExW (see page 101)	Read identity information from Belgian eID card
◆	ReadIdentityW (see page 101)	Read identity information from Belgian eID card
◆	ReadNonRepudiationCertificate (see page 102)	Read Non Repudiation Certificate to memory
◆	ReadNonRepudiationCertificateEx (see page 102)	Read Non Repudiation Certificate to memory
◆	ReadPhoto (see page 102)	Reads a photo from a card
◆	ReadPhotoAsBitmap (see page 103)	Reads the picture from the card, converts it to bitmap and returns the bitmap handle Description: Reads the photo from the Belgian eID card and returns the bitmap handle Reading the photo from the card is a time consuming operation. Do it only when needed.
◆	ReadPhotoAsBitmapEx (see page 103)	Reads the picture from the card, converts it to bitmap and returns the bitmap handle Description: Reads the photo from the Belgian eID card and returns the Windows bitmap handle Reading the photo from the card is a time consuming operation. Do it only when needed.
◆	ReadPhotoEx (see page 104)	Reads a photo from a card
◆	ReadRootCaCertificate (see page 104)	Read Root Ca Certificate to memory
◆	ReadRootCaCertificateEx (see page 104)	Read Root Ca Certificate to memory
◆	ReadRrnCertificate (see page 105)	Read Rrn Certificate to memory
◆	ReadRrnCertificateEx (see page 105)	Read Rrn Certificate to memory
◆	ReadSISCardA (see page 106)	Read Belgian SIS card.
◆	ReadSISCardExA (see page 106)	Read Belgian SIS card.
◆	ReadSISCardExW (see page 106)	Read Belgian SIS card.
◆	ReadSISCardW (see page 107)	Read Belgian SIS card.
◆	RecycleBinEmpty (see page 107)	Returns TRUE if Windows Recycle Bin is empty
◆	ReloadReadersList (see page 107)	Reloads the list of the available card readers
◆	RemoveCallback (see page 108)	Remove callback procedure for card events

RemoveStartupA (see page 108)	Removes the application from the list of the automatically started applications
RemoveStartupW (see page 108)	Removes the application from the list of the automatically started applications
RestoreWindowSubclassA (see page 109)	Restores window standard procedure
RestoreWindowSubclassW (see page 109)	Restores window standard procedure
SaveAddressA (see page 109)	This is function SaveAddressA.
SaveAddressW (see page 110)	This is function SaveAddressW.
SaveAuthenticationCertificateA (see page 110)	Save Authentication Certificate to a file
SaveAuthenticationCertificateExW (see page 110)	Save Authentication Certificate to a file
SaveAuthenticationCertificateW (see page 111)	Save Authentication Certificate to a file
SaveCaCertificateA (see page 111)	Save Ca Certificate to a file
SaveCaCertificateExW (see page 111)	Save Ca Certificate to a file
SaveCaCertificateW (see page 112)	Save Ca Certificate to a file
SaveCardToToXMLStreamExA (see page 112)	Read eID card and save the information to XML buffer
SaveCardToToXMLStreamExW (see page 112)	Read eID card and save the information to XML buffer
SaveCardToXmlA (see page 113)	Read eID card and save the information to XML file
SaveCardToXmlExA (see page 113)	Read eID card and save the information to XML file
SaveCardToXmlExW (see page 114)	Read eID card and save the information to XML file
SaveCardToXmlW (see page 114)	Read eID card and save the information to XML file
SaveContainer (see page 114)	Save container to the file
SaveIdentityA (see page 115)	Saves identity information to a file
SaveIdentityW (see page 115)	Saves identity information to a file
SaveNonRepudiationCertificateA (see page 116)	Save Non Repudiation Certificate to a file
SaveNonRepudiationCertificateExW (see page 116)	Save Non Repudiation Certificate to a file
SaveNonRepudiationCertificateW (see page 116)	Save Non Repudiation Certificate to a file
SavePersonCsvToStreamA (see page 117)	Read eID card and save the identity information to CSV memory buffer
SavePersonCsvToStreamW (see page 117)	Read eID card and save the identity information to CSV memory buffer
SavePersonToCsvA (see page 117)	Read eID card and save the identity information and address to CSV file
SavePersonToCsvExA (see page 118)	Read eID card and save the identity information and address to CSV file
SavePersonToCsvExW (see page 118)	Read eID card and save the identity information and address to CSV file
SavePersonToCsvW (see page 119)	Read eID card and save the identity information and address to CSV file
SavePhotoA (see page 119)	Save photo to a file
SavePhotoAsBitmapA (see page 119)	Save the picture from the card to Windows Bitmap file Description: Reads the photo from the Belgian eID card and writes it to the file as bitmap image. Reading the photo from the card is a time consuming operation. Do it only when needed.

SavePhotoAsBitmapExA (see page 120)	Reads the picture from the card and saves it to Windows Bitmap file Description: Reads the photo from the Belgian eID card and writes it to the file as bitmap image. Reading the photo from the card is a time consuming operation. Do it only when needed.
SavePhotoAsBitmapExW (see page 120)	Reads the picture from the card and saves it to Windows Bitmap file Description: Reads the photo from the Belgian eID card and writes it to the file as bitmap image. Reading the photo from the card is a time consuming operation. Do it only when needed.
SavePhotoAsBitmapW (see page 121)	Save the picture from the card to Windows Bitmap file Description: Reads the photo from the Belgian eID card and writes it to the file as bitmap image. Reading the photo from the card is a time consuming operation. Do it only when needed.
SavePhotoAsJpegA (see page 121)	Save the picture from the card to JPG file Description: Reads the photo from the Belgian eID card and writes it to the file as JPG image. Reading the photo from the card is a time consuming operation. Do it only when needed.
SavePhotoAsJpegExA (see page 121)	Save the picture from the card to JPG file Description: Reads the photo from the Belgian eID card and writes it to the file as JPG image. Reading the photo from the card is a time consuming operation. Do it only when needed.
SavePhotoAsJpegExW (see page 122)	Save the picture from the card to JPG file Description: Reads the photo from the Belgian eID card and writes it to the file as JPG image. Reading the photo from the card is a time consuming operation. Do it only when needed.
SavePhotoAsJpegW (see page 122)	Save the picture from the card to JPG file Description: Reads the photo from the Belgian eID card and writes it to the file as JPG image. Reading the photo from the card is a time consuming operation. Do it only when needed.
SavePhotoW (see page 122)	Saves photo to a file
SaveRootCaCertificateA (see page 123)	Save Root Ca Certificate to a file
SaveRootCaCertificateExW (see page 123)	Save Root Ca Certificate to a file
SaveRootCaCertificateW (see page 124)	Save Root Ca Certificate to a file
SaveRrnCertificateA (see page 124)	Save RRN Certificate to a file
SaveRrnCertificateExW (see page 124)	Save RRN Certificate to a file
SaveRrnCertificateW (see page 125)	Save RRN Certificate to a file
SelectReader (see page 125)	When more than 1 reader connected, select the reader with specified number
SelectReaderByNameA (see page 125)	Select active smart card reader by providing the reader name
SelectReaderByNameW (see page 126)	Select active smart card reader by providing the reader name
SendAPDU (see page 126)	This is function SendAPDU.
ServerAccessible (see page 126)	Check if the specified server is accessible
SetCallback (see page 127)	Activates callback procedure for card status change event
SetContainerCanonization (see page 127)	Set XML canonization standard of the ASIC container
SetContainerTimeServer (see page 128)	Set time server URI for digital signature of the ASIC container
SetMWCompatibility (see page 128)	Set the compatibility mode with the old version of the official EID MiddleWare
SetStartupA (see page 128)	Register application to run when Windows starts
SetStartupW (see page 129)	Register application to run when Windows starts

◆	SetSupportSIS ( see page 129)	Activates or deactivates SIS card support by engine
◆	ShellCopyFileA ( see page 129)	Copies file to the new location
◆	ShellCopyFileW ( see page 130)	Copies file to the new location
◆	ShutdownWindows ( see page 130)	Logs off the interactive user, shuts down the system.
◆	SignPdfFile ( see page 131)	Digitally sign PDF file using Belgian EID card
◆	SignPdfFileEx ( see page 131)	Digitally sign PDF file using Belgian EID card
◆	StartEngine ( see page 131)	Activates the Swelio Engine.
◆	StopEngine ( see page 132)	Deactivates the Swelio Engine
◆	StretchNativeBitmap ( see page 132)	This is function StretchNativeBitmap.
◆	StripFileNameA ( see page 132)	Replaces environment variable names with values
◆	StripFileNameW ( see page 133)	Replaces environment variable names with values
◆	SuspendWindows ( see page 133)	Suspends Windows
◆	TurnMonitorOff ( see page 133)	Turns the monitor off
◆	TurnMonitorOn ( see page 133)	Turns the monitor on
◆	UpdateWindowPosition ( see page 134)	Updated the window position
◆	VerifyContainer ( see page 134)	Verify the signatures and the integrity of the ASIC container
◆	VerifyPinA ( see page 134)	Verify PIN code
◆	VerifyPinExA ( see page 135)	Verify PIN code
◆	VerifyPinExW ( see page 135)	Verify PIN code
◆	VerifyPinW ( see page 136)	Verify PIN code
◆	VerifySignature ( see page 136)	Verifies the signature from the specified hash value.
◆	WriteBufferToFileA ( see page 136)	Writes the memory buffer to file
◆	WriteBufferToFileW ( see page 137)	Writes the memory buffer to file
◆	fpreset ( see page 137)	This is function fpreset.

## 1.1.1 ActivateCard Function

Established communication between the card and the reader

### C++

```
BOOL WINAPI ActivateCard( );
```

### File

**File:** Swelio.h ( see page 173)

### Returns

Returns TRUE if the card is activated, otherwise returns FALSE

### Description

The ActivateCard function establishes a connection between the calling application and a smart card contained by a specific reader. If no card exists in the specified reader, an error is returned.

### Example

```
if (IsCardPresent())
{
    ActivateCard();
}
```



## 1.1.2 ActivateCardEx Function

Established communication between the card and the reader

### C++

```
BOOL WINAPI ActivateCardEx(int readerNumber);
```

### File

File: Swelio.h (see page 173)

### Parameters

Parameters	Description
int readerNumber	The zero-based index of the card reader

### Returns

Returns TRUE if the card is activated, otherwise returns FALSE

### Description

The ActivateCard (see page 10) function establishes a connection between the calling application and a smart card contained by a specific reader. If no card exists in the specified reader, an error is returned.

## 1.1.3 AddFileToContainer Function

Add existing file to the container

### C++

```
BOOL WINAPI AddFileToContainer(LPVOID container, LPSTR fileName);
```

### File

File: Swelio.h (see page 173)

### Parameters

Parameters	Description
LPVOID container	Container handle, which is allocated by calling InitializeContainer (see page 77) function
LPSTR fileName	The name of the file which will be added to the container

### Returns

Returns true if the operation is successful, otherwise returns false

### Description

Call this function to include the file to the container. The file must exist.

## 1.1.4 AddRemoveMessageFilter Function

Adds or removes a message from the User Interface Privilege Isolation (UIPI) message filter.

### C++

```
void WINAPI AddRemoveMessageFilter(UINT message, DWORD dwFlags);
```

**File****File:** System.h (see page 178)**Parameters**

Parameters	Description
UINT message	Specifies the message to add to or remove from the filter.
DWORD dwFlags	Specifies the action to be performed. One of the following values.

**Description**

This function changes the message filter for Windows Vista or better. UIPI is a security feature that prevents messages from being received from a lower integrity level sender. All such messages with a value above WM\_USER are blocked by default. The filter, somewhat contrary to intuition, is a list of messages that are allowed through. Therefore, adding a message to the filter allows that message to be received from a lower integrity sender, while removing a message blocks that message from being received.

Certain messages with a value less than WM\_USER are required to pass through the filter regardless of the filter setting. You can call this function to remove one of those messages from the filter and it will return TRUE. However, the message will still be received by the calling process.

## 1.1.5 AllocateBuffer Function

Allocates the buffer in memory

**C++**

```
void* WINAPI AllocateBuffer(int bufferSize);
```

**File****File:** FileOperations.h (see page 170)**Parameters**

Parameters	Description
int bufferSize	The size of the buffer

**Returns**

The pointer to the allocated memory block

**Description**

AllocateBuffer allocates a block of the given size on the heap, and returns the address of this memory. The bytes of the allocated buffer are not set to zero. To dispose of the buffer, use DeallocateBuffer (see page 29) function.

## 1.1.6 AllocateDefaultHWNDAs Function

This function creates the invisible tool window

**C++**

```
HWND WINAPI AllocateDefaultHWNDAs();
```

**File****File:** System.h (see page 178)



**Returns**

If the function succeeds, the return value is a handle to the new window. If the function fails, the return value is NULL.

**Description**

This function creates the invisible zero-size tool window that can be used for internal purposes, like processing the special Windows messages for synchronization, etc...

## 1.1.7 AllocateDefaultHWNDW Function

This function creates the invisible tool window

**C++**

```
HWND WINAPI AllocateDefaultHWNDW( );
```

**File**

**File:** System.h ([see page 178](#))

**Returns**

If the function succeeds, the return value is a handle to the new window. If the function fails, the return value is NULL.

**Description**

This function creates the invisible zero-size tool window that can be used for internal purposes, like processing the special Windows messages for synchronization, etc...

## 1.1.8 AllocateHWNDW Function

This function creates the invisible tool window using the provided window procedure

**C++**

```
HWND WINAPI AllocateHWNDW(LONG_PTR method);
```

**File**

**File:** System.h ([see page 178](#))

**Returns**

If the function succeeds, the return value is a handle to the new window. If the function fails, the return value is NULL.

**Description**

This function creates the invisible zero-size tool window that can be used for internal purposes, like processing the special Windows messages for synchronization, etc...

## 1.1.9 AllocateHWNDW Function

This function creates the invisible tool window using the provided window procedure

**C++**

```
HWND WINAPI AllocateHWNDW(LONG_PTR method);
```

**File**

**File:** System.h ([see page 178](#))

**Returns**

If the function succeeds, the return value is a handle to the new window. If the function fails, the return value is NULL.

**Description**

This function creates the invisible zero-size tool window that can be used for internal purposes, like processing the special Windows messages for synchronization, etc...

## 1.1.10 AllocateLayeredWindowA Function

This function creates the layered window using the provided window class name

**C++**

```
HWND WINAPI AllocateLayeredWindowA(LPCSTR className);
```

**File**

**File:** System.h ([see page 178](#))

**Returns**

If the function succeeds, the return value is a handle to the new window. If the function fails, the return value is NULL.

**Description**

This function creates the layered window using the provided window class name

## 1.1.11 AllocateLayeredWindowW Function

This function creates the layered window using the provided window class name

**C++**

```
HWND WINAPI AllocateLayeredWindowW(LPCWSTR className);
```

**File**

**File:** System.h ([see page 178](#))

**Returns**

If the function succeeds, the return value is a handle to the new window. If the function fails, the return value is NULL.

**Description**

This function creates the layered window using the provided window class name

## 1.1.12 AllocateWindowClassA Function

This function creates the standard window using the provided window class name

**C++**

```
HWND WINAPI AllocateWindowClassA(LPCSTR className);
```

**File**

**File:** System.h ([see page 178](#))

**Returns**

If the function succeeds, the return value is a handle to the new window. If the function fails, the return value is NULL.

**Description**

This function creates the standard window using the provided window class name

## 1.1.13 AllocateWindowClassW Function

This function creates the standard window using the provided window class name

**C++**

```
HWND WINAPI AllocateWindowClassW(LPCWSTR className);
```

**File**

**File:** System.h (🔗 see page 178)

**Returns**

If the function succeeds, the return value is a handle to the new window. If the function fails, the return value is NULL.

**Description**

This function creates the standard window using the provided window class name

## 1.1.14 AlphaBlendBitmap Function

This is function AlphaBlendBitmap.

**C++**

```
void WINAPI AlphaBlendBitmap(HBITMAP src, HDC hdc, int left, int top, int width, int height, int alpha);
```

**File**

**File:** Graphics.h (🔗 see page 171)

## 1.1.15 AlphaBlendNative Function

This is function AlphaBlendNative.

**C++**

```
void WINAPI AlphaBlendNative(HDC hdcDest, int xoriginDest, int yoriginDest, int wDest, int hDest, HDC hdcSrc, int xoriginSrc, int yoriginSrc, int wSrc, int hSrc, BYTE alpha);
```

**File**

**File:** Graphics.h (🔗 see page 171)

## 1.1.16 BringWindowToFront Function

This function brings the specified window to the top of the z-order.

**C++**

```
void WINAPI BringWindowToFront (HWND window);
```

**File**

**File:** System.h (🔗 see page 178)

**Parameters**

Parameters	Description
HWND window	Handle to the window to bring to the top of the z-order.

## 1.1.17 CardDecryptFileA Function

Decrypt file using Belgian Id card

**C++**

```
BOOL WINAPI CardDecryptFileA (LPSTR szSource, LPSTR szDestination);
```

**File**

**File:** Encryption.h (🔗 see page 169)

**Parameters**

Parameters	Description
LPSTR szSource	The name of the encrypted file
LPSTR szDestination	The name of the decrypted file

**Description**

Decrypt file which was encrypted using CardEncryptFile function

## 1.1.18 CardDecryptFileW Function

Decrypt file using Belgian Id card

**C++**

```
BOOL WINAPI CardDecryptFileW (LPWSTR szSource, LPWSTR szDestination);
```

**File**

**File:** Encryption.h (🔗 see page 169)

**Parameters**

Parameters	Description
LPWSTR szSource	The name of the encrypted file
LPWSTR szDestination	The name of the decrypted file

**Description**

Decrypt file which was encrypted using CardEncryptFile function

## 1.1.19 CardEncryptFileA Function

Encrypt file using Belgian Id card

**C++**

```
BOOL WINAPI CardEncryptFileA(LPSTR szSource, LPSTR szDestination);
```

**File**

**File:** Encryption.h (see page 169)

**Parameters**

Parameters	Description
LPSTR szSource	The name of the source file
LPSTR szDestination	The name of the encrypted file

**Description**

Encrypt file using Belgian Id card. The card must be inserted in the reader

## 1.1.20 CardEncryptFileW Function

Encrypt file using Belgian Id card

**C++**

```
BOOL WINAPI CardEncryptFileW(LPWSTR szSource, LPWSTR szDestination);
```

**File**

**File:** Encryption.h (see page 169)

**Parameters**

Parameters	Description
LPWSTR szSource	The name of the source file
LPWSTR szDestination	The name of the encrypted file

**Description**

Encrypt file using Belgian Id card. The card must be inserted in the reader

## 1.1.21 CardSignCMS Function

Sign data with eID card according to CMS standard

**C++**

```
BOOL WINAPI CardSignCMS(int readerNumber, BYTE * data, UINT dataLen, BYTE * signature, UINT * signatureLen);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader.
BYTE * data	the data to sign
UINT dataLen	the size of the data buffer
BYTE * signature	the signature buffer
UINT * signatureLen	the size of the signature buffer

**Returns**

Returns true if the operation is successful, otherwise returns false

**Description**

Create CMS signature for data buffer. Can be used for digital signature of PDF documents in combination with external PDF library

## 1.1.22 CardSignCMSEx Function

Sign data with eID card according to CMS standard

**C++**

```
BOOL WINAPI CardSignCMSEx(int readerNumber, BYTE* data, UINT dataLen, BYTE* signature,
UINT* signatureLen, ErrorInformation* Error);
```

**File**

File: Swelio.h (see page 173)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader.
BYTE* data	the data to sign
UINT dataLen	the size of the data buffer
BYTE* signature	the signature buffer
UINT* signatureLen	the size of the signature buffer
error	error information

**Returns**

Returns true if the operation is successful, otherwise returns false

**Description**

Create CMS signature for data buffer. Can be used for digital signature of PDF documents in combination with external PDF library

## 1.1.23 CardSignCadesT Function

Sign data with eID card according to CADES-T standard

**C++**

```
BOOL WINAPI CardSignCadesT(int readerNumber, BYTE * data, UINT dataLen, BYTE * signature,
UINT * signatureLen);
```

**File**

File: Swelio.h (see page 173)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader.
BYTE * data	the data to sign
UINT dataLen	the size of the data buffer

BYTE * signature	the signature buffer
UINT * signatureLen	the size of the signature buffer

**Returns**

Returns true if the operation is successful, otherwise returns false

**Description**

Create CADES-T signature for data buffer. Can be used for digital signature of PDF documents in combination with external PDF library

## 1.1.24 CardSignCadesTEx Function

Sign data with eID card according to CADES-T standard

**C++**

```
BOOL WINAPI CardSignCadesTEx(int readerNumber, BYTE* data, UINT dataLen, BYTE* signature,
UINT* signatureLen, ErrorInformation* error);
```

**File**

File: Swelio.h (see page 173)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader.
BYTE* data	the data to sign
UINT dataLen	the size of the data buffer
BYTE* signature	the signature buffer
UINT* signatureLen	the size of the signature buffer
ErrorInformation* error	error information

**Returns**

Returns true if the operation is successful, otherwise returns false

**Description**

Create CADES-T signature for data buffer. Can be used for digital signature of PDF documents in combination with external PDF library

## 1.1.25 CertSignCMS Function

Sign data with the certificate file according to CMS standard

**C++**

```
BOOL WINAPI CertSignCMS(LPWSTR certificate, LPWSTR password, BYTE * data, UINT dataLen,
BYTE * signature, UINT * signatureLen);
```

**File**

File: Swelio.h (see page 173)

**Parameters**

Parameters	Description
LPWSTR certificate	The name of the certificate file

LPWSTR password	The private key password
BYTE * data	the data to sign
UINT dataLen	the size of the data buffer
BYTE * signature	the signature buffer
UINT * signatureLen	the size of the signature buffer

### Returns

Returns true if the operation is successful, otherwise returns false

### Description

Create CMS signature for data buffer. Can be used for digital signature of PDF documents in combination with external PDF library

## 1.1.26 CertSignCMSData Function

Sign data with the certificate according to CMS standard

### C++

```
BOOL WINAPI CertSignCMSData(BYTE* certificate, DWORD certLen, LPWSTR password, BYTE* data,
UINT dataLen, BYTE* signature, UINT* signatureLen, ErrorInformation* error);
```

### File

File: Swelio.h (see page 173)

### Parameters

Parameters	Description
BYTE* certificate	The byte array with the signing certificate content
DWORD certLen	The size of the signing certificate
LPWSTR password	The password of the signing certificate
BYTE* data	The data to be signed
UINT dataLen	The size of the data to be signed
BYTE* signature	The value of the signature
UINT* signatureLen	The size of the signature
ErrorInformation* error	The error information

### Returns

Returns true if the operation is successful, otherwise returns false

## 1.1.27 CertSignCMSEx Function

Sign data with the certificate file according to CMS standard

### C++

```
BOOL WINAPI CertSignCMSEx(LPWSTR certificate, LPWSTR password, BYTE* data, UINT dataLen,
BYTE* signature, UINT* signatureLen, ErrorInformation* error);
```

### File

File: Swelio.h (see page 173)



**Parameters**

Parameters	Description
LPWSTR certificate	The name of the certificate file
LPWSTR password	The private key password
BYTE* data	the data to sign
UINT dataLen	the size of the data buffer
BYTE* signature	the signature buffer
UINT* signatureLen	the size of the signature buffer
ErrorInformation* error	error information

**Returns**

Returns true if the operation is successful, otherwise returns false

**Description**

Create CMS signature for data buffer. Can be used for digital signature of PDF documents in combination with external PDF library

## 1.1.28 CertSignCadesT Function

Sign data with the certificate file according to CADES-T standard

**C++**

```
BOOL WINAPI CertSignCadesT(LPWSTR certificate, LPWSTR password, BYTE * data, UINT dataLen,
    BYTE * signature, UINT * signatureLen);
```

**File**

File: Swelio.h (see page 173)

**Parameters**

Parameters	Description
LPWSTR certificate	The name of the certificate file
LPWSTR password	The private key password
BYTE * data	the data to sign
UINT dataLen	the size of the data buffer
BYTE * signature	the signature buffer
UINT * signatureLen	the size of the signature buffer

**Returns**

Returns true if the operation is successful, otherwise returns false

**Description**

Create CADES-T signature for data buffer. Can be used for digital signature of PDF documents in combination with external PDF library

## 1.1.29 CertSignCadesTData Function

Sign data with the certificate according to CADES-T standard

**C++**

```
BOOL WINAPI CertSignCadesTData(BYTE* certificate, DWORD certLen, LPWSTR password, BYTE*
```

```
data, UINT dataLen, BYTE* signature, UINT* signatureLen, ErrorInformation* error);
```

**File**

**File:** Swelio.h (📄 see page 173)

**Parameters**

Parameters	Description
BYTE* certificate	The byte array with the signing certificate content
DWORD certLen	The size of the signing certificate
LPWSTR password	The password of the signing certificate
BYTE* data	The data to be signed
UINT dataLen	The size of the data to be signed
BYTE* signature	The value of the signature
UINT* signatureLen	The size of the signature
ErrorInformation* error	The error information

**Returns**

Returns true if the operation is successful, otherwise returns false

## 1.1.30 CertSignCadesTEx Function

Sign data with the certificate file according to CADES-T standard

**C++**

```
BOOL WINAPI CertSignCadesTEx(LPWSTR certificate, LPWSTR password, BYTE* data, UINT dataLen,  
BYTE* signature, UINT* signatureLen, ErrorInformation* error);
```

**File**

**File:** Swelio.h (📄 see page 173)

**Parameters**

Parameters	Description
LPWSTR certificate	The name of the certificate file
LPWSTR password	The private key password
BYTE* data	the data to sign
UINT dataLen	the size of the data buffer
BYTE* signature	the signature buffer
UINT* signatureLen	the size of the signature buffer
ErrorInformation* error	error information

**Returns**

Returns true if the operation is successful, otherwise returns false

**Description**

Create CADES-T signature for data buffer. Can be used for digital signature of PDF documents in combination with external PDF library

## 1.1.31 CheckMD5 Function

Checks the MD5 hash value of the memory buffer

**C++**

```
BOOL WINAPI CheckMD5(BYTE* source, int sourceSize, BYTE* buffer, int bufferSize);
```

**File**

**File:** Encryption.h (🔗 see page 169)

**Parameters**

Parameters	Description
BYTE* source	The source bytes
int sourceSize	The size of the source buffer
BYTE* buffer	The hash value buffer
int bufferSize	The size of the hash value buffer

**Returns**

Returns TRUE if the hash value is correct, otherwise returns false

**Description**

This function checks if the provided value of the hash is valid

## 1.1.32 CheckSHA1 Function

Checks the SHA1 hash value of the memory buffer

**C++**

```
BOOL WINAPI CheckSHA1(BYTE* source, int sourceSize, BYTE* buffer, int bufferSize);
```

**File**

**File:** Encryption.h (🔗 see page 169)

**Parameters**

Parameters	Description
BYTE* source	The source bytes
int sourceSize	The size of the source buffer
BYTE* buffer	The hash value buffer
int bufferSize	The size of the hash value buffer

**Returns**

Returns TRUE if the hash value is correct, otherwise returns false

**Description**

This function checks if the provided value of the hash is valid

## 1.1.33 CheckSHA256 Function

Checks the SHA256 hash value of the memory buffer

**C++**

```
BOOL WINAPI CheckSHA256(BYTE* source, int sourceSize, BYTE* buffer, int bufferSize);
```

**File**

**File:** Encryption.h (🔗 see page 169)

**Parameters**

Parameters	Description
BYTE* source	The source bytes
int sourceSize	The size of the source buffer
BYTE* buffer	The hash value buffer
int bufferSize	The size of the hash value buffer

**Returns**

Returns TRUE if the hash value is correct, otherwise returns false

**Description**

This function checks if the provided value of the hash is valid

## 1.1.34 ClearFileAttributesA Function

This function sets the file attributes to normal.

**C++**

```
void WINAPI ClearFileAttributesA(LPSTR fileName);
```

**File**

File: FileOperations.h (see page 170)

**Parameters**

Parameters	Description
LPSTR fileName	The name of the file

**Description**

This function removed additional file attributes, like system, read-only and hidden.

## 1.1.35 ClearFileAttributesW Function

This function sets the file attributes to normal.

**C++**

```
void WINAPI ClearFileAttributesW(LPWSTR fileName);
```

**File**

File: FileOperations.h (see page 170)

**Parameters**

Parameters	Description
LPWSTR fileName	The name of the file

**Description**

This function removed additional file attributes, like system, read-only and hidden.

## 1.1.36 ClearUnusedMemory Function

Clears unused memory and minimized the application memory usage

**C++**

```
void WINAPI ClearUnusedMemory();
```

**File**

**File:** System.h (see page 178)

## 1.1.37 CloneFont Function

This is function CloneFont.

**C++**

```
HFONT WINAPI CloneFont(HFONT hFont);
```

**File**

**File:** Graphics.h (see page 171)

## 1.1.38 ContainerCertificate Function

Assign certificate for signing ASIC container

**C++**

```
BOOL WINAPI ContainerCertificate(LPVOID container, LPWSTR fileName, LPWSTR password);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
LPVOID container	Container handle, which is allocated by calling InitializeContainer (see page 77) function

**Returns**

Returns true if the operation is successful, otherwise returns false

**Description**

Use this function to assign certificate which will be used to sign container at the moment when container will be saved to the file

## 1.1.39 ContainerEidCertificate Function

Select EID card certificate to sign ASIC container

**C++**

```
BOOL WINAPI ContainerEidCertificate(LPVOID container, int readerNumber);
```

**File**

**File:** Swelio.h ([↗](#) see page 173)

**Parameters**

Parameters	Description
LPVOID container	Container handle, which is allocated by calling InitializeContainer ( <a href="#">↗</a> see page 77) function

**Returns**

Returns true if the operation is successful, otherwise returns false

**Description**

Call this function to select EID certificate to sign ASIC container

## 1.1.40 ContainerPickCertificate Function

Pick certificate to sign ASIC container

**C++**

```
BOOL WINAPI ContainerPickCertificate(LPVOID container);
```

**File**

**File:** Swelio.h ([↗](#) see page 173)

**Parameters**

Parameters	Description
LPVOID container	Container handle, which is allocated by calling InitializeContainer ( <a href="#">↗</a> see page 77) function

**Returns**

Returns true if the operation is successful, otherwise returns false

**Description**

Use this function to let user pick the certificate to sign ASIC container The dialog to choose the certificate will be shown to the user

## 1.1.41 CopyNativeBitmap Function

This is function CopyNativeBitmap.

**C++**

```
void WINAPI CopyNativeBitmap(HBITMAP src, HDC dstDC, int width, int height, int left, int top);
```

**File**

**File:** Graphics.h ([↗](#) see page 171)

## 1.1.42 CreateCardBuffer Function

Creates XML buffer

**C++**

```
void* WINAPI CreateCardBuffer();
```

**File**

**File:** Swelio.h (see page 173)

**Returns**

The memory buffer to store information

**Description**

Use this function to create XML buffer

## 1.1.43 CreateNativeBitmap Function

This is function CreateNativeBitmap.

**C++**

```
HBITMAP WINAPI CreateNativeBitmap(INT width, INT height, __deref_opt_out void ** ppvBits);
```

**File**

**File:** Graphics.h (see page 171)

## 1.1.44 CreateUnicodeFileA Function

Creates UNICODE file

**C++**

```
void WINAPI CreateUnicodeFileA(LPCSTR fileName);
```

**File**

**File:** FileOperations.h (see page 170)

**Parameters**

Parameters	Description
LPCSTR fileName	The name of the file

## 1.1.45 CreateUnicodeFileW Function

Creates UNICODE file

**C++**

```
void WINAPI CreateUnicodeFileW(LPCWSTR fileName);
```

**File**

**File:** FileOperations.h (🔗 see page 170)

**Parameters**

Parameters	Description
LPCWSTR fileName	The name of the file

## 1.1.46 CreateWindowsFont Function

This is function CreateWindowsFont.

**C++**

```
HFONT WINAPI CreateWindowsFont(LPCWSTR fontFamily, INT size, INT fontStyle, INT fontQuality);
```

**File**

**File:** Graphics.h (🔗 see page 171)

## 1.1.47 CurrentIPAddressA Function

Returns the IP address

**C++**

```
void WINAPI CurrentIPAddressA(LPSTR address, UINT len);
```

**File**

**File:** SystemInfo.h (🔗 see page 179)

## 1.1.48 CurrentIPAddressW Function

Returns the IP address

**C++**

```
void WINAPI CurrentIPAddressW(LPWSTR address, UINT len);
```

**File**

**File:** SystemInfo.h (🔗 see page 179)

## 1.1.49 DeactivateCard Function

Terminates a cennection between a smart card and a reader

**C++**

```
void WINAPI DeactivateCard();
```

**File**

**File:** Swelio.h (🔗 see page 173)



**Description**

The DeactivateCard function terminates a connection previously opened between the calling application and a smart card in the target reader.

## 1.1.50 DeactivateCardEx Function

Terminates a connection between a smart card and a reader

**C++**

```
void WINAPI DeactivateCardEx(int readerNumber);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader

**Description**

The DeactivateCard (see page 28) function terminates a connection previously opened between the calling application and a smart card in the target reader.

## 1.1.51 DeallocateBuffer Function

Deallocates the memory buffer

**C++**

```
void WINAPI DeallocateBuffer(void* buffer);
```

**File**

**File:** FileOperations.h (see page 170)

**Parameters**

Parameters	Description
void* buffer	The pointer to the memory buffer

**Description**

DeallocateBuffer frees a memory block previously allocated with AllocateBuffer (see page 12). Use this procedure to dispose of a memory block obtained with AllocateBuffer (see page 12).

## 1.1.52 DeallocateHWND A Function

This function destroys the specified window.

**C++**

```
BOOL WINAPI DeallocateHWND (HWND hwnd);
```

**File**

**File:** System.h (see page 178)

**Parameters**

Parameters	Description
HWND hwnd	Handle to the window to be destroyed

**Description**

This function restores the window default procedure and destroys the window

## 1.1.53 DeallocateHWNDW Function

This function destroys the specified window.

**C++**

```
BOOL WINAPI DeallocateHWNDW(HWND hwnd);
```

**File**

**File:** System.h (🔗 see page 178)

**Parameters**

Parameters	Description
HWND hwnd	Handle to the window to be destroyed

**Description**

This function restores the window default procedure and destroys the window

## 1.1.54 DecryptFileAESA Function

Decrypts file using AES algorithm.

**C++**

```
BOOL WINAPI DecryptFileAESA(LPSTR szSource, LPSTR szDestination, LPSTR szPassword);
```

**File**

**File:** Encryption.h (🔗 see page 169)

**Parameters**

Parameters	Description
LPSTR szSource	The source file name
LPSTR szDestination	The decrypted file name
LPSTR szPassword	The password

**Returns**

Returns TRUE if the file is successfully decrypted, otherwise returns FALSE.

**Description**

Use this function to decrypt the file using AES algorithm

## 1.1.55 DecryptFileAESW Function

Decrypts file using AES algorithm.

**C++**

```
BOOL WINAPI DecryptFileAESW(LPWSTR szSource, LPWSTR szDestination, LPWSTR szPassword);
```

**File**

**File:** Encryption.h (🔗 see page 169)

**Parameters**

Parameters	Description
LPWSTR szSource	The source file name
LPWSTR szDestination	The decrypted file name
LPWSTR szPassword	The password

**Returns**

Returns TRUE if the file is successfully decrypted, otherwise returns FALSE.

**Description**

Use this function to decrypt the file using AES algorithm

## 1.1.56 DeleteCardBuffer Function

Deletes XML buffer

**C++**

```
void WINAPI DeleteCardBuffer(void* buffer);
```

**File**

**File:** Swelio.h (🔗 see page 173)

**Parameters**

Parameters	Description
void* buffer	The memory buffer to store information

**Description**

Use this function to delete the buffer

## 1.1.57 DeleteToRecycleBinA Function

Deletes file to the Windows Recycle Bin

**C++**

```
void WINAPI DeleteToRecycleBinA(LPSTR fileName, BOOL silent);
```

**File**

**File:** FileOperations.h (🔗 see page 170)

**Parameters**

Parameters	Description
LPSTR fileName	The name of the file
BOOL silent	Do not display a progress dialog box

**Description**

Use this function to delete the file to Windows Recycle Bin

## 1.1.58 DeleteToRecycleBinW Function

Deletes file to WIndows Recycle Bin

**C++**

```
void WINAPI DeleteToRecycleBinW(LPWSTR fileName, BOOL silent);
```

**File**

File: FileOperations.h (see page 170)

**Parameters**

Parameters	Description
LPWSTR fileName	The name of the file
BOOL silent	Do not display a progress dialog box

**Description**

Use this function to delete the file to Windows Recycle Bin

## 1.1.59 DestroyFont Function

This is function DestroyFont.

**C++**

```
void WINAPI DestroyFont(HFONT hFont);
```

**File**

File: Graphics.h (see page 171)

## 1.1.60 DestroyImageBuffer Function

Destroys the memory buffer

**C++**

```
void WINAPI DestroyImageBuffer(void* buffer);
```

**File**

File: quicol.h (see page 179)

**Parameters**

Parameters	Description
void* buffer	The memory buffer

**Description**

Destroys the memory buffer created to hold the image returned by GetPNGA (see page 69) (Ansi) or GetPNGW (see page 70) (Unicode) functions

## 1.1.61 DirectoryExistsA Function

Determines whether a specified directory exists.

**C++**

```
BOOL WINAPI DirectoryExistsA(LPSTR fileName);
```

**File**

**File:** FileOperations.h (see page 170)

**Parameters**

Parameters	Description
LPSTR fileName	The name of the directory

**Returns**

Returns TRUE if the directory exists, otherwise returns FALSE

**Description**

Call DirectoryExists to determine whether the directory specified by the Directory parameter exists. If the directory exists, the function returns True. If the directory does not exist, the function returns False. If a full path name is entered, DirectoryExists searches for the directory along the designated path. Otherwise, the Directory parameter is interpreted as a relative path name from the current directory.

## 1.1.62 DirectoryExistsW Function

Determines whether a specified directory exists.

**C++**

```
BOOL WINAPI DirectoryExistsW(LPWSTR fileName);
```

**File**

**File:** FileOperations.h (see page 170)

**Parameters**

Parameters	Description
LPWSTR fileName	The name of the directory

**Returns**

Returns TRUE if the directory exists, otherwise returns FALSE

**Description**

Call DirectoryExists to determine whether the directory specified by the Directory parameter exists. If the directory exists, the function returns True. If the directory does not exist, the function returns False. If a full path name is entered, DirectoryExists searches for the directory along the designated path. Otherwise, the Directory parameter is interpreted as a relative path name from the current directory.

# 1.1.63 DisplayCertificate Function

Displays the dialog window with certificate information

C++

```
void WINAPI DisplayCertificate(PEidCertificate certificate);
```

File

File: Swelio.h (🔗 see page 173)

Parameters

Parameters	Description
PEidCertificate certificate	The certificate data

Description

Use this function to show the certificate for the user

# 1.1.64 DpiY Function

This is function DpiY.

C++

```
int WINAPI DpiY();
```

File

File: Graphics.h (🔗 see page 171)

# 1.1.65 DrawAlphaText Function

This is function DrawAlphaText.

C++

```
void WINAPI DrawAlphaText(HDC hDC, LPCWSTR s, HFONT hFont, COLORREF color, int width, int height, UINT flags);
```

File

File: Graphics.h (🔗 see page 171)

# 1.1.66 DrawAlphaTextRect Function

This is function DrawAlphaTextRect.

C++

```
void WINAPI DrawAlphaTextRect(HDC hDC, LPCWSTR s, HFONT hFont, COLORREF color, LPRECT lpRect, UINT flags);
```

File

File: Graphics.h (🔗 see page 171)

## 1.1.67 DrawLayeredWindow Function

Repaints the surface of the layered window

**C++**

```
void WINAPI DrawLayeredWindow(HWND handle, int left, int top, int width, int height, HDC buffer, COLORREF colorKey, byte alpha, BOOL redrawOnly);
```

**File**

**File:** System.h (🔗 see page 178)

**Parameters**

Parameters	Description
HWND handle	The handle of the window
int left	The horizontal coordinate of the window
int top	The vertical coordinate of the window
int width	The width of the window
int height	The height of the window
HDC buffer	Handle to a DC for the surface that defines the layered window
COLORREF colorKey	COLORREF structure that specifies the color key to be used when composing the layered window.
byte alpha	Specifies an alpha transparency value to be used on the entire source bitmap
BOOL redrawOnly	Only redraw and do not update the window position

## 1.1.68 DrawNativeBitmap Function

This is function DrawNativeBitmap.

**C++**

```
void WINAPI DrawNativeBitmap(HBITMAP src, HBITMAP dst, int width, int height);
```

**File**

**File:** Graphics.h (🔗 see page 171)

## 1.1.69 DrawTextDirect Function

This is function DrawTextDirect.

**C++**

```
void WINAPI DrawTextDirect(HDC hDC, LPCWSTR s, HFONT hFont, COLORREF color, int left, int top);
```

**File**

**File:** Graphics.h (🔗 see page 171)

## 1.1.70 DrawTextDirectEx Function

This is function DrawTextDirectEx.

**C++**

```
void WINAPI DrawTextDirectEx(HDC hDC, LPCWSTR s, HFONT hFont, COLORREF color, COLORREF background, int left, int top);
```

**File**

**File:** Graphics.h (🔗 see page 171)

## 1.1.71 DrawTextGlow Function

This is function DrawTextGlow.

**C++**

```
void WINAPI DrawTextGlow(HDC hDC, LPCWSTR s, HFONT hFont, COLORREF foreColor, int left, int top);
```

**File**

**File:** Graphics.h (🔗 see page 171)

## 1.1.72 DrawTextLine Function

This is function DrawTextLine.

**C++**

```
void WINAPI DrawTextLine(HDC hDC, LPCWSTR s, HFONT hFont, COLORREF color, int left, int top);
```

**File**

**File:** Graphics.h (🔗 see page 171)

## 1.1.73 DrawTextOutline Function

This is function DrawTextOutline.

**C++**

```
void WINAPI DrawTextOutline(HDC hDC, LPCWSTR s, HFONT hFont, COLORREF foreColor, COLORREF backColor, int left, int top);
```

**File**

**File:** Graphics.h (🔗 see page 171)



# 1.1.74 DrawTextRect Function

This is function DrawTextRect.

C++

```
void WINAPI DrawTextRect(HDC hDC, LPCWSTR s, HFONT hFont, COLORREF color, COLORREF background, LPRECT lpRect, UINT flags);
```

File

File: Graphics.h (🔗 see page 171)

# 1.1.75 EmToPixels Function

This is function EmToPixels.

C++

```
int WINAPI EmToPixels(int em);
```

File

File: Graphics.h (🔗 see page 171)

# 1.1.76 EmptyRecycleBin Function

Empties the recycle bin

C++

```
void WINAPI EmptyRecycleBin();
```

File

File: System.h (🔗 see page 178)

Description

Removes all files from the Windows recycle bin

# 1.1.77 EncodeCertificate Function

Performs Base64 encoding of the certificate

C++

```
int WINAPI EncodeCertificate(PEidCertificate certificate, BYTE* buffer, int bufferSize);
```

File

File: Swelio.h (🔗 see page 173)

Parameters

Parameters	Description
PEidCertificate certificate	The certificate data

BYTE* buffer	The Base64 encoded certificate buffer
int bufferSize	The size of the buffer

**Returns**

Returns the size of the buffer needed to hold the encoded certificate

## 1.1.78 EncodePhoto Function

Performs Base64 encoding of the photo

**C++**

```
int WINAPI EncodePhoto(PeidPicture photo, BYTE* buffer, int bufferSize);
```

**File**

File: Swelio.h (see page 173)

**Parameters**

Parameters	Description
PeidPicture photo	The pointer to EidPicture (see page 156) structure
BYTE* buffer	The Base64 encoded photo buffer
int bufferSize	The size of the buffer

**Returns**

Returns the size of the buffer needed to hold the encoded photo

**Description**

Use this function for Base64 encoding of the photo

## 1.1.79 EncryptFileAESA Function

Encrypts file using AES algorithm.

**C++**

```
BOOL WINAPI EncryptFileAESA(LPSTR szSource, LPSTR szDestination, LPSTR szPassword);
```

**File**

File: Encryption.h (see page 169)

**Parameters**

Parameters	Description
LPSTR szSource	The source file name
LPSTR szDestination	The encrypted file name
LPSTR szPassword	The password

**Returns**

Returns TRUE if the file is successfully encrypted, otherwise returns FALSE

**Description**

Use this function to encrypt the file using AES algorithm

## 1.1.80 EncryptFileAESW Function

Encrypts file using AES algorithm.

### C++

```
BOOL WINAPI EncryptFileAESW(LPWSTR szSource, LPWSTR szDestination, LPWSTR szPassword);
```

### File

**File:** Encryption.h (🔗 see page 169)

### Parameters

Parameters	Description
LPWSTR szSource	The source file name
LPWSTR szDestination	The encrypted file name
LPWSTR szPassword	The password

### Returns

Returns TRUE if the file is successfully encrypted, otherwise returns FALSE

### Description

Use this function to encrypt the file using AES algorithm

## 1.1.81 FileCloseA Function

Concludes input/output (I/O) to a file opened using the FileCreateRewrite function.

### C++

```
void WINAPI FileCloseA(HANDLE handle);
```

### File

**File:** FileOperations.h (🔗 see page 170)

### Parameters

Parameters	Description
HANDLE handle	The handle of the file

### Description

Closes the file handle of the specified file when its not in use anymore

## 1.1.82 FileCloseW Function

Concludes input/output (I/O) to a file opened using the FileCreateRewrite function.

### C++

```
void WINAPI FileCloseW(HANDLE handle);
```

### File

**File:** FileOperations.h (🔗 see page 170)

**Parameters**

Parameters	Description
HANDLE handle	The handle of the file

**Description**

Closes the file handle of the specified file when its not in use anymore

## 1.1.83 FileCopyA Function

The CopyFile function copies an existing file to a new file.

**C++**

```
BOOL WINAPI FileCopyA(LPSTR oldName, LPSTR newName);
```

**File**

**File:** FileOperations.h (see page 170)

**Parameters**

Parameters	Description
LPSTR oldName	The name of the source file
LPSTR newName	The name of the destination file

**Returns**

The result of the function is TRUE when the file is successfully copied to the new location, otherwise the result is FALSE.

**Description**

This function makes a copy of the file with the new name or path.

## 1.1.84 FileCopyW Function

The CopyFile function copies an existing file to a new file.

**C++**

```
BOOL WINAPI FileCopyW(LPWSTR oldName, LPWSTR newName);
```

**File**

**File:** FileOperations.h (see page 170)

**Parameters**

Parameters	Description
LPWSTR oldName	The name of the source file
LPWSTR newName	The name of the destination file

**Returns**

The result of the function is TRUE when the file is successfully copied to the new location, otherwise the result is FALSE.

**Description**

This function makes a copy of the file with the new name or path.

## 1.1.85 FileCreateRewriteA Function

Creates new or overwrites existing file

### C++

```
HANDLE WINAPI FileCreateRewriteA(LPCSTR fileName);
```

### File

**File:** FileOperations.h (see page 170)

### Returns

The result of the function is the handle of the file

### Description

This function creates the new file with provided file name if the file with given name does not exists. If the file exists, it will be overwritten and the current content of the file will be lost

## 1.1.86 FileCreateRewriteW Function

Creates new or overwrites existing file

### C++

```
HANDLE WINAPI FileCreateRewriteW(LPCWSTR fileName);
```

### File

**File:** FileOperations.h (see page 170)

### Returns

The result of the function is the handle of the file

### Description

This function creates the new file with provided file name if the file with given name does not exists. If the file exists, it will be overwritten and the current content of the file will be lost

## 1.1.87 FileDeleteA Function

Deletes a file from disk.

### C++

```
void WINAPI FileDeleteA(LPSTR fileName);
```

### File

**File:** FileOperations.h (see page 170)

### Parameters

Parameters	Description
LPSTR fileName	The name of the file

### Description

DeleteFile deletes the file named by fileName from the disk.

# 1.1.88 FileDeleteW Function

Deletes a file from disk.

C++

```
void WINAPI FileDeleteW(LPWSTR fileName);
```

File

File: FileOperations.h (🔗 see page 170)

Parameters

Parameters	Description
LPWSTR fileName	The name of the file

Description

DeleteFile deletes the file named by fileName from the disk.

# 1.1.89 FileExistsA Function

Tests whether a specified file exists.

C++

```
BOOL WINAPI FileExistsA(LPSTR fileName);
```

File

File: FileOperations.h (🔗 see page 170)

Returns

FileExists returns TRUE if the file specified by FileName exists. If the file does not exist, FileExists returns FALSE.

Description

Use this function to check if the file with provided name exists.

# 1.1.90 FileExistsW Function

Tests whether a specified file exists.

C++

```
BOOL WINAPI FileExistsW(LPWSTR fileName);
```

File

File: FileOperations.h (🔗 see page 170)

Returns

FileExists returns TRUE if the file specified by FileName exists. If the file does not exist, FileExists returns FALSE.

Description

Use this function to check if the file with provided name exists.

## 1.1.91 FileExtensionIsA Function

Checks the file extension

### C++

```
BOOL WINAPI FileExtensionIsA(LPCSTR fileName, LPCSTR ext);
```

### File

**File:** FileOperations.h (see page 170)

### Parameters

Parameters	Description
LPCSTR fileName	The name of the file
LPCSTR ext	The file name extension

### Returns

Returns true if the file has a specified extension, otherwise returns false.

### Description

This function checks if the file has a given extension

## 1.1.92 FileExtensionIsW Function

Checks the file extension

### C++

```
BOOL WINAPI FileExtensionIsW(LPCWSTR fileName, LPCWSTR ext);
```

### File

**File:** FileOperations.h (see page 170)

### Parameters

Parameters	Description
LPCWSTR fileName	The name of the file
LPCWSTR ext	The file name extension

### Returns

Returns true if the file has a specified extension, otherwise returns false.

### Description

This function checks if the file has a given extension

## 1.1.93 FileGetSizeA Function

Retrieves the size of a specified file.

### C++

```
DWORD WINAPI FileGetSizeA(LPCSTR fileName);
```

**File**

**File:** FileOperations.h (🔗 see page 170)

**Parameters**

Parameters	Description
LPCSTR fileName	The name of the file

**Returns**

The size of the file in bytes.

**Description**

This function determines the size of the file specified by the file name.

## 1.1.94 FileGetSizeW Function

Retrieves the size of a specified file.

**C++**

```
DWORD WINAPI FileGetSizeW(LPCWSTR fileName);
```

**File**

**File:** FileOperations.h (🔗 see page 170)

**Parameters**

Parameters	Description
LPCWSTR fileName	The name of the file

**Returns**

The size of the file in bytes.

**Description**

This function determines the size of the file specified by the name of the file

## 1.1.95 FileIsExeA Function

Checks if the file is a Windows executable

**C++**

```
BOOL WINAPI FileIsExeA(LPSTR fileName);
```

**File**

**File:** FileOperations.h (🔗 see page 170)

**Parameters**

Parameters	Description
LPSTR fileName	The name of the file

**Returns**

Returns TRUE if the file is a Windows executable, otherwise returns FALSE.



## 1.1.96 FileIsExeW Function

Checks if the file is a Windows executable

**C++**

```
BOOL WINAPI FileIsExeW(LPWSTR fileName);
```

**File**

**File:** FileOperations.h (see page 170)

**Parameters**

Parameters	Description
LPWSTR fileName	The name of the file

**Returns**

Returns TRUE if the file is a Windows executable, otherwise returns FALSE.

## 1.1.97 FileIsIconA Function

Checks if the file is a Windows icon (.ico) file

**C++**

```
BOOL WINAPI FileIsIconA(LPSTR fileName);
```

**File**

**File:** FileOperations.h (see page 170)

**Parameters**

Parameters	Description
LPSTR fileName	The name of the file

**Returns**

Returns TRUE if the file is a Windows icon (.ico) file, otherwise returns FALSE.

## 1.1.98 FileIsIconW Function

Checks if the file is a Windows icon (.ico) file

**C++**

```
BOOL WINAPI FileIsIconW(LPWSTR fileName);
```

**File**

**File:** FileOperations.h (see page 170)

**Parameters**

Parameters	Description
LPWSTR fileName	The name of the file

**Returns**

Returns TRUE if the file is a Windows icon (.ico) file, otherwise returns FALSE.

## 1.1.99 FileIsImageA Function

Checks if the file is an image file

**C++**

```
BOOL WINAPI FileIsImageA(LPCSTR fileName);
```

**File**

**File:** FileOperations.h (🔗 see page 170)

**Parameters**

Parameters	Description
LPCSTR fileName	The name of the file

**Returns**

Returns TRUE if the file is an image file, otherwise returns FALSE.

## 1.1.100 FileIsImageW Function

Checks if the file is an image file

**C++**

```
BOOL WINAPI FileIsImageW(LPCWSTR fileName);
```

**File**

**File:** FileOperations.h (🔗 see page 170)

**Parameters**

Parameters	Description
LPCWSTR fileName	The name of the file

**Returns**

Returns TRUE if the file is an image file, otherwise returns FALSE.

## 1.1.101 FileIsLink Function

Checks to see if the file specified by file name is a Microsoft Windows shortcut (.Lnk) file (and is neither a file nor a folder).

**C++**

```
BOOL WINAPI FileIsLink(LPCWSTR fileName);
```

**File**

**File:** FileOperations.h (🔗 see page 170)

**Parameters**

Parameters	Description
LPCWSTR fileName	The name of the file to check

**Returns**

Returns TRUE if the file is a Microsoft Windows shortcut, otherwise returns FALSE

**Description**

This function is used to check if the file with provided file name is a Windows shortcut or not

## 1.1.102 FileOrFolderExistsA Function

Checks if the file or folder with the given name exists

**C++**

```
BOOL WINAPI FileOrFolderExistsA(LPSTR fileName);
```

**File**

**File:** FileOperations.h (🔗 see page 170)

**Parameters**

Parameters	Description
LPSTR fileName	The file or folder name

**Returns**

Returns TRUE if the file or folder exists, otherwise returns FALSE.

## 1.1.103 FileOrFolderExistsW Function

Checks if the file or folder with the given name exists

**C++**

```
BOOL WINAPI FileOrFolderExistsW(LPWSTR fileName);
```

**File**

**File:** FileOperations.h (🔗 see page 170)

**Parameters**

Parameters	Description
LPWSTR fileName	The file or folder name

**Returns**

Returns TRUE if the file or folder exists, otherwise returns FALSE.

## 1.1.104 FileRenameA Function

Renames the file

**C++**

```
BOOL WINAPI FileRenameA(LPSTR oldName, LPSTR newName);
```

**File**

**File:** FileOperations.h (🔗 see page 170)

**Parameters**

Parameters	Description
LPSTR oldName	File to be renamed
LPSTR newName	New name of the file

**Returns**

Returns TRUE if the file was successfully renamed, otherwise returns FALSE

## 1.1.105 FileRenameW Function

Renames the file

**C++**

```
BOOL WINAPI FileRenameW(LPWSTR oldName, LPWSTR newName);
```

**File**

File: FileOperations.h (see page 170)

**Parameters**

Parameters	Description
LPWSTR oldName	File to be renamed
LPWSTR newName	New name of the file

**Returns**

Returns TRUE if the file was successfully renamed, otherwise returns FALSE

## 1.1.106 FileWriteA Function

Writes string to the file

**C++**

```
void WINAPI FileWriteA(HANDLE handle, LPSTR text);
```

**File**

File: FileOperations.h (see page 170)

**Parameters**

Parameters	Description
HANDLE handle	The handle of the file
LPSTR text	The text to write

## 1.1.107 FileWriteCharA Function

Writes one character to the file

**C++**

```
void WINAPI FileWriteCharA(HANDLE handle, CHAR text);
```

**File**

**File:** FileOperations.h (🔗 see page 170)

**Parameters**

Parameters	Description
HANDLE handle	The handle of the file
CHAR text	The character to write

## 1.1.108 FileWriteCharW Function

Writes one character to the file

**C++**

```
void WINAPI FileWriteCharW(HANDLE handle, WCHAR text);
```

**File**

**File:** FileOperations.h (🔗 see page 170)

**Parameters**

Parameters	Description
HANDLE handle	The handle of the file
WCHAR text	The character to write

## 1.1.109 FileWriteNewLineA Function

Writes new line sequence to the file

**C++**

```
void WINAPI FileWriteNewLineA(HANDLE handle);
```

**File**

**File:** FileOperations.h (🔗 see page 170)

**Parameters**

Parameters	Description
HANDLE handle	The handle of the file

## 1.1.110 FileWriteNewLineW Function

Writes new line sequence to the file

**C++**

```
void WINAPI FileWriteNewLineW(HANDLE handle);
```

**File**

**File:** FileOperations.h (🔗 see page 170)

**Parameters**

Parameters	Description
HANDLE handle	The handle of the file

## 1.1.111 FileWriteW Function

Writes string to the file

**C++**

```
void WINAPI FileWriteW(HANDLE handle, LPWSTR text);
```

**File**

**File:** FileOperations.h (see page 170)

**Parameters**

Parameters	Description
HANDLE handle	The handle of the file
LPWSTR text	The text to write

## 1.1.112 FreeContainer Function

Deallocates ASIC container

**C++**

```
void WINAPI FreeContainer(LPVOID container);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
LPVOID container	Container handle, which is allocated by calling InitializeContainer (see page 77) function

**Returns**

None

**Description**

Call this function to deallocate container memory and release container handle.

## 1.1.113 FullPathA Function

Gets the full path to the file based on file name

**C++**

```
BOOL WINAPI FullPathA(LPSTR fileName, LPSTR fullName);
```

**File**

**File:** FileOperations.h (see page 170)

**Parameters**

Parameters	Description
LPSTR fileName	The name of the file
LPSTR fullName	The full path to the file

## 1.1.114 FullPathW Function

Gets the full path to the file based on file name

**C++**

```
BOOL WINAPI FullPathW(LPWSTR fileName, LPWSTR fullName);
```

**File**

**File:** FileOperations.h (see page 170)

**Parameters**

Parameters	Description
LPWSTR fileName	The name of the file
LPWSTR fullName	The full path to the file

## 1.1.115 GenerateAuthenticationSignatureA Function

Generate authentication signature

**C++**

```
BOOL WINAPI GenerateAuthenticationSignatureA(LPSTR pinCode, BYTE* dataHash, int hashSize, BYTE* signature, LPDWORD signatureSize);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
LPSTR pinCode	The card PIN code value
BYTE* dataHash	The hash value buffer
int hashSize	The size of the hash data buffer
BYTE* signature	The output buffer that contains the generated signature
LPDWORD signatureSize	The size of the signature buffer

**Returns**

Returns TRUE if the signature is successfully generated, otherwise returns FALSE

**Description**

Generate authentication signature using provided hash value

## 1.1.116 GenerateAuthenticationSignatureExA Function

Generate authentication signature

**C++**

```
BOOL WINAPI GenerateAuthenticationSignatureExA(int readerNumber, LPSTR pinCode, BYTE* dataHash, int hashSize, BYTE* signature, LPDWORD signatureSize);
```

**File**

File: Swelio.h (see page 173)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader.
LPSTR pinCode	The card PIN code value
BYTE* dataHash	The hash value buffer
int hashSize	The size of the hash data buffer
BYTE* signature	The output buffer that contains the generated signature
LPDWORD signatureSize	The size of the signature buffer

**Returns**

Returns TRUE if the signature is successfully generated, otherwise returns FALSE

**Description**

Generate authentication signature using provided hash value

## 1.1.117 GenerateAuthenticationSignatureExW Function

Generate authentication signature

**C++**

```
BOOL WINAPI GenerateAuthenticationSignatureExW(int readerNumber, LPWSTR pinCode, BYTE* dataHash, int hashSize, BYTE* signature, LPDWORD signatureSize);
```

**File**

File: Swelio.h (see page 173)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader.
LPWSTR pinCode	The card PIN code value
BYTE* dataHash	The hash value buffer
int hashSize	The size of the hash data buffer
BYTE* signature	The output buffer that contains the generated signature
LPDWORD signatureSize	The size of the signature buffer

**Returns**

Returns TRUE if the signature is successfully generated, otherwise returns FALSE

**Description**

Generate authentication signature using provided hash value



## 1.1.118 GenerateAuthenticationSignatureW Function

Generate authentication signature

### C++

```
BOOL WINAPI GenerateAuthenticationSignatureW(LPWSTR pinCode, BYTE* dataHash, int hashSize,
BYTE* signature, LPDWORD signatureSize);
```

### File

File: Swelio.h (see page 173)

### Parameters

Parameters	Description
LPWSTR pinCode	The card PIN code value
BYTE* dataHash	The hash value buffer
int hashSize	The size of the hash data buffer
BYTE* signature	The output buffer that contains the generated signature
LPDWORD signatureSize	The size of the signature buffer

### Returns

Returns TRUE if the signature is successfully generated, otherwise returns FALSE

### Description

Generate authentication signature using provided hash value

## 1.1.119 GenerateBMPA Function

Generates Windows Bitmap file with QR Code image

### C++

```
void WINAPI GenerateBMPA(LPSTR fileName, LPSTR text, int margin, int size, int level);
```

### File

File: quicol.h (see page 179)

### Parameters

Parameters	Description
LPSTR fileName	The name of the file
LPSTR text	The text to encode
int margin	The margin from the border in points
int size	The size of the one point in pixels
int level	The error correction level

### Description

Generate Windows Bitmap file with encoded text as QR Code image

## 1.1.120 GenerateBMPW Function

Generates Windows Bitmap file with QR Code image

**C++**

```
void WINAPI GenerateBMPW(LPWSTR fileName, LPWSTR text, int margin, int size, int level);
```

**File**

**File:** quicol.h (see page 179)

**Parameters**

Parameters	Description
LPWSTR fileName	The name of the file
LPWSTR text	The text to encode
int margin	The margin from the border in points
int size	The size of the one point in pixels
int level	The error correction level

**Description**

Generate Windows Bitmap file with encoded text as QR Code image

## 1.1.121 GenerateNonRepudiationSignatureA Function

Generate non repudiation signature

**C++**

```
BOOL WINAPI GenerateNonRepudiationSignatureA(LPSTR pinCode, BYTE* dataHash, int hashSize, BYTE* signature, LPDWORD signatureSize);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
LPSTR pinCode	The card PIN code value
BYTE* dataHash	The hash value buffer
int hashSize	The size of the hash data buffer
BYTE* signature	The output buffer that contains the generated signature
LPDWORD signatureSize	The size of the signature buffer

**Returns**

Returns TRUE if the signature is successfully generated, otherwise returns FALSE

**Description**

Generate non repudiation signature using provided hash value

## 1.1.122 GenerateNonRepudiationSignatureExA Function

Generate non repudiation signature

**C++**

```
BOOL WINAPI GenerateNonRepudiationSignatureExA(int readerNumber, LPSTR pinCode, BYTE* dataHash, int hashSize, BYTE* signature, LPDWORD signatureSize);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader.
LPSTR pinCode	The card PIN code value
BYTE* dataHash	The hash value buffer
int hashSize	The size of the hash data buffer
BYTE* signature	The output buffer that contains the generated signature
LPDWORD signatureSize	The size of the signature buffer

**Returns**

Returns TRUE if the signature is successfully generated, otherwise returns FALSE

**Description**

Generate non repudiation signature using provided hash value

## 1.1.123 GenerateNonRepudiationSignatureExW Function

Generate non repudiation signature

**C++**

```
BOOL WINAPI GenerateNonRepudiationSignatureExW(int readerNumber, LPWSTR pinCode, BYTE* dataHash, int hashSize, BYTE* signature, LPDWORD signatureSize);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader.
LPWSTR pinCode	The card PIN code value
BYTE* dataHash	The hash value buffer
int hashSize	The size of the hash data buffer
BYTE* signature	The output buffer that contains the generated signature
LPDWORD signatureSize	The size of the signature buffer

**Returns**

Returns TRUE if the signature is successfully generated, otherwise returns FALSE

**Description**

Generate non repudiation signature using provided hash value

## 1.1.124 GenerateNonRepudiationSignatureW Function

Generate non repudiation signature

### C++

```
BOOL WINAPI GenerateNonRepudiationSignatureW(LPWSTR pinCode, BYTE* dataHash, int hashSize,  
BYTE* signature, LPDWORD signatureSize);
```

### File

File: Swelio.h (see page 173)

### Parameters

Parameters	Description
LPWSTR pinCode	The card PIN code value
BYTE* dataHash	The hash value buffer
int hashSize	The size of the hash data buffer
BYTE* signature	The output buffer that contains the generated signature
LPDWORD signatureSize	The size of the signature buffer

### Returns

Returns TRUE if the signature is successfully generated, otherwise returns FALSE

### Description

Generate non repudiation signature using provided hash value

## 1.1.125 GeneratePNGA Function

Generates PNG file with QR Code image

### C++

```
void WINAPI GeneratePNGA(LPSTR fileName, LPSTR text, int margin, int size, int level);
```

### File

File: quicol.h (see page 179)

### Parameters

Parameters	Description
LPSTR fileName	The name of the file
LPSTR text	The text to encode
int margin	The margin from the border in points
int size	The size of the one point in pixels
int level	The error correction level

### Description

Generates PNG file with encoded text as QR Code image

## 1.1.126 GeneratePNGW Function

Generates PNG file with QR Code image

**C++**

```
void WINAPI GeneratePNGW(LPWSTR fileName, LPWSTR text, int margin, int size, int level);
```

**File**

**File:** quicol.h (see page 179)

**Parameters**

Parameters	Description
LPWSTR fileName	The name of the file
LPWSTR text	The text to encode
int margin	The margin from the border in points
int size	The size of the one point in pixels
int level	The error correction level

**Description**

Generates PNG file with encoded text as QR Code image

## 1.1.127 GenerateQRCodeA Function

Read eID card and save the identity information and address to PNG QR Code file

**C++**

```
BOOL WINAPI GenerateQRCodeA(LPSTR fileName);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
LPSTR fileName	File name to store information

**Returns**

Returns TRUE if the information is retrieved from the card, otherwise returns FALSE

**Description**

Use this function to read the information about the owner of the card and generate the QR Code PNG image

## 1.1.128 GenerateQRCodeExA Function

Read eID card and save the identity information and address to PNG QR Code file

**C++**

```
BOOL WINAPI GenerateQRCodeExA(int readerNumber, LPSTR fileName);
```

**File**

**File:** Swelio.h (🔗 see page 173)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader.
LPSTR fileName	File name to store information

**Returns**

Returns TRUE if the information is retrieved from the card, otherwise returns FALSE

**Description**

Use this function to read the information about the owner of the card and generate the QR Code PNG image

## 1.1.129 GenerateQRCodeExW Function

Read eID card and save the identity information and address to PNG QR Code file

**C++**

```
BOOL WINAPI GenerateQRCodeExW(int readerNumber, LPWSTR fileName);
```

**File**

**File:** Swelio.h (🔗 see page 173)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader.
LPWSTR fileName	File name to store information

**Returns**

Returns TRUE if the information is retrieved from the card, otherwise returns FALSE

**Description**

Use this function to read the information about the owner of the card and generate the QR Code PNG image

## 1.1.130 GenerateQRCodeW Function

Read eID card and save the identity information and address to PNG QR Code file

**C++**

```
BOOL WINAPI GenerateQRCodeW(LPWSTR fileName);
```

**File**

**File:** Swelio.h (🔗 see page 173)

**Parameters**

Parameters	Description
LPWSTR fileName	File name to store information

**Returns**

Returns TRUE if the information is retrieved from the card, otherwise returns FALSE

**Description**

Use this function to read the information about the owner of the card and generate the QR Code PNG image

## 1.1.131 GetAllFiles Function

Returns the names of files in a specified directory.

**C++**

```
void WINAPI GetAllFiles(FOLDERENUMPROC lpEnumProc, LPWSTR folderName, LPWSTR searchMask, LPARAM lParam);
```

**File**

**File:** FileOperations.h (see page 170)

**Parameters**

Parameters	Description
FOLDERENUMPROC lpEnumProc	Callback function
LPWSTR folderName	The name of the folder
LPWSTR searchMask	The search string to match against the names of files in path.
LPARAM lParam	Specifies an application-defined value to be passed to the callback function

**Description**

This function enumerates all files in the specified folder which names match the searchMask parameter and calls the callback function passing the name of the file to it.

## 1.1.132 GetCardBufferA Function

Gets XML or CSV information from the memory buffer

**C++**

```
void WINAPI GetCardBufferA(void* buffer, void* strDest, int count);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
void* buffer	Memory buffer with information
void* strDest	Destination buffer
int count	Destination buffer size

**Returns**

None

**Description**

Use this function to get the card information in CSV or XML format from the memory buffer

## 1.1.133 GetCardBufferSize Function

This function returns the size of the buffer needed to hold the information from the eID card in the XML or CSV format

### C++

```
int WINAPI GetCardBufferSize(void* buffer);
```

### File

File: Swelio.h (see page 173)

### Returns

The size of the XML or CSV buffer

### Description

Use this function to get the size of the XML buffer before allocating the buffer in memory

## 1.1.134 GetCardBufferW Function

Gets XML or CSV information from the memory buffer

### C++

```
void WINAPI GetCardBufferW(void* buffer, void* strDest, int count);
```

### File

File: Swelio.h (see page 173)

### Parameters

Parameters	Description
void* buffer	Memory buffer with information
void* strDest	Destination buffer
int count	Destination buffer size

### Returns

None

### Description

Use this function to get the card information in CSV or XML format from the memory buffer

## 1.1.135 GetCardSerialNumber Function

Get the serial number of EID card

### C++

```
BOOL WINAPI GetCardSerialNumber(int readerNumber, BYTE* serialNumber, LPDWORD serialNumberSize);
```

### File

File: Swelio.h (see page 173)



**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader.
BYTE* serialNumber	The memory buffer for getting the card serial number
LPDWORD serialNumberSize	the size of the memory buffer in bytes

**Returns**

Returns TRUE if the information is retrieved from the card, otherwise returns FALSE

**Description**

Use this function to get the serial number of EID card into the memory buffer

## 1.1.136 GetCardVersion Function

Get the applet version number for card in the reader with specified number

**C++**

```
int WINAPI GetCardVersion(int readerNumber);
```

**File**

File: Swelio.h (see page 173)

**Parameters**

Parameters	Description
int readerNumber	The reader index, starting from 0

**Returns**

The card applet version number

**Description**

Get the version number of eid card applet using the zero-based reader index. The first reader has number 0, second - 1, etc... You can read the information only from one selected reader at once.

## 1.1.137 GetContainerError Function

Gets the value of the container validation error message based on the message index

**C++**

```
BOOL WINAPI GetContainerError(LPVOID container, int index, LPWSTR buffer, int bufferSize);
```

**File**

File: Swelio.h (see page 173)

**Parameters**

Parameters	Description
LPVOID container	Container handle, which is allocated by calling InitializeContainer (see page 77) function
int index	error zero-based index
LPWSTR buffer	preallocated buffer for getting error message text
bufSize	the size of buffer in characters

Returns

Returns true if the operation is successful, otherwise returns false

Description

Call this function to retrieve the validation error message

1.1.138 GetContainerErrorsCount Function

Get the number of error messages availavle in the container validation report

C++

```
int WINAPI GetContainerErrorsCount(LPVOID container);
```

File

File: Swelio.h (see page 173)

Parameters

Parameters	Description
LPVOID container	Container handle, which is allocated by calling InitializeContainer (see page 77) function

Returns

Returns the number of the container validation errors

Description

Call this function to get the number of the container validation errors

1.1.139 GetEncodedCertificateSize Function

Returns the size of the Base64 encoded certificate

C++

```
int WINAPI GetEncodedCertificateSize(PCertCertificate certificate);
```

File

File: Swelio.h (see page 173)

Parameters

Parameters	Description
PCertCertificate certificate	The certificate data

Returns

Returns the size of the buffer needed to hold the encoded certificate

Description

Use this function to calculate the size of the buffer needed to encode the certificate

## 1.1.140 GetEncodedPhotoSize Function

Calculates buffer size for Base64 encoded photo

**C++**

```
int WINAPI GetEncodedPhotoSize(PeidPicture photo);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
PeidPicture photo	The pointer to EidPicture (see page 156) structure

**Returns**

The desired size of the buffer

**Description**

Use this function to calculate the size of the buffer needed for Base64 encoding of the photo This can be useful for including the photo data to the text document, for example to XML file

## 1.1.141 GetFileMD5A Function

Gets the MD5 hash value for the file

**C++**

```
BOOL WINAPI GetFileMD5A(LPSTR fileName, BYTE* buffer, int bufferSize);
```

**File**

**File:** Encryption.h (see page 169)

**Parameters**

Parameters	Description
LPSTR fileName	The name of the file
BYTE* buffer	The buffer to store the hash value
int bufferSize	The size of the buffer

**Returns**

The result of the function is equal to TRUE if operation is completed successfully, otherwise the result is FALSE

**Description**

Calculates MD5 hash value for the given file

## 1.1.142 GetFileMD5W Function

Gets the MD5 hash value for the file

**C++**

```
BOOL WINAPI GetFileMD5W(LPWSTR fileName, BYTE* buffer, int bufferSize);
```

**File**

**File:** Encryption.h (🔗 see page 169)

**Parameters**

Parameters	Description
LPWSTR fileName	The name of the file
BYTE* buffer	The buffer to store the hash value
int bufferSize	The size of the buffer

**Returns**

The result of the function is equal to TRUE if operation is completed successfully, otherwise the result is FALSE

**Description**

Calculates MD5 hash value for the given file

## 1.1.143 GetFileSHA1A Function

Gets the SHA1 hash value for the file

**C++**

```
BOOL WINAPI GetFileSHA1A(LPWSTR fileName, BYTE* buffer, int bufferSize);
```

**File**

**File:** Encryption.h (🔗 see page 169)

**Parameters**

Parameters	Description
LPWSTR fileName	The name of the file
BYTE* buffer	The buffer to store the hash value
int bufferSize	The size of the buffer

**Returns**

The result of the function is equal to TRUE if operation is completed successfully, otherwise the result is FALSE

**Description**

Calculates SHA1 hash value for the given file

## 1.1.144 GetFileSHA1W Function

Gets the SHA1 hash value for the file

**C++**

```
BOOL WINAPI GetFileSHA1W(LPWSTR fileName, BYTE* buffer, int bufferSize);
```

**File**

**File:** Encryption.h (🔗 see page 169)

**Parameters**

Parameters	Description
LPWSTR fileName	The name of the file
BYTE* buffer	The buffer to store the hash value

int bufferSize	The size of the buffer
----------------	------------------------

**Returns**

The result of the function is equal to TRUE if operation is completed successfully, otherwise the result is FALSE

**Description**

Calculates SHA1 hash value for the given file

## 1.1.145 GetFileSHA256A Function

Gets the SHA256 hash value for the file

**C++**

```
BOOL WINAPI GetFileSHA256A(LPSTR fileName, BYTE* buffer, int bufferSize);
```

**File**

**File:** Encryption.h (🔗 see page 169)

**Parameters**

Parameters	Description
LPSTR fileName	The name of the file
BYTE* buffer	The buffer to store the hash value
int bufferSize	The size of the buffer

**Returns**

The result of the function is equal to TRUE if operation is completed successfully, otherwise the result is FALSE

**Description**

Calculates SHA256 hash value for the given file

## 1.1.146 GetFileSHA256W Function

Gets the SHA256 hash value for the file

**C++**

```
BOOL WINAPI GetFileSHA256W(LPWSTR fileName, BYTE* buffer, int bufferSize);
```

**File**

**File:** Encryption.h (🔗 see page 169)

**Parameters**

Parameters	Description
LPWSTR fileName	The name of the file
BYTE* buffer	The buffer to store the hash value
int bufferSize	The size of the buffer

**Returns**

The result of the function is equal to TRUE if operation is completed successfully, otherwise the result is FALSE

**Description**

Calculates SHA256 hash value for the given file

## 1.1.147 GetFileSHA384A Function

This is function GetFileSHA384A.

**C++**

```
BOOL WINAPI GetFileSHA384A(LPSTR fileName, BYTE* buffer, int bufferSize);
```

**File**

**File:** Encryption.h (🔗 see page 169)

## 1.1.148 GetFileSHA384W Function

This is function GetFileSHA384W.

**C++**

```
BOOL WINAPI GetFileSHA384W(LPWSTR fileName, BYTE* buffer, int bufferSize);
```

**File**

**File:** Encryption.h (🔗 see page 169)

## 1.1.149 GetFileSHA512A Function

This is function GetFileSHA512A.

**C++**

```
BOOL WINAPI GetFileSHA512A(LPSTR fileName, BYTE* buffer, int bufferSize);
```

**File**

**File:** Encryption.h (🔗 see page 169)

## 1.1.150 GetFileSHA512W Function

This is function GetFileSHA512W.

**C++**

```
BOOL WINAPI GetFileSHA512W(LPWSTR fileName, BYTE* buffer, int bufferSize);
```

**File**

**File:** Encryption.h (🔗 see page 169)

## 1.1.151 GetFilesCountA Function

Calculates the number of files in the given folder

**C++**

```
int WINAPI GetFilesCountA(LPSTR folderName);
```

**File**

**File:** FileOperations.h (🔗 see page 170)

**Parameters**

Parameters	Description
LPSTR folderName	The name of the folder

**Returns**

The number of files in the given folder

## 1.1.152 GetFilesCountW Function

Calculates the number of files in the given folder

**C++**

```
int WINAPI GetFilesCountW(LPWSTR folderName);
```

**File**

**File:** FileOperations.h (🔗 see page 170)

**Parameters**

Parameters	Description
LPWSTR folderName	The name of the folder

**Returns**

The number of files in the given folder

## 1.1.153 GetHBitmapA Function

Generates Windows Bitmap in memory with QR Code image

**C++**

```
HBITMAP WINAPI GetHBitmapA(LPSTR text, int margin, int size, int level);
```

**File**

**File:** quicol.h (🔗 see page 179)

**Parameters**

Parameters	Description
LPSTR text	The text to encode
int margin	The margin from the border in points
int size	The size of the one point in pixels
int level	The error correction level

**Returns**

The result of the function is HBITMAP handle. You have to destroy it by yourself when its not needed anymore.

**Description**

Generates Windows Bitmap in memory file with encoded text as QR Code image

## 1.1.154 GetHBitmapW Function

Generates Windows Bitmap in memory with QR Code image

**C++**

```
HBITMAP WINAPI GetHBitmapW(LPWSTR text, int margin, int size, int level);
```

**File**

**File:** quicol.h (🔗 see page 179)

**Parameters**

Parameters	Description
LPWSTR text	The text to encode
int margin	The margin from the border in points
int size	The size of the one point in pixels
int level	The error correction level

**Returns**

The result of the function is HBITMAP handle. You have to destroy it by yourself when its not needed anymore.

**Description**

Generates Windows Bitmap in memory file with encoded text as QR Code image

## 1.1.155 GetISOCodeA Function

Returns the country ISO code based on the nationality string

**C++**

```
BOOL WINAPI GetISOCodeA(LPCSTR nationality, LPSTR iso, int bufferSize);
```

**File**

**File:** NationalityConverter.h (🔗 see page 172)

**Parameters**

Parameters	Description
LPCSTR nationality	The nationality string
LPSTR iso	The ISO code memory buffer
int bufferSize	The size if the memory buffer

**Returns**

Returns TRUE if the ISO code is successfully obtained; FALSE otherwise

**Description**

This function converts the nationality string stored on ID card to the country ISO code



## 1.1.156 GetISOCodeW Function

Returns the country ISO code based on the nationality string

**C++**

```
BOOL WINAPI GetISOCodeW(LPCWSTR nationality, LPWSTR iso, int bufferSize);
```

**File**

**File:** NationalityConverter.h (🔗 see page 172)

**Parameters**

Parameters	Description
LPCWSTR nationality	The nationality string
LPWSTR iso	The ISO code memory buffer
int bufferSize	The size of the memory buffer

**Returns**

Returns TRUE if the ISO code is successfully obtained; FALSE otherwise

**Description**

This function converts the nationality string stored on ID card to the country ISO code

## 1.1.157 GetMD5 Function

Gets the MD5 hash value for the content of the memory buffer

**C++**

```
BOOL WINAPI GetMD5(BYTE* source, int sourceSize, BYTE* buffer, int bufferSize);
```

**File**

**File:** Encryption.h (🔗 see page 169)

**Parameters**

Parameters	Description
BYTE* source	The source memory block
int sourceSize	The size of the source memory block
BYTE* buffer	The buffer for the hash value
int bufferSize	The size of the destination buffer

**Returns**

The result of the function is equal to TRUE if operation is completed successfully, otherwise the result is FALSE

**Description**

Calculates MD5 hash value for the given memory buffer

## 1.1.158 GetPNGA Function

Writes PNG image to the memory buffer.

**C++**

```
void WINAPI GetPNGA(LPSTR text, int margin, int size, int level, LPINT bufSize,  
__deref_opt_out void ** ppvBits);
```

**File**

**File:** quicol.h (see page 179)

**Parameters**

Parameters	Description
LPSTR text	The text to encode
int margin	The margin from the image border in points
int size	The size of the point in pixels
int level	The error correction level
LPINT bufSize	The size of the output buffer
__deref_opt_out void ** ppvBits	The buffer when the resulting image is stored

**Description**

Writes PNG image to the memory buffer. Can be useful for web development.

## 1.1.159 GetPNGW Function

Writes PNG image to the memory buffer.

**C++**

```
void WINAPI GetPNGW(LPWSTR text, int margin, int size, int level, LPINT bufSize,  
__deref_opt_out void ** ppvBits);
```

**File**

**File:** quicol.h (see page 179)

**Parameters**

Parameters	Description
LPWSTR text	The text to encode
int margin	The margin from the image border in points
int size	The size of the point in pixels
int level	The error correction level
LPINT bufSize	The size of the output buffer
__deref_opt_out void ** ppvBits	The buffer when the resulting image is stored

**Description**

Writes PNG image to the memory buffer. Can be useful for web development.

## 1.1.160 GetReaderIndexA Function

Returns the zero-based reader index with specified name

**C++**

```
int WINAPI GetReaderIndexA(LPSTR readerName);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
LPSTR readerName	The name of the reader

**Returns**

The zero-based reader index

**Description**

Use this function to get the zero-based index of the card reader with specified name

## 1.1.161 GetReaderIndexW Function

Returns the zero-based reader index with specified name

**C++**

```
int WINAPI GetReaderIndexW(LPWSTR readerName);
```

**File**

File: Swelio.h (see page 173)

**Parameters**

Parameters	Description
LPWSTR readerName	The name of the reader

**Returns**

The zero-based reader index

**Description**

Use this function to get the zero-based index of the card reader with specified name

## 1.1.162 GetReaderNameA Function

Returns the name of the card reader

**C++**

```
int WINAPI GetReaderNameA(int readerNumber, LPSTR strDest, int count);
```

**File**

File: Swelio.h (see page 173)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader
LPSTR strDest	Destination string
int count	The number of characters to be copied

**Returns**

Returns the reader name length

**Description**

Returns the name of the card reader with the specified zero-based index

# 1.1.163 GetReaderNameLenA Function

Returns the length of the reader name

**C++**

```
int WINAPI GetReaderNameLenA(int readerNumber);
```

**File**

File: Swelio.h (see page 173)

**Returns**

The length of the reader name

**Description**

Returns the length of the reader name for the smart card reader with specified zero-based index

# 1.1.164 GetReaderNameLenW Function

Returns the length of the reader name

**C++**

```
int WINAPI GetReaderNameLenW(int readerNumber);
```

**File**

File: Swelio.h (see page 173)

**Returns**

The length of the reader name

**Description**

Returns the length of the reader name for the smart card reader with specified zero-based index

# 1.1.165 GetReaderNameW Function

Returns the name of the card reader

**C++**

```
int WINAPI GetReaderNameW(int readerNumber, LPWSTR strDest, int count);
```

**File**

File: Swelio.h (see page 173)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader
LPWSTR strDest	Destination string
int count	Number of characters to be copied

**Returns**

The character count of the reader name

**Description**

Returns the name of the card reader with the specified zero-based index

## 1.1.166 GetReadersCount Function

Get number of card readers connected to PC

**C++**

```
int WINAPI GetReadersCount(VOID);
```

**File**

**File:** Swelio.h (see page 173)

**Returns**

The number of the connected smart card readers

**Description**

Checks how many smart card readers are connected to PC. If there is no readers connected then the usage of the Swelio Engine is not possible. The engine can control the change of the number of the card readers and can raise an event when the reader is connected or disconnected from PC

## 1.1.167 GetSHA1 Function

Gets the SHA1 hash value for the content of the memory buffer

**C++**

```
BOOL WINAPI GetSHA1(BYTE* source, int sourceSize, BYTE* buffer, int bufferSize);
```

**File**

**File:** Encryption.h (see page 169)

**Parameters**

Parameters	Description
BYTE* source	The source memory block
int sourceSize	The size of the source memory block
BYTE* buffer	The buffer for the hash value
int bufferSize	The size of the destination buffer

**Returns**

The result of the function is equal to TRUE if operation is completed successfully, otherwise the result is FALSE

**Description**

Calculates SHA1 hash value for the given memory buffer

## 1.1.168 GetSHA256 Function

Gets the SHA256 hash value for the content of the memory buffer

**C++**

```
BOOL WINAPI GetSHA256(BYTE* source, int sourceSize, BYTE* buffer, int bufferSize);
```

**File**

**File:** Encryption.h (🔗 see page 169)

**Parameters**

Parameters	Description
BYTE* source	The source memory block
int sourceSize	The size of the source memory block
BYTE* buffer	The buffer for the hash value
int bufferSize	The size of the destination buffer

**Returns**

The result of the function is equal to TRUE if operation is completed successfully, otherwise the result is FALSE

**Description**

Calculates SHA256 hash value for the given memory buffer

## 1.1.169 GetSHA384 Function

This is function GetSHA384.

**C++**

```
BOOL WINAPI GetSHA384(BYTE* source, int sourceSize, BYTE* buffer, int bufferSize);
```

**File**

**File:** Encryption.h (🔗 see page 169)

## 1.1.170 GetSHA512 Function

This is function GetSHA512.

**C++**

```
BOOL WINAPI GetSHA512(BYTE* source, int sourceSize, BYTE* buffer, int bufferSize);
```

**File**

**File:** Encryption.h (🔗 see page 169)

## 1.1.171 GetSelectedReaderIndex Function

Returns the index of the active smart card reader

**C++**

```
int WINAPI GetSelectedReaderIndex();
```

**File**

**File:** Swelio.h (🔗 see page 173)

**Returns**

The index of the selected card reader. The first reader has index 0.

**Description**

The zero-based index of the selected card reader. If there is only one reader is connected to PC then this reader has the index 0 and it's a default (selected) reader.

## 1.1.172 GetStartupA Function

Checks if the application is registered to run when Windows starts

**C++**

```
BOOL WINAPI GetStartupA(LPCSTR appName);
```

**File**

**File:** System.h ([see page 178](#))

**Parameters**

Parameters	Description
LPCSTR appName	The name of the application

## 1.1.173 GetStartupW Function

Checks if the application is registered to run when Windows starts

**C++**

```
BOOL WINAPI GetStartupW(LPCWSTR appName);
```

**File**

**File:** System.h ([see page 178](#))

**Parameters**

Parameters	Description
LPCWSTR appName	The name of the application

## 1.1.174 GetSupportSIS Function

Checks if the SIS cards are supported by the engine

**C++**

```
BOOL WINAPI GetSupportSIS();
```

**File**

**File:** Swelio.h ([see page 173](#))

**Returns**

Returns TRUE if SIS card support is activated, otherwise returns FALSE

**Description**

The SIS card reading operation takes more time than the reading of the eID card. By default when the card is inserted in the

reader the engine will try to detect the card type and the card insertion event will be raised for eID cards only. If you want to support the SIS cards in your application then you have to activate it using SetSupportSIS (see page 129) function. Use GetSupportSIS function to check if the SIS card support is activated.

## 1.1.175 GetTextLineSize Function

This is function GetTextLineSize.

### C++

```
SIZE WINAPI GetTextLineSize(LPCWSTR s, HFONT hFont);
```

### File

File: Graphics.h (see page 171)

## 1.1.176 GetTextSize Function

This is function GetTextSize.

### C++

```
SIZE WINAPI GetTextSize(LPCWSTR s, HFONT hFont, UINT flags);
```

### File

File: Graphics.h (see page 171)

## 1.1.177 GetTextSizeEx Function

This is function GetTextSizeEx.

### C++

```
SIZE WINAPI GetTextSizeEx(LPCWSTR s, HFONT hFont, UINT proposedWidth, UINT flags, BOOL margins);
```

### File

File: Graphics.h (see page 171)

## 1.1.178 HibernateWindows Function

Hibernates Windows

### C++

```
BOOL WINAPI HibernateWindows();
```

### File

File: System.h (see page 178)



## 1.1.179 IgnoreHardwareEvents Function

Ignore USB reader insert / remove events

**C++**

```
void WINAPI IgnoreHardwareEvents ( BOOL value );
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
BOOL value	true - to ignore reader remove / insert events

**Returns**

None

## 1.1.180 IgnoreServiceEvents Function

Ignore smartcard service stop events when reporting readers list change

**C++**

```
void WINAPI IgnoreServiceEvents ( BOOL value );
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
BOOL value	To ignore or not the service event

**Returns**

None

## 1.1.181 InitializeContainer Function

Initializes ASIC container

**C++**

```
LPVOID WINAPI InitializeContainer ( );
```

**File**

**File:** Swelio.h (see page 173)

**Returns**

Retrns container handle pointer

**Description**

This functions initializes container handle needed for all container operations. Must be called first prior to other container-related calls

## 1.1.182 IsAnimatedGIFA Function

Checks if the file is an animated GIF image file

**C++**

```
BOOL WINAPI IsAnimatedGIFA(LPSTR fileName);
```

**File**

**File:** FileOperations.h (🔗 see page 170)

**Parameters**

Parameters	Description
LPSTR fileName	The name of the file

**Returns**

Returns TRUE if the file is an animated GIF image file, otherwise returns FALSE.

## 1.1.183 IsAnimatedGIFW Function

Checks if the file is an animated GIF image file

**C++**

```
BOOL WINAPI IsAnimatedGIFW(LPWSTR fileName);
```

**File**

**File:** FileOperations.h (🔗 see page 170)

**Parameters**

Parameters	Description
LPWSTR fileName	The name of the file

**Returns**

Returns TRUE if the file is an animated GIF image file, otherwise returns FALSE.

## 1.1.184 IsCardActivated Function

Checks the connection between a smart card and a reader

**C++**

```
BOOL WINAPI IsCardActivated();
```

**File**

**File:** Swelio.h (🔗 see page 173)

**Description**

This function checks the connection between the calling application and a smart card in the target reader.

## 1.1.185 IsCardActivatedEx Function

Checks the connection between a smart card and a reader

### C++

```
BOOL WINAPI IsCardActivatedEx(int readerNumber);
```

### File

File: Swelio.h (see page 173)

### Parameters

Parameters	Description
int readerNumber	The zero-based index of the card reader

### Description

This function checks the connection between the calling application and a smart card in the target reader.

## 1.1.186 IsCardPresent Function

Checks if the card is present in the card reader

### C++

```
BOOL WINAPI IsCardPresent();
```

### File

File: Swelio.h (see page 173)

### Returns

Returns TRUE if the card is inserted in the reader, otherwise returns FALSE

### Description

Use IsCardPresent function to check if the card is inserted in the card reader or not

## 1.1.187 IsCardPresentEx Function

Checks if the card is present in the card reader

### C++

```
BOOL WINAPI IsCardPresentEx(int readerNumber);
```

### File

File: Swelio.h (see page 173)

### Parameters

Parameters	Description
int readerNumber	The zero-based index of the card reader

### Returns

Returns TRUE if the card is inserted in the reader, otherwise returns FALSE

Description

Use isCardPresent function to check if the card is inserted in the card reader or not

1.1.188 IsCardStillInserted Function

Checks if the card is still inserted in the card reader

C++

```
BOOL WINAPI IsCardStillInserted( ) ;
```

File

File: Swelio.h (🔗 see page 173)

Description

This function checks if the card is still present in the card reader

1.1.189 IsCardStillInsertedEx Function

Checks if the card is still inserted in the card reader

C++

```
BOOL WINAPI IsCardStillInsertedEx( int readerNumber ) ;
```

File

File: Swelio.h (🔗 see page 173)

Parameters

Parameters	Description
int readerNumber	The zero-based index of the card reader

Description

This function checks if the card is still present in the card reader

1.1.190 IsCitrixSession Function

Checks if application is running in Citrix session

C++

```
BOOL WINAPI IsCitrixSession( ) ;
```

File

File: SystemInfo.h (🔗 see page 179)

1.1.191 IsConnectedToInternet Function

Checks if PC is connected to Internet

**C++**

```
BOOL WINAPI IsConnectedToInternet( );
```

**File**

**File:** SystemInfo.h (🔗 see page 179)

## 1.1.192 IsDirectoryA Function

Verifies that a path is a valid directory.

**C++**

```
BOOL WINAPI IsDirectoryA(LPSTR folderName);
```

**File**

**File:** FileOperations.h (🔗 see page 170)

**Returns**

Returns TRUE if the path is a valid directory, or FALSE otherwise.

**Description**

This function verifies if provided value is the name of the folder

## 1.1.193 IsDirectoryW Function

Verifies that a path is a valid directory.

**C++**

```
BOOL WINAPI IsDirectoryW(LPWSTR folderName);
```

**File**

**File:** FileOperations.h (🔗 see page 170)

**Returns**

Returns TRUE if the path is a valid directory, or FALSE otherwise.

**Description**

This function verifies if provided value is the name of the folder

## 1.1.194 IsEIDCard Function

Check if Belgian EID card is inserted into card reader

**C++**

```
BOOL WINAPI IsEIDCard( );
```

**File**

**File:** Swelio.h (🔗 see page 173)

**Returns**

Returns TRUE, if Belgian eID card is inserted in the reader. If there is no card in the reader or the card of other type is inserted,

returns FALSE

#### Description

If the card is inserted in the reader, this function performs the card type check.

## 1.1.195 IsEIDCardEx Function

Check if Belgian EID card is inserted into card reader

#### C++

```
BOOL WINAPI IsEIDCardEx(int readerNumber);
```

#### File

File: Swelio.h (see page 173)

#### Parameters

Parameters	Description
int readerNumber	The zero-based index of the card reader.

#### Returns

Returns TRUE, if Belgian eID card is inserted in the reader. If there is no card in the reader or the card of other type is inserted, returns FALSE

#### Description

If the card is inserted in the reader, this function performs the card type check.

## 1.1.196 IsEngineActive Function

Checks if the Swelio Engine is activated

#### C++

```
BOOL WINAPI IsEngineActive();
```

#### File

File: Swelio.h (see page 173)

#### Returns

Returns TRUE if the Swelio Engine is active, otherwise returns FALSE.

#### Description

This function checks if the Engine already activated using the StartEngine (see page 131) function.

## 1.1.197 IsFemaleA Function

Checks if the card owner is female

#### C++

```
BOOL WINAPI IsFemaleA(PEidIdentityA identity);
```

#### File

File: Swelio.h (see page 173)

**Parameters**

Parameters	Description
PEidIdentityA identity	The person identity information structure

**Returns**

Returns TRUE if the card owner is female, otherwise returns FALSE

**Description**

Use this function to check the gender of the card owner

## 1.1.198 IsFemaleW Function

Checks if the card owner is female

**C++**

```
BOOL WINAPI IsFemaleW(PEidIdentityW identity);
```

**File**

File: Swelio.h (see page 173)

**Parameters**

Parameters	Description
PEidIdentityW identity	The person identity information structure

**Returns**

Returns TRUE if the card owner is female, otherwise returns FALSE

**Description**

Use this function to check the gender of the card owner

## 1.1.199 IsMaleA Function

Checks if the card owner is male

**C++**

```
BOOL WINAPI IsMaleA(PEidIdentityA identity);
```

**File**

File: Swelio.h (see page 173)

**Parameters**

Parameters	Description
PEidIdentityA identity	The person identity information structure

**Returns**

Returns TRUE if the card owner is male, otherwise returns FALSE

**Description**

Use this function to check the gender of the card owner

# 1.1.200 IsMaleW Function

Checks if the card owner is male

C++

```
BOOL WINAPI IsMaleW(PEidIdentityW identity);
```

File

File: Swelio.h (🔗 see page 173)

Parameters

Parameters	Description
PEidIdentityW identity	The person identity information structure

Returns

Returns TRUE if the card owner is male, otherwise returns FALSE

Description

Use this function to check the gender of the card owner

# 1.1.201 IsMediaCenter Function

Checks if the Media Center version of Windows is installed

C++

```
BOOL WINAPI IsMediaCenter();
```

File

File: SystemInfo.h (🔗 see page 179)

# 1.1.202 IsMetroActive Function

Checks if metro interface is active

C++

```
BOOL WINAPI IsMetroActive();
```

File

File: SystemInfo.h (🔗 see page 179)

# 1.1.203 IsMultiTouchReady Function

Checks if the system is multi touch ready

C++

```
BOOL WINAPI IsMultiTouchReady();
```



**File**

**File:** SystemInfo.h (see page 179)

## 1.1.204 IsNativeWin64 Function

Checks if the application is native 64 bit executable

**C++**

```
BOOL WINAPI IsNativeWin64();
```

**File**

**File:** SystemInfo.h (see page 179)

## 1.1.205 IsRemoteSession Function

Checks if application is running in RDP session

**C++**

```
BOOL WINAPI IsRemoteSession();
```

**File**

**File:** SystemInfo.h (see page 179)

## 1.1.206 IsSISCard Function

Check if Belgian SIS card is inserted into card reader

**C++**

```
BOOL WINAPI IssISCard();
```

**File**

**File:** Swelio.h (see page 173)

**Returns**

Returns TRUE, if Belgian SIS card is inserted in the reader. If there is no card in the reader or the card of other type is inserted, returns FALSE

**Description**

If the card is inserted in the reader, this function performs the card type check.

## 1.1.207 IsSISCardEx Function

Check if Belgian SIS card is inserted into card reader

**C++**

```
BOOL WINAPI IssISCardEx(int readerNumber);
```

**File**

**File:** Swelio.h (🔗 see page 173)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader.

**Returns**

Returns TRUE, if Belgian SIS card is inserted in the reader. If there is no card in the reader or the card of other type is inserted, returns FALSE

**Description**

If the card is inserted in the reader, this function performs the card type check.

## 1.1.208 IsTabletPC Function

Checks if the application is running on the Tablet PC

**C++**

```
BOOL WINAPI IsTabletPC( );
```

**File**

**File:** SystemInfo.h (🔗 see page 179)

## 1.1.209 IsUnicodeFileA Function

Checks if the file is UNICODE file

**C++**

```
BOOL WINAPI IsUnicodeFileA(LPCSTR fileName);
```

**File**

**File:** FileOperations.h (🔗 see page 170)

**Parameters**

Parameters	Description
LPCSTR fileName	The name of the file

**Returns**

Returns TRUE if file is stored in UNICODE format, otherwise returns FALSE.

**Description**

This function checks the file encoding based on BOM (Byte Order Mark).

## 1.1.210 IsUnicodeFileW Function

Checks if the file is UNICODE file

**C++**

```
BOOL WINAPI IsValidUnicodeFileW(LPCWSTR fileName);
```

**File**

**File:** FileOperations.h (see page 170)

**Parameters**

Parameters	Description
LPCWSTR fileName	The name of the file

**Returns**

Returns TRUE if file is stored in UNICODE format, otherwise returns FALSE.

**Description**

This function checks the file encoding based on BOM (Byte Order Mark).

## 1.1.211 IsValidFileNameA Function

Checks if provided string is a valid file name

**C++**

```
BOOL WINAPI IsValidFileNameA(LPSTR fileName);
```

**File**

**File:** FileOperations.h (see page 170)

**Parameters**

Parameters	Description
LPSTR fileName	The name of the file

**Returns**

Returns TRUE if provided string is valid file name, otherwise returns FALSE

**Description**

Checks if provided string is a valid file name and does not contain any illegal characters

## 1.1.212 IsValidFileNameW Function

Checks if provided string is a valid file name

**C++**

```
BOOL WINAPI IsValidFileNameW(LPWSTR fileName);
```

**File**

**File:** FileOperations.h (see page 170)

**Parameters**

Parameters	Description
LPWSTR fileName	The name of the file

**Returns**

Returns TRUE if provided string is valid file name, otherwise returns FALSE

**Description**

Checks if provided string is a valid file name and does not contain any illegal characters

## 1.1.213 IsValidPathNameA Function

Checks if provided string is a valid file path

**C++**

```
BOOL WINAPI IsValidPathNameA(LPSTR fileName);
```

**File**

**File:** FileOperations.h (🔗 see page 170)

**Parameters**

Parameters	Description
LPSTR fileName	The file path to check

**Returns**

Returns TRUE if provided string is valid file path, otherwise returns FALSE

**Description**

Checks if provided string is a valid file path and does not contain any illegal characters

## 1.1.214 IsValidPathNameW Function

Checks if provided string is a valid file path

**C++**

```
BOOL WINAPI IsValidPathNameW(LPWSTR fileName);
```

**File**

**File:** FileOperations.h (🔗 see page 170)

**Parameters**

Parameters	Description
LPWSTR fileName	The file path to check

**Returns**

Returns TRUE if provided string is valid file path, otherwise returns FALSE

**Description**

Checks if provided string is a valid file path and does not contain any illegal characters

## 1.1.215 IsWindows10 Function

Checks if PC is running Windows 10 or better

**C++**

```
BOOL WINAPI IsWindows10();
```

**File**

**File:** SystemInfo.h (see page 179)

## 1.1.216 IsWindows7 Function

Checks if PC is running Windows 7 or better

**C++**

```
BOOL WINAPI IsWindows7();
```

**File**

**File:** SystemInfo.h (see page 179)

## 1.1.217 IsWindows8 Function

Checks if PC is Running Windows 8 or better

**C++**

```
BOOL WINAPI IsWindows8();
```

**File**

**File:** SystemInfo.h (see page 179)

## 1.1.218 IsWindowsVista Function

Checks if PC is running Windows Vista or better

**C++**

```
BOOL WINAPI IsWindowsVista();
```

**File**

**File:** SystemInfo.h (see page 179)

## 1.1.219 IsWindowsXP Function

Checks if PC is running Windows XP

**C++**

```
BOOL WINAPI IsWindowsXP();
```

**File**

**File:** SystemInfo.h (see page 179)

## 1.1.220 IsWindowsXPSP2 Function

Checks if PC is running Windows XP with Service Pack 2 installed

**C++**

```
BOOL WINAPI IsWindowsXPSP2();
```

**File**

**File:** SystemInfo.h ([↗](#) see page 179)

## 1.1.221 IsWow64 Function

Checks if the 32 bit application runs on 64 bit Windows

**C++**

```
BOOL WINAPI IsWow64();
```

**File**

**File:** SystemInfo.h ([↗](#) see page 179)

## 1.1.222 LayeredWndProcA Function

The default window procedure for the layered window

**C++**

```
LRESULT CALLBACK LayeredWndProcA(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam);
```

**File**

**File:** System.h ([↗](#) see page 178)

## 1.1.223 LayeredWndProcW Function

The default window procedure for the layered window

**C++**

```
LRESULT CALLBACK LayeredWndProcW(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam);
```

**File**

**File:** System.h ([↗](#) see page 178)

## 1.1.224 LoadAddressA Function

This is function LoadAddressA.

**C++**

```
void WINAPI LoadAddressA(LPSTR fileName, PEidAddressA address);
```

**File**

File: Swelio.h ([see page 173](#))

## 1.1.225 LoadAddressW Function

This is function LoadAddressW.

**C++**

```
void WINAPI LoadAddressW(LPWSTR fileName, PEidAddressW address);
```

**File**

File: Swelio.h ([see page 173](#))

## 1.1.226 LoadBitmapJPG Function

This is function LoadBitmapJPG.

**C++**

```
HBITMAP WINAPI LoadBitmapJPG(LPCWSTR szFile, LPINT lpiWidth, LPINT lpiHeight,  
__deref_opt_out void ** ppvBits);
```

**File**

File: Graphics.h ([see page 171](#))

## 1.1.227 LoadBitmapPNG Function

This is function LoadBitmapPNG.

**C++**

```
HBITMAP WINAPI LoadBitmapPNG(LPCWSTR szFile, LPINT lpiWidth, LPINT lpiHeight,  
__deref_opt_out void ** ppvBits);
```

**File**

File: Graphics.h ([see page 171](#))

## 1.1.228 LoadCertificateA Function

Reads the certificate from a file

**C++**

```
void WINAPI LoadCertificateA(LPSTR fileName, PEidCertificate certificate);
```

**File**

File: Swelio.h ([see page 173](#))

**Parameters**

Parameters	Description
LPSTR fileName	The source file name
PEidCertificate certificate	The pointer to EidCertificate (see page 149) structure

**Description**

Use this function to read the certificate from the file

## 1.1.229 LoadCertificateW Function

Reads the certificate from a file

**C++**

```
void WINAPI LoadCertificateW(LPWSTR fileName, PEidCertificate certificate);
```

**File**

File: Swelio.h (see page 173)

**Parameters**

Parameters	Description
LPWSTR fileName	The source file name
PEidCertificate certificate	The pointer to EidCertificate (see page 149) structure

**Description**

Use this function to read the certificate from the file

## 1.1.230 LoadIdentityA Function

Reads the raw identity information from a file

**C++**

```
void WINAPI LoadIdentityA(LPSTR fileName, PEidIdentityA identity);
```

**File**

File: Swelio.h (see page 173)

**Parameters**

Parameters	Description
LPSTR fileName	The name of the source file
PEidIdentityA identity	The pointer to EidIdentityA (see page 150) structure

**Description**

Use this function to read back the identity information stored to the file using SaveldIdentityA (see page 115) function

## 1.1.231 LoadIdentityW Function

Reads the raw identity information from a file



C++

```
void WINAPI LoadIdentityW(LPWSTR fileName, PEidIdentityW identity);
```

File

File: Swelio.h (see page 173)

Parameters

Parameters	Description
LPWSTR fileName	The name of the source file
PEidIdentityW identity	The pointer to EidIdentityW (see page 154) structure

Description

Use this function to read back the identity information stored to the file using SaveldIdentityW (see page 115) function

## 1.1.232 LoadPNGResource Function

This is function LoadPNGResource.

C++

```
HBITMAP WINAPI LoadPNGResource(HMODULE handle, LPCWSTR szName, LPINT lpiWidth, LPINT lpiHeight, __deref_opt_out void ** ppvBits);
```

File

File: Graphics.h (see page 171)

## 1.1.233 LoadPhotoA Function

Loads photo from a file

C++

```
void WINAPI LoadPhotoA(PeidPicture photo, LPSTR fileName);
```

File

File: Swelio.h (see page 173)

Parameters

Parameters	Description
PeidPicture photo	The pointer to EidPicture (see page 156) structure
LPSTR fileName	Destination file name

Description

Loads raw picture data from a file

## 1.1.234 LoadPhotoW Function

Loads photo from a file

C++

```
void WINAPI LoadPhotoW(PeidPicture photo, LPWSTR fileName);
```

**File**

**File:** Swelio.h (🔗 see page 173)

**Parameters**

Parameters	Description
PeidPicture photo	The pointer to EidPicture (🔗 see page 156) structure
LPWSTR fileName	Destination file name

**Description**

Loads raw picture data from a file

## 1.1.235 MakeCompatibleBitmap Function

This is function MakeCompatibleBitmap.

**C++**

```
HBITMAP WINAPI MakeCompatibleBitmap(HBITMAP source, int width, int height);
```

**File**

**File:** Graphics.h (🔗 see page 171)

## 1.1.236 MakeSoundFromFileA Function

Plays the wave sound from the file

**C++**

```
void WINAPI MakeSoundFromFileA(LPCSTR soundName);
```

**File**

**File:** System.h (🔗 see page 178)

**Parameters**

Parameters	Description
LPCSTR soundName	The name of the file

**Description**

This function plays a sound specified by the given file name.

## 1.1.237 MakeSoundFromFileW Function

Plays the wave sound from the file

**C++**

```
void WINAPI MakeSoundFromFileW(LPCWSTR soundName);
```

**File**

**File:** System.h (🔗 see page 178)

**Parameters**

Parameters	Description
LPCWSTR soundName	The name of the file

**Description**

This function plays a sound specified by the given file name.

## 1.1.238 MakeSoundFromResourceA Function

Plays the wave sound from the resource

**C++**

```
void WINAPI MakeSoundFromResourceA(HMODULE hModule, LPCSTR soundName);
```

**File**

**File:** System.h (🔗 see page 178)

**Parameters**

Parameters	Description
HMODULE hModule	Handle to the executable file that contains the resource to be loaded.
LPCSTR soundName	A string that specifies the sound to play.

**Description**

This function plays a sound specified by the given resource name.

## 1.1.239 MakeSoundFromResourceW Function

Plays the wave sound from the resource

**C++**

```
void WINAPI MakeSoundFromResourceW(HMODULE hModule, LPCWSTR soundName);
```

**File**

**File:** System.h (🔗 see page 178)

**Parameters**

Parameters	Description
HMODULE hModule	Handle to the executable file that contains the resource to be loaded.
LPCWSTR soundName	A string that specifies the sound to play.

**Description**

This function plays a sound specified by the given resource name.

## 1.1.240 NonRepudiationSignatureAlgoA Function

This is function NonRepudiationSignatureAlgoA.

**C++**

```
BOOL WINAPI NonRepudiationSignatureAlgoA(int readerNumber, LPSTR pinCode, BYTE* dataHash,  
int hashSize, BYTE* signature, LPDWORD signatureSize, BYTE algorithm);
```

**File**

File: Swelio.h ([see page 173](#))

## 1.1.241 NonRepudiationSignatureAlgoW Function

This is function NonRepudiationSignatureAlgoW.

**C++**

```
BOOL WINAPI NonRepudiationSignatureAlgoW(int readerNumber, LPWSTR pinCode, BYTE* dataHash,  
int hashSize, BYTE* signature, LPDWORD signatureSize, BYTE algorithm);
```

**File**

File: Swelio.h ([see page 173](#))

## 1.1.242 PointsToPixels Function

This is function PointsToPixels.

**C++**

```
int WINAPI PointsToPixels(int points);
```

**File**

File: Graphics.h ([see page 171](#))

## 1.1.243 PortAvailable Function

Checks if the port with specified number is available

**C++**

```
BOOL WINAPI PortAvailable(int port);
```

**File**

File: SystemInfo.h ([see page 179](#))

## 1.1.244 ReadAddressA Function

Read address information from Belgian eID card

**C++**

```
BOOL WINAPI ReadAddressA(PEidAddressA address);
```

**File**

File: Swelio.h ([see page 173](#))

**Parameters**

Parameters	Description
PEidAddressA address	The pointer to the address information structure

**Returns**

TRUE when information is successfully received from the card; otherwise returns FALSE

## 1.1.245 ReadAddressExA Function

Read address information from Belgian eID card

**C++**

```
BOOL WINAPI ReadAddressExA(int readerNumber, PEidAddressA address);
```

**File**

File: Swelio.h (see page 173)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader.
PEidAddressA address	The pointer to the address information structure

**Returns**

TRUE when information is successfully received from the card; otherwise returns FALSE

## 1.1.246 ReadAddressExW Function

Read address information from Belgian eID card

**C++**

```
BOOL WINAPI ReadAddressExW(int readerNumber, PEidAddressW address);
```

**File**

File: Swelio.h (see page 173)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader.
PEidAddressW address	The pointer to the address information structure

**Returns**

TRUE when information is successfully received from the card; otherwise returns FALSE

## 1.1.247 ReadAddressW Function

Read address information from Belgian eID card

**C++**

```
BOOL WINAPI ReadAddressW(PEidAddressW address);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
PEidAddressW address	the pointer to the address information structure

**Returns**

TRUE when information is successfully received from the card; otherwise returns FALSE

## 1.1.248 ReadAuthenticationCertificate Function

Read Authentication Certificate to memory

**C++**

```
BOOL WINAPI ReadAuthenticationCertificate(PEidCertificate certificate);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
certificate	The pointer to EidCertificate (see page 149) structure

**Returns**

TRUE when certificate is successfully received from the card; otherwise returns FALSE

**Description**

Read Authentication Certificate from the card to EidCertificate (see page 149) structure

## 1.1.249 ReadAuthenticationCertificateEx Function

Read Authentication Certificate to memory

**C++**

```
BOOL WINAPI ReadAuthenticationCertificateEx(int readerNumber, PEidCertificate certificate);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader
certificate	The pointer to EidCertificate (see page 149) structure

**Returns**

TRUE when certificate is successfully received from the card; otherwise returns FALSE

**Description**

Read Authentication Certificate from the card to EidCertificate (see page 149) structure

## 1.1.250 ReadBufferFromFileA Function

Reads the content of the file to the memory buffer

**C++**

```
void WINAPI ReadBufferFromFileA(LPSTR fileName, BYTE* buffer, int bufferSize);
```

**File**

**File:** FileOperations.h (see page 170)

**Parameters**

Parameters	Description
LPSTR fileName	The name of the file
BYTE* buffer	The address of the memory block
int bufferSize	The size of the memory block

**Description**

Use this function to retrieve the content of the file to the memory block

## 1.1.251 ReadBufferFromFileW Function

Reads the content of the file to the memory buffer

**C++**

```
void WINAPI ReadBufferFromFileW(LPWSTR fileName, BYTE* buffer, int bufferSize);
```

**File**

**File:** FileOperations.h (see page 170)

**Parameters**

Parameters	Description
LPWSTR fileName	The name of the file
BYTE* buffer	The address of the memory block
int bufferSize	The size of the memory block

**Description**

Use this function to retrieve the content of the file to the memory block

## 1.1.252 ReadCaCertificate Function

Read Ca Certificate to memory

**C++**

```
BOOL WINAPI ReadCaCertificate(PEidCertificate certificate);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
certificate	The pointer to EidCertificate (see page 149) structure

**Returns**

TRUE when certificate is successfully received from the card; otherwise returns FALSE

**Description**

Read Ca Certificate to EidCertificate (see page 149) structure

## 1.1.253 ReadCaCertificateEx Function

Read Ca Certificate to memory

**C++**

```
BOOL WINAPI ReadCaCertificateEx(int readerNumber, PEidCertificate certificate);
```

**File**

File: Swelio.h (see page 173)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader
certificate	The pointer to EidCertificate (see page 149) structure

**Returns**

TRUE when certificate is successfully received from the card; otherwise returns FALSE

**Description**

Read Ca Certificate to EidCertificate (see page 149) structure

## 1.1.254 ReadIdentityA Function

Read identity information from Belgian eID card

**C++**

```
BOOL WINAPI ReadIdentityA(PEidIdentityA identity);
```

**File**

File: Swelio.h (see page 173)

**Parameters**

Parameters	Description
PEidIdentityA identity	The pointer to the identity information structure

**Returns**

TRUE when information is successfully received from the card; otherwise returns FALSE



# 1.1.255 ReadIdentityExA Function

Read identity information from Belgian eID card

C++

```
BOOL WINAPI ReadIdentityExA(int readerNumber, PEidIdentityA identity);
```

File

File: Swelio.h (see page 173)

Parameters

Parameters	Description
int readerNumber	The zero-based index of the card reader.
PEidIdentityA identity	The pointer to the identity information structure

Returns

Returns TRUE when information is successfully received from the card; otherwise returns FALSE

# 1.1.256 ReadIdentityExW Function

Read identity information from Belgian eID card

C++

```
BOOL WINAPI ReadIdentityExW(int readerNumber, PEidIdentityW identity);
```

File

File: Swelio.h (see page 173)

Parameters

Parameters	Description
int readerNumber	The zero-based index of the card reader.
PEidIdentityW identity	The pointer to the identity information structure

Returns

TRUE when information is successfully received from the card; otherwise returns FALSE

# 1.1.257 ReadIdentityW Function

Read identity information from Belgian eID card

C++

```
BOOL WINAPI ReadIdentityW(PEidIdentityW identity);
```

File

File: Swelio.h (see page 173)

Parameters

Parameters	Description
PEidIdentityW identity	The pointer to the identity information structure

**Returns**

TRUE when information is successfully received from the card; otherwise returns FALSE

## 1.1.258 ReadNonRepudiationCertificate Function

Read Non Repudiation Certificate to memory

**C++**

```
BOOL WINAPI ReadNonRepudiationCertificate(PEidCertificate certificate);
```

**File**

File: Swelio.h ([↗](#) see page 173)

**Parameters**

Parameters	Description
certificate	The pointer to EidCertificate ( <a href="#">↗</a> see page 149) structure

**Returns**

TRUE when certificate is successfully received from the card; otherwise returns FALSE

**Description**

Read Non Repudiation Certificate to EidCertificate ([↗](#) see page 149) structure

## 1.1.259 ReadNonRepudiationCertificateEx Function

Read Non Repudiation Certificate to memory

**C++**

```
BOOL WINAPI ReadNonRepudiationCertificateEx(int readerNumber, PEidCertificate certificate);
```

**File**

File: Swelio.h ([↗](#) see page 173)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader
certificate	The pointer to EidCertificate ( <a href="#">↗</a> see page 149) structure

**Returns**

TRUE when certificate is successfully received from the card; otherwise returns FALSE

**Description**

Read Non Repudiation Certificate to EidCertificate ([↗](#) see page 149) structure

## 1.1.260 ReadPhoto Function

Reads a photo from a card

**C++**

```
BOOL WINAPI ReadPhoto(PeidPicture photo);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
PeidPicture photo	The pointer to EidPicture (see page 156) structure

**Returns**

Returns TRUE if the photo is retrieved from the card, otherwise return FALSE

**Description**

Reads a photo from Belgian eID card to EidPicture (see page 156) structure. This structure holds the raw image bytes and the length of the image bytes array

## 1.1.261 ReadPhotoAsBitmap Function

Reads the picture from the card, converts it to bitmap and returns the bitmap handle  
Description: Reads the photo from the Belgian eID card and returns the bitmap handle  
Reading the photo from the card is a time consuming operation. Do it only when needed.

**C++**

```
HBITMAP WINAPI ReadPhotoAsBitmap();
```

**File**

**File:** Swelio.h (see page 173)

**Returns**

A handle to a bitmap indicates success. NULL indicates failure.

## 1.1.262 ReadPhotoAsBitmapEx Function

Reads the picture from the card, converts it to bitmap and returns the bitmap handle  
Description: Reads the photo from the Belgian eID card and returns the Windows bitmap handle  
Reading the photo from the card is a time consuming operation. Do it only when needed.

**C++**

```
HBITMAP WINAPI ReadPhotoAsBitmapEx(int readerNumber);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader.

**Returns**

A handle to a bitmap indicates success. NULL indicates failure.

## 1.1.263 ReadPhotoEx Function

Reads a photo from a card

### C++

```
BOOL WINAPI ReadPhotoEx(int readerNumber, PeidPicture photo);
```

### File

File: Swelio.h (see page 173)

### Parameters

Parameters	Description
int readerNumber	The zero-based index of the card reader.
PeidPicture photo	The pointer to EidPicture (see page 156) structure

### Returns

Returns TRUE if the photo is retrieved from the card, otherwise return FALSE

### Description

Reads a photo from Belgian eID card to EidPicture (see page 156) structure

## 1.1.264 ReadRootCaCertificate Function

Read Root Ca Certificate to memory

### C++

```
BOOL WINAPI ReadRootCaCertificate(PEidCertificate certificate);
```

### File

File: Swelio.h (see page 173)

### Parameters

Parameters	Description
certificate	The pointer to EidCertificate (see page 149) structure

### Returns

TRUE when certificate is successfully received from the card; otherwise returns FALSE

### Description

Read Root Ca Certificate to EidCertificate (see page 149) structure

## 1.1.265 ReadRootCaCertificateEx Function

Read Root Ca Certificate to memory

### C++

```
BOOL WINAPI ReadRootCaCertificateEx(int readerNumber, PEidCertificate certificate);
```

### File

File: Swelio.h (see page 173)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader
certificate	The pointer to EidCertificate ( <a href="#">↗</a> see page 149) structure

**Returns**

TRUE when certificate is successfully received from the card; otherwise returns FALSE

**Description**

Read Root Ca Certificate to EidCertificate ([↗](#) see page 149) structure

## 1.1.266 ReadRrnCertificate Function

Read Rrn Certificate to memory

**C++**

```
BOOL WINAPI ReadRrnCertificate(PEidCertificate certificate);
```

**File**

**File:** Swelio.h ([↗](#) see page 173)

**Parameters**

Parameters	Description
certificate	The pointer to EidCertificate ( <a href="#">↗</a> see page 149) structure

**Returns**

TRUE when certificate is successfully received from the card; otherwise returns FALSE

**Description**

Read Rrn Certificate to EidCertificate ([↗](#) see page 149) structure

## 1.1.267 ReadRrnCertificateEx Function

Read Rrn Certificate to memory

**C++**

```
BOOL WINAPI ReadRrnCertificateEx(int readerNumber, PEidCertificate certificate);
```

**File**

**File:** Swelio.h ([↗](#) see page 173)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader
certificate	The pointer to EidCertificate ( <a href="#">↗</a> see page 149) structure

**Returns**

TRUE when certificate is successfully received from the card; otherwise returns FALSE

**Description**

Read Rrn Certificate to EidCertificate ([↗](#) see page 149) structure

# 1.1.268 ReadSISCardA Function

Read Belgian SIS card.

C++

```
BOOL WINAPI ReadSISCardA(P SISRecordA identity);
```

File

File: Swelio.h (see page 173)

Parameters

Parameters	Description
P SISRecordA	The pointer to SISRecordA (see page 166) structure

Returns

TRUE when information is successfully received from the card; otherwise returns FALSE

Description

Read the public information from the Belgian SIS card. The SIS card is the social security card of each Belgian resident (Belgian or foreigner)

# 1.1.269 ReadSISCardExA Function

Read Belgian SIS card.

C++

```
BOOL WINAPI ReadSISCardExA(int readerNumber, P SISRecordA identity);
```

File

File: Swelio.h (see page 173)

Parameters

Parameters	Description
int readerNumber	The zero-based index of the card reader.
P SISRecordA	The pointer to SISRecordA (see page 166) structure

Returns

TRUE when information is successfully received from the card; otherwise returns FALSE

Description

Read the public information from the Belgian SIS card. The SIS card is the social security card of each Belgian resident (Belgian or foreigner)

# 1.1.270 ReadSISCardExW Function

Read Belgian SIS card.

C++

```
BOOL WINAPI ReadSISCardExW(int readerNumber, P SISRecordW identity);
```

File

File: Swelio.h (see page 173)

Parameters

Parameters	Description
int readerNumber	The zero-based index of the card reader.
PSISRecordW	The pointer to SISRecordW (see page 166) structure

Returns

TRUE when information is successfully received from the card; otherwise returns FALSE

Description

Read the public information from the Belgian SIS card. The SIS card is the social security card of each Belgian resident (Belgian or foreigner)

# 1.1.271 ReadSISCardW Function

Read Belgian SIS card.

C++

```
BOOL WINAPI ReadSISCardW(PSISRecordW identity);
```

File

File: Swelio.h (see page 173)

Parameters

Parameters	Description
PSISRecordW	The pointer to SISRecordW (see page 166) structure

Returns

TRUE when information is successfully received from the card; otherwise returns FALSE

Description

Read the public information from the Belgian SIS card. The SIS card is the social security card of each Belgian resident (Belgian or foreigner)

# 1.1.272 RecycleBinEmpty Function

Returns TRUE if Windows Recycle Bin is empty

C++

```
BOOL WINAPI RecycleBinEmpty();
```

File

File: SystemInfo.h (see page 179)

# 1.1.273 ReloadReadersList Function

Reloads the list of the available card readers

C++

```
void WINAPI ReloadReadersList();
```

File

File: Swelio.h (see page 173)

Description

When the card reader is inserted or removed you may need to reload the list of the available card readers

# 1.1.274 RemoveCallback Function

Remove callback procedure for card events

C++

```
void WINAPI RemoveCallback();
```

File

File: Swelio.h (see page 173)

Description

Use this function to deactivate card events callback procedure

# 1.1.275 RemoveStartupA Function

Removes the application from the list of the automatically started applications

C++

```
void WINAPI RemoveStartupA(LPCSTR appName);
```

File

File: System.h (see page 178)

Parameters

Parameters	Description
LPCSTR appName	The name of the application

Description

For application that starts automatically when Windows starts removes it from the automatically launching applications list

# 1.1.276 RemoveStartupW Function

Removes the application from the list of the automatically started applications

C++

```
void WINAPI RemoveStartupW(LPCWSTR appName);
```

File

File: System.h (see page 178)



**Parameters**

Parameters	Description
LPCWSTR appName	The name of the application

**Description**

For application that starts automatically when Windows starts removes it from the automatically launching applications list

## 1.1.277 RestoreWindowSubclassA Function

Restores window standard procedure

**C++**

```
void WINAPI RestoreWindowSubclassA(HWND hwnd);
```

**File**

**File:** System.h (🔗 see page 178)

**Parameters**

Parameters	Description
HWND hwnd	The window handle

## 1.1.278 RestoreWindowSubclassW Function

Restores window standard procedure

**C++**

```
void WINAPI RestoreWindowSubclassW(HWND hwnd);
```

**File**

**File:** System.h (🔗 see page 178)

**Parameters**

Parameters	Description
HWND hwnd	The window handle

## 1.1.279 SaveAddressA Function

This is function SaveAddressA.

**C++**

```
void WINAPI SaveAddressA(LPSTR fileName);
```

**File**

**File:** Swelio.h (🔗 see page 173)

# 1.1.280 SaveAddressW Function

This is function SaveAddressW.

**C++**

```
void WINAPI SaveAddressW(LPWSTR fileName);
```

**File**

**File:** Swelio.h (see page 173)

# 1.1.281 SaveAuthenticationCertificateA Function

Save Authentication Certificate to a file

**C++**

```
void WINAPI SaveAuthenticationCertificateA(LPSTR fileName);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
LPSTR fileName	File name to store the certificate

**Description**

Read Authentication Certificate from the card and save it to a file.

# 1.1.282 SaveAuthenticationCertificateExW Function

Save Authentication Certificate to a file

**C++**

```
void WINAPI SaveAuthenticationCertificateExW(LPWSTR fileName, int readerNumber);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
LPWSTR fileName	File name to store the certificate
int readerNumber	The reader index, starting from 0

**Description**

Read Authentication Certificate from the card and save it to a file.

## 1.1.283 SaveAuthenticationCertificateW Function

Save Authentication Certificate to a file

**C++**

```
void WINAPI SaveAuthenticationCertificateW(LPWSTR fileName);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
LPWSTR fileName	File name to store the certificate

**Description**

Read Authentication Certificate from the card and save it to a file.

## 1.1.284 SaveCaCertificateA Function

Save Ca Certificate to a file

**C++**

```
void WINAPI SaveCaCertificateA(LPSTR fileName);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
LPSTR fileName	File name to store the certificate

**Description**

Read Ca Certificate from the card and save it to a file

## 1.1.285 SaveCaCertificateExW Function

Save Ca Certificate to a file

**C++**

```
void WINAPI SaveCaCertificateExW(LPWSTR fileName, int readerNumber);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
LPWSTR fileName	File name to store the certificate
int readerNumber	The reader index, starting from 0

**Description**

Read Ca Certificate from the card and save it to a file

## 1.1.286 SaveCaCertificateW Function

Save Ca Certificate to a file

**C++**

```
void WINAPI SaveCaCertificateW(LPWSTR fileName);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
LPWSTR fileName	File name to store the certificate

**Description**

Read Ca Certificate from the card and save it to a file

## 1.1.287 SaveCardToToXMLStreamExA Function

Read eID card and save the information to XML buffer

**C++**

```
BOOL WINAPI SaveCardToToXMLStreamExA(int readerNumber, void* buffer);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader.
void* buffer	The memory buffer to store information

**Returns**

Returns TRUE if the information is retrieved from the card, otherwise returns FALSE

**Description**

Use this function to read the information about the owner of the card and save it to XML buffer in the memory.

## 1.1.288 SaveCardToToXMLStreamExW Function

Read eID card and save the information to XML buffer

**C++**

```
BOOL WINAPI SaveCardToToXMLStreamExW(int readerNumber, void* buffer);
```

**File**

**File:** Swelio.h (🔗 see page 173)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader.
void* buffer	The memory buffer to store information

**Returns**

Returns TRUE if the information is retrieved from the card, otherwise returns FALSE

**Description**

Use this function to read the information about the owner of the card and save it to XML buffer in the memory.

## 1.1.289 SaveCardToXmlA Function

Read eID card and save the information to XML file

**C++**

```
BOOL WINAPI SaveCardToXmlA(LPSTR fileName);
```

**File**

**File:** Swelio.h (🔗 see page 173)

**Parameters**

Parameters	Description
LPSTR fileName	File name to store information

**Returns**

Returns TRUE if the information is retrieved from the card, otherwise returns FALSE

**Description**

Use this function to read the information about the owner of the card and save it to XML file.

## 1.1.290 SaveCardToXmlExA Function

Read eID card and save the information to XML file

**C++**

```
BOOL WINAPI SaveCardToXmlExA(int readerNumber, LPSTR fileName);
```

**File**

**File:** Swelio.h (🔗 see page 173)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader.
LPSTR fileName	File name to store information

**Returns**

Returns TRUE if the information is retrieved from the card, otherwise returns FALSE

**Description**

Use this function to read the information about the owner of the card and save it to XML file.

## 1.1.291 SaveCardToXmlExW Function

Read eID card and save the information to XML file

**C++**

```
BOOL WINAPI SaveCardToXmlExW(int readerNumber, LPWSTR fileName);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader.
LPWSTR fileName	File name to store information

**Returns**

Returns TRUE if the information is retrieved from the card, otherwise returns FALSE

**Description**

Use this function to read the information about the owner of the card and save it to XML file.

## 1.1.292 SaveCardToXmlW Function

Read eID card and save the information to XML file

**C++**

```
BOOL WINAPI SaveCardToXmlW(LPWSTR fileName);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
LPWSTR fileName	File name to store information

**Returns**

Returns TRUE if the information is retrieved from the card, otherwise returns FALSE

**Description**

Use this function to read the information about the owner of the card and save it to XML file.

## 1.1.293 SaveContainer Function

Save container to the file

C++

```
BOOL WINAPI SaveContainer(LPVOID container, LPWSTR fileName);
```

File

File: Swelio.h (see page 173)

Parameters

Parameters	Description
LPVOID container	Container handle, which is allocated by calling InitializeContainer (see page 77) function
LPWSTR fileName	Desired name of the container file

Returns

Returns true if the operation is successful, otherwise returns false

Description

After adding all necessary files to the container call this function to save container to the file

# 1.1.294 SaveIdentityA Function

Saves identity information to a file

C++

```
void WINAPI SaveIdentityA(LPSTR fileName);
```

File

File: Swelio.h (see page 173)

Parameters

Parameters	Description
LPSTR fileName	The name of the destination file

Description

Use this function to store the raw identity information from the Belgian eID card to a file. You can use LoadIdentityA (see page 92) to read this information from the file to EidlIdentityA (see page 150) structure

# 1.1.295 SaveIdentityW Function

Saves identity information to a file

C++

```
void WINAPI SaveIdentityW(LPWSTR fileName);
```

File

File: Swelio.h (see page 173)

Parameters

Parameters	Description
LPWSTR fileName	The name of the destination file

Description

Use this function to store the raw identity information from the Belgian eID card to a file. You can use LoadIdentityW (see

page 92) to read this information from the file to EidIdentityW (see page 154) structure

# 1.1.296 SaveNonRepudiationCertificateA Function

Save Non Repudiation Certificate to a file

C++

```
void WINAPI SaveNonRepudiationCertificateA(LPSTR fileName) ;
```

File

File: Swelio.h (see page 173)

Parameters

Parameters	Description
LPSTR fileName	File name to store the certificate

Description

Read Non Repudiation Certificate from the card and save it to a file

# 1.1.297 SaveNonRepudiationCertificateExW Function

Save Non Repudiation Certificate to a file

C++

```
void WINAPI SaveNonRepudiationCertificateExW(LPWSTR fileName, int readerNumber) ;
```

File

File: Swelio.h (see page 173)

Parameters

Parameters	Description
LPWSTR fileName	File name to store the certificate
int readerNumber	The reader index, starting from 0

Description

Read Non Repudiation Certificate from the card and save it to a file

# 1.1.298 SaveNonRepudiationCertificateW Function

Save Non Repudiation Certificate to a file

C++

```
void WINAPI SaveNonRepudiationCertificateW(LPWSTR fileName) ;
```

File

File: Swelio.h (see page 173)

Parameters

Parameters	Description
LPWSTR fileName	File name to store the certificate



**Description**

Read Non Repudiation Certificate from the card and save it to a file

## 1.1.299 SavePersonCsvToStreamA Function

Read eID card and save the identity information to CSV memory buffer

**C++**

```
BOOL WINAPI SavePersonCsvToStreamA(int readerNumber, void* buffer);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
int readerNumber	the card reader index
void* buffer	Memory buffer

**Returns**

Returns TRUE if the information is retrieved from the card, otherwise returns FALSE

**Description**

Use this function to read the information about the owner of the card and save it to CSV memory buffer.

## 1.1.300 SavePersonCsvToStreamW Function

Read eID card and save the identity information to CSV memory buffer

**C++**

```
BOOL WINAPI SavePersonCsvToStreamW(int readerNumber, void* buffer);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
int readerNumber	the card reader index
void* buffer	Memory buffer

**Returns**

Returns TRUE if the information is retrieved from the card, otherwise returns FALSE

**Description**

Use this function to read the information about the owner of the card and save it to CSV memory buffer.

## 1.1.301 SavePersonToCsvA Function

Read eID card and save the identity information and address to CSV file

**C++**

```
BOOL WINAPI SavePersonToCsvA(LPSTR fileName);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
LPSTR fileName	File name to store information

**Returns**

Returns TRUE if the information is retrieved from the card, otherwise returns FALSE

**Description**

Use this function to read the information about the owner of the card and save it to CSV file.

## 1.1.302 SavePersonToCsvExA Function

Read eID card and save the identity information and address to CSV file

**C++**

```
BOOL WINAPI SavePersonToCsvExA(int readerNumber, LPSTR fileName);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader.
LPSTR fileName	File name to store information

**Returns**

Returns TRUE if the information is retrieved from the card, otherwise returns FALSE

**Description**

Use this function to read the information about the owner of the card and save it to CSV file.

## 1.1.303 SavePersonToCsvExW Function

Read eID card and save the identity information and address to CSV file

**C++**

```
BOOL WINAPI SavePersonToCsvExW(int readerNumber, LPWSTR fileName);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader.
LPWSTR fileName	File name to store information

**Returns**

Returns TRUE if the information is retrieved from the card, otherwise returns FALSE

**Description**

Use this function to read the information about the owner of the card and save it to CSV file.

## 1.1.304 SavePersonToCsvW Function

Read eID card and save the identity information and address to CSV file

**C++**

```
BOOL WINAPI SavePersonToCsvW(LPWSTR fileName);
```

**File**

File: Swelio.h (see page 173)

**Parameters**

Parameters	Description
LPWSTR fileName	File name to store information

**Returns**

Returns TRUE if the information is retrieved from the card, otherwise returns FALSE

**Description**

Use this function to read the information about the owner of the card and save it to CSV file.

## 1.1.305 SavePhotoA Function

Save photo to a file

**C++**

```
void WINAPI SavePhotoA(PeidPicture photo, LPSTR fileName);
```

**File**

File: Swelio.h (see page 173)

**Parameters**

Parameters	Description
PeidPicture photo	The pointer to EidPicture (see page 156) structure
LPSTR fileName	Destination file name

**Description**

Save the raw picture data to a file

## 1.1.306 SavePhotoAsBitmapA Function

Save the picture from the card to Windows Bitmap file Description: Reads the photo from the Belgian eID card and writes it to the file as bitmap image. Reading the photo from the card is a time consuming operation. Do it only when needed.

**C++**

```
BOOL WINAPI SavePhotoAsBitmapA(LPSTR fileName);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
LPSTR fileName	File name to store the photo

**Returns**

Returns TRUE if the photo is retrieved from the card, otherwise return FALSE

## 1.1.307 SavePhotoAsBitmapExA Function

Reads the picture from the card and saves it to Windows Bitmap file  
Description: Reads the photo from the Belgian eID card and writes it to the file as bitmap image. Reading the photo from the card is a time consuming operation. Do it only when needed.

**C++**

```
BOOL WINAPI SavePhotoAsBitmapExA(int readerNumber, LPSTR fileName);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader.
LPSTR fileName	File name to store the photo

**Returns**

Returns TRUE if the photo is retrieved from the card, otherwise return FALSE

## 1.1.308 SavePhotoAsBitmapExW Function

Reads the picture from the card and saves it to Windows Bitmap file  
Description: Reads the photo from the Belgian eID card and writes it to the file as bitmap image. Reading the photo from the card is a time consuming operation. Do it only when needed.

**C++**

```
BOOL WINAPI SavePhotoAsBitmapExW(int readerNumber, LPWSTR fileName);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
int readerNumber	the zero-based index of the card reader.
LPWSTR fileName	File name to store the photo

**Returns**

Returns TRUE if the photo is retrieved from the card, otherwise return FALSE

## 1.1.309 SavePhotoAsBitmapW Function

Save the picture from the card to Windows Bitmap file  
Description: Reads the photo from the Belgian eID card and writes it to the file as bitmap image. Reading the photo from the card is a time consuming operation. Do it only when needed.

**C++**

```
BOOL WINAPI SavePhotoAsBitmapW(LPWSTR fileName);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
LPWSTR fileName	File name to store the photo

**Returns**

Returns TRUE if the photo is retrieved from the card, otherwise return FALSE

## 1.1.310 SavePhotoAsJpegA Function

Save the picture from the card to JPG file  
Description: Reads the photo from the Belgian eID card and writes it to the file as JPG image. Reading the photo from the card is a time consuming operation. Do it only when needed.

**C++**

```
BOOL WINAPI SavePhotoAsJpegA(LPSTR fileName);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
LPSTR fileName	File name to store the photo

**Returns**

Returns TRUE if the photo is retrieved from the card, otherwise return FALSE

## 1.1.311 SavePhotoAsJpegExA Function

Save the picture from the card to JPG file  
Description: Reads the photo from the Belgian eID card and writes it to the file as JPG image. Reading the photo from the card is a time consuming operation. Do it only when needed.

**C++**

```
BOOL WINAPI SavePhotoAsJpegExA(int readerNumber, LPSTR fileName);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
int readerNumber	the zero-based index of the card reader.
LPSTR fileName	File name to store the photo

**Returns**

Returns TRUE if the photo is retrieved from the card, otherwise return FALSE

## 1.1.312 SavePhotoAsJpegExW Function

Save the picture from the card to JPG file Description: Reads the photo from the Belgian eID card and writes it to the file as JPG image. Reading the photo from the card is a time consuming operation. Do it only when needed.

**C++**

```
BOOL WINAPI SavePhotoAsJpegExW(int readerNumber, LPWSTR fileName);
```

**File**

File: Swelio.h (see page 173)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader.
LPWSTR fileName	File name to store the photo

**Returns**

Returns TRUE if the photo is retrieved from the card, otherwise return FALSE

## 1.1.313 SavePhotoAsJpegW Function

Save the picture from the card to JPG file Description: Reads the photo from the Belgian eID card and writes it to the file as JPG image. Reading the photo from the card is a time consuming operation. Do it only when needed.

**C++**

```
BOOL WINAPI SavePhotoAsJpegW(LPWSTR fileName);
```

**File**

File: Swelio.h (see page 173)

**Parameters**

Parameters	Description
LPWSTR fileName	File name to store the photo

**Returns**

Returns TRUE if the photo is retrieved from the card, otherwise return FALSE

## 1.1.314 SavePhotoW Function

Saves photo to a file

**C++**

```
void WINAPI SavePhotoW(PeidPicture photo, LPWSTR fileName);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
PeidPicture photo	The pointer to EidPicture (see page 156) structure
LPWSTR fileName	Destination file name

**Description**

Saves the raw picture data to a file

## 1.1.315 SaveRootCaCertificateA Function

Save Root Ca Certificate to a file

**C++**

```
void WINAPI SaveRootCaCertificateA(LPSTR fileName);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
LPSTR fileName	File name to store the certificate

**Description**

Read Root CA certificate from the card and save it to a file

## 1.1.316 SaveRootCaCertificateExW Function

Save Root Ca Certificate to a file

**C++**

```
void WINAPI SaveRootCaCertificateExW(LPWSTR fileName, int readerNumber);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
LPWSTR fileName	File name to store the certificate
int readerNumber	The reader index, starting from 0

**Description**

Read Root CA certificate from the card and save it to a file

# 1.1.317 SaveRootCaCertificateW Function

Save Root Ca Certificate to a file

C++

```
void WINAPI SaveRootCaCertificateW(LPWSTR fileName);
```

File

File: Swelio.h (see page 173)

Parameters

Parameters	Description
LPWSTR fileName	File name to store the certificate

Description

Read Root CA certificate from the card and save it to a file

# 1.1.318 SaveRrnCertificateA Function

Save RRN Certificate to a file

C++

```
void WINAPI SaveRrnCertificateA(LPSTR fileName);
```

File

File: Swelio.h (see page 173)

Parameters

Parameters	Description
LPSTR fileName	File name to store the certificate

Description

Read RRN certificate from the card and save it to a file

# 1.1.319 SaveRrnCertificateExW Function

Save RRN Certificate to a file

C++

```
void WINAPI SaveRrnCertificateExW(LPWSTR fileName, int readerNumber);
```

File

File: Swelio.h (see page 173)

Parameters

Parameters	Description
LPWSTR fileName	File name to store the certificate
int readerNumber	The reader index, starting from 0



**Description**

Read RRN certificate from the card and save it to a file

## 1.1.320 SaveRrnCertificateW Function

Save RRN Certificate to a file

**C++**

```
void WINAPI SaveRrnCertificateW(LPWSTR fileName);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
LPWSTR fileName	File name to store the certificate

**Description**

Read RRN certificate from the card and save it to a file

## 1.1.321 SelectReader Function

When more than 1 reader connected, select the reader with specified number

**C++**

```
BOOL WINAPI SelectReader(int readerNumber);
```

**File**

**File:** Swelio.h (see page 173)

**Parameters**

Parameters	Description
int readerNumber	The reader index, starting from 0

**Returns**

TRUE if the reader is selected, FALSE if the reader with specified number does not exist

**Description**

Selects the default card reader using the zero-based reader index. The first reader has number 0, second - 1, etc... You can read the information only from one selected reader at once.

## 1.1.322 SelectReaderByNameA Function

Select active smart card reader by providing the reader name

**C++**

```
BOOL WINAPI SelectReaderByNameA(LPSTR readerName);
```

**File**

**File:** Swelio.h (🔗 see page 173)

**Parameters**

Parameters	Description
LPSTR readerName	The name of the card reader

**Returns**

TRUE if the reader is selected. If the reader with specified name is not found - returns FALSE

**Description**

Activates the reader with specified name

## 1.1.323 SelectReaderByNameW Function

Select active smart card reader by providing the reader name

**C++**

```
BOOL WINAPI SelectReaderByNameW(LPWSTR readerName);
```

**File**

**File:** Swelio.h (🔗 see page 173)

**Parameters**

Parameters	Description
LPWSTR readerName	The name of the card reader

**Returns**

Returns TRUE if the reader is selected. If the reader with specified name is not found - returns FALSE

**Description**

Activates the reader with specified name

## 1.1.324 SendAPDU Function

This is function SendAPDU.

**C++**

```
BOOL WINAPI SendAPDU(int readerNumber, LPCBYTE apdu, DWORD apduLen, PCHAR result, LPDWORD len);
```

**File**

**File:** Swelio.h (🔗 see page 173)

## 1.1.325 ServerAccessible Function

Check if the specified server is accessible

C++

```
BOOL WINAPI ServerAccessible(LPWSTR name);
```

File

File: Network.h (🔗 see page 172)

# 1.1.326 SetCallback Function

Activates callback procedure for card status change event

C++

```
void WINAPI SetCallback(CALLBACK_HANDLER callback, LPVOID userContext);
```

File

File: Swelio.h (🔗 see page 173)

Parameters

Parameters	Description
CALLBACK_HANDLER callback	The pointer to callback procedure
LPVOID userContext	The user defined value passed to the callback procedure

Description

Your application can be notified about insertion or removal of the card from the card reader and the changes of the available card readers list (the reader is connected or disconnected from PC) Use this function to install the callback procedure

# 1.1.327 SetContainerCanonization Function

Set XML canonization standard of the ASIC container

C++

```
BOOL WINAPI SetContainerCanonization(LPVOID container, LPWSTR canonization);
```

File

File: Swelio.h (🔗 see page 173)

Parameters

Parameters	Description
LPVOID container	Container handle, which is allocated by calling InitializeContainer (🔗 see page 77) function
LPWSTR canonization	XML canonization URI string

Returns

Returns true if the operation is successful, otherwise returns false

Description

Call this function to set the canonization string value of the ASIC container

# 1.1.328 SetContainerTimeServer Function

Set time server URI for digital signature of the ASIC container

C++

```
BOOL WINAPI SetContainerTimeServer(LPVOID container, LPWSTR timeServer);
```

File

File: Swelio.h (🔗 see page 173)

Parameters

Parameters	Description
LPVOID container	Container handle, which is allocated by calling InitializeContainer (🔗 see page 77) function
canonization	the address of the time server

Returns

Returns true if the operation is successful, otherwise returns false

Description

Call this function to set the timeserver URI value

# 1.1.329 SetMWCompatibility Function

Set the compatibility mode with the old version of the oficial EID MiddleWare

C++

```
void WINAPI SetMWCompatibility();
```

File

File: Swelio.h (🔗 see page 173)

Description

The compatibility mode can be useful when the MiddleWare version 1.x or 2.x is installed on the target PC. Usually the more recent MiddleWare is used and this function is provided for backward compatibility only

# 1.1.330 SetStartupA Function

Register application to run when Windows starts

C++

```
void WINAPI SetStartupA(LPCSTR appName, LPCWSTR appPath);
```

File

File: System.h (🔗 see page 178)

Parameters

Parameters	Description
LPCSTR appName	The name of the application

LPCWSTR appPath	The path to the application executable
-----------------	--

## 1.1.331 SetStartupW Function

Register application to run when Windows starts

**C++**

```
void WINAPI SetStartupW(LPCWSTR appName, LPCWSTR appPath);
```

**File**

**File:** System.h (🔗 see page 178)

**Parameters**

Parameters	Description
LPCWSTR appName	The name of the application
LPCWSTR appPath	The path to the application executable

## 1.1.332 SetSupportSIS Function

Activates or deactivates SIS card support by engine

**C++**

```
void WINAPI SetSupportSIS(BOOL value);
```

**File**

**File:** Swelio.h (🔗 see page 173)

**Parameters**

Parameters	Description
BOOL value	The SIS card support status

**Description**

Use SetSupportSIS to activate or deactivate the SIS card detection and reading. Even if SIS card support is activated it can be used only with ACR38U card readers Other card readers are not supported.

## 1.1.333 ShellCopyFileA Function

Copies file to the new location

**C++**

```
void WINAPI ShellCopyFileA(LPSTR oldName, LPSTR newName);
```

**File**

**File:** FileOperations.h (🔗 see page 170)

**Parameters**

Parameters	Description
LPSTR oldName	The source file name
LPSTR newName	The destination file name

**Description**

Copies file to the new location using Windows shell copy routine

## 1.1.334 ShellCopyFileW Function

Copies file to the new location

**C++**

```
void WINAPI ShellCopyFileW(LPWSTR oldName, LPWSTR newName);
```

**File**

**File:** FileOperations.h (see page 170)

**Parameters**

Parameters	Description
LPWSTR oldName	The source file name
LPWSTR newName	The destination file name

**Description**

Copies file to the new location using Windows shell copy routine

## 1.1.335 ShutdownWindows Function

Logs off the interactive user, shuts down the system.

**C++**

```
BOOL WINAPI ShutdownWindows(UINT flags);
```

**File**

**File:** System.h (see page 178)

**Returns**

If the function succeeds returns TRUE, otherwise returns FALSE

**Description**

Logs off the interactive user, shuts down the system, or shuts down and restarts the system. It sends the WM\_QUERYENDSESSION message to all applications to determine if they can be terminated.

This function accepts the following parameter:

flags : The shutdown type. This parameter must include one of the following values:

Value	Meaning
EWX_LOGOFF	Shuts down all processes running in the logon session of the process that called the ExitWindowsEx function. Then it logs the user off.
EWX_POWEROFF	Shuts down the system and turns off the power. The system must support the power-off feature.
EWX_REBOOT	Shuts down the system and then restarts the system.
EWX_RESTARTAPPS	Shuts down the system and then restarts it
EWX_SHUTDOWN	Shuts down the system to a point at which it is safe to turn off the power.

## 1.1.336 SignPdfFile Function

Digitally sign PDF file using Belgian EID card

### C++

```
BOOL WINAPI SignPdfFile(int readerNumber, LPWSTR fileName);
```

### File

File: Swelio.h (see page 173)

### Parameters

Parameters	Description
LPWSTR fileName	The name of the PDF file

### Returns

Returns true if the operation is successful, otherwise returns false

### Description

Sign PDF document using ID card. The card must be inserted in the reader before calling this function

## 1.1.337 SignPdfFileEx Function

Digitally sign PDF file using Belgian EID card

### C++

```
BOOL WINAPI SignPdfFileEx(LPVOID container, LPWSTR fileName);
```

### File

File: Swelio.h (see page 173)

### Parameters

Parameters	Description
LPVOID container	Pointer to container structure which is allocated by calling InitializeContainer (see page 77) function
LPWSTR fileName	The name of the PDF file

### Returns

Returns true if the operation is successful, otherwise returns false

### Description

Sign PDF document using ID card. The card must be inserted in the reader before calling this function

## 1.1.338 StartEngine Function

Activates the Swelio Engine.

### C++

```
BOOL WINAPI StartEngine();
```

**File**

**File:** Swelio.h (🔗 see page 173)

**Returns**

Returns TRUE if the Swelio Engine is successfully started; otherwise returns FALSE

**Description**

This procedure must be called first before any other functions from Swelio library can be used.

## 1.1.339 StopEngine Function

Deactivates the Swelio Engine

**C++**

```
void WINAPI StopEngine();
```

**File**

**File:** Swelio.h (🔗 see page 173)

**Description**

Deactivates the Swelio Engine and clean up the used memory. Call this procedure at the end of you application once to finalize the usage of the Swelio Engine.

## 1.1.340 StretchNativeBitmap Function

This is function StretchNativeBitmap.

**C++**

```
BOOL WINAPI StretchNativeBitmap(HBITMAP src, HBITMAP dst, int srcWidth, int srcHeight, int dstWidth, int dstHeight);
```

**File**

**File:** Graphics.h (🔗 see page 171)

## 1.1.341 StripFileNameA Function

Replaces environment variable names with values

**C++**

```
void WINAPI StripFileNameA(LPCSTR fileName, LPSTR fullName);
```

**File**

**File:** FileOperations.h (🔗 see page 170)

**Parameters**

Parameters	Description
LPCSTR fileName	The source file name
LPSTR fullName	The expanded file name



**Description**

This function expands environment-variable strings and replaces them with their defined values in the file name.

## 1.1.342 StripFileNameW Function

Replaces environment variable names with values

**C++**

```
void WINAPI StripFileNameW(LPCWSTR fileName, LPWSTR fullName);
```

**File**

**File:** FileOperations.h (🔗 see page 170)

**Parameters**

Parameters	Description
LPCWSTR fileName	The source file name
LPWSTR fullName	The expanded file name

**Description**

This function expands environment-variable strings and replaces them with their defined values in the file name.

## 1.1.343 SuspendWindows Function

Suspends Windows

**C++**

```
BOOL WINAPI SuspendWindows();
```

**File**

**File:** System.h (🔗 see page 178)

## 1.1.344 TurnMonitorOff Function

Turns the monitor off

**C++**

```
void WINAPI TurnMonitorOff();
```

**File**

**File:** System.h (🔗 see page 178)

## 1.1.345 TurnMonitorOn Function

Turns the monitor on

**C++**

```
void WINAPI TurnMonitorOn();
```

File

File: System.h (🔗 see page 178)

# 1.1.346 UpdateWindowPosition Function

Updated the window position

C++

```
void WINAPI UpdateWindowPosition(HWND handle, int x, int y);
```

File

File: System.h (🔗 see page 178)

Parameters

Parameters	Description
HWND handle	The handle of the window
int x	New horizontal coordinate
int y	New vertical coordinate

# 1.1.347 VerifyContainer Function

Verify the signatures and the integrity of the ASIC container

C++

```
BOOL WINAPI VerifyContainer(LPVOID container, LPWSTR fileName);
```

File

File: Swelio.h (🔗 see page 173)

Parameters

Parameters	Description
LPVOID container	Container handle, which is allocated by calling InitializeContainer (🔗 see page 77) function
LPWSTR fileName	Container file name

Returns

Returns true if the operation is successful, otherwise returns false

Description

Call this function to verify the integrity of the ASIC container

# 1.1.348 VerifyPinA Function

Verify PIN code

C++

```
BOOL WINAPI VerifyPinA(LPSTR value);
```

**File**

**File:** Swelio.h (🔗 see page 173)

**Parameters**

Parameters	Description
LPSTR value	PIN code to verify

**Returns**

TRUE when the correct PIN code is provided; otherwise returns FALSE

## 1.1.349 VerifyPinExA Function

Verify PIN code

**C++**

```
BOOL WINAPI VerifyPinExA(int readerNumber, LPSTR value);
```

**File**

**File:** Swelio.h (🔗 see page 173)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader.
LPSTR value	PIN code to verify

**Returns**

TRUE when the correct PIN code is provided; otherwise returns FALSE

## 1.1.350 VerifyPinExW Function

Verify PIN code

**C++**

```
BOOL WINAPI VerifyPinExW(int readerNumber, LPWSTR value);
```

**File**

**File:** Swelio.h (🔗 see page 173)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader.
LPWSTR value	PIN code to verify

**Returns**

TRUE when the correct PIN code is provided; otherwise returns FALSE

# 1.1.351 VerifyPinW Function

Verify PIN code

C++

```
BOOL WINAPI VerifyPinW(LPWSTR value);
```

File

File: Swelio.h (see page 173)

Parameters

Parameters	Description
LPWSTR value	PIN code to verify

Returns

TRUE when the correct PIN code is provided; otherwise returns FALSE

# 1.1.352 VerifySignature Function

Verifies the signature from the specified hash value.

C++

```
BOOL WINAPI VerifySignature(PeIdCertificate certificate, BYTE* buffer, int bufferSize, BYTE* signature, DWORD signatureSize);
```

File

File: Swelio.h (see page 173)

Parameters

Parameters	Description
PeIdCertificate certificate	The public certificate
BYTE* buffer	The hash buffer
int bufferSize	The size of the hash buffer
BYTE* signature	The signature to be verified.
DWORD signatureSize	The size of the signature buffer

Returns

Returns TRUE if the signature is valid for the hash; otherwise, FALSE.

Description

Verify the signature using the public certificate of the signer

# 1.1.353 WriteBufferToFileA Function

Writes the memory buffer to file

C++

```
void WINAPI WriteBufferToFileA(LPSTR fileName, BYTE* buffer, int bufferSize);
```

File

File: FileOperations.h (🔗 see page 170)

Parameters

Parameters	Description
LPSTR fileName	The name of the file
BYTE* buffer	The address of the memory block
int bufferSize	The size of the memory block

Description

This function stores the content of the memory buffer to the file.

# 1.1.354 WriteBufferToFileW Function

Writes the memory buffer to file

C++

```
void WINAPI WriteBufferToFileW(LPWSTR fileName, BYTE* buffer, int bufferSize);
```

File

File: FileOperations.h (🔗 see page 170)

Parameters

Parameters	Description
LPWSTR fileName	The name of the file
BYTE* buffer	The address of the memory block
int bufferSize	The size of the memory block

Description

This function stores the content of the memory buffer to the file.

# 1.1.355 fpreset Function

This is function fpreset.

C++

```
void fpreset();
```


File

File: System.h (🔗 see page 178)



# 1.2 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

## Enumerations

	Name	Description
	tagCardEventType ( <a href="#">see page 139</a> )	The type of the reader event
	CardEventType ( <a href="#">see page 148</a> )	The type of the reader event

## Structures

	Name	Description
	structErrorInformation ( <a href="#">see page 139</a> )	Information about error when operation with the card or certificate is performed
	tagEidAddressA ( <a href="#">see page 139</a> )	EID address information, stored on the card - ANSI version
	tagEidAddressW ( <a href="#">see page 140</a> )	EID address information, stored on the card - UNICODE version
	tagEidCertificate ( <a href="#">see page 140</a> )	Certificate, stored on EID card
	tagEidIdentityA ( <a href="#">see page 140</a> )	Identity information stored on EID card - ANSI version
	tagEidIdentityExA ( <a href="#">see page 142</a> )	Identity information stored on EID card - ANSI version
	tagEidIdentityExW ( <a href="#">see page 143</a> )	Identity information stored on EID card - UNICODE version
	tagEidIdentityW ( <a href="#">see page 145</a> )	Identity information stored on EID card - UNICODE version
	tagEidPicture ( <a href="#">see page 146</a> )	Raw picture data from EID card
	tagSISRecordA ( <a href="#">see page 147</a> )	Public information stored on Belgian SIS card - ANSI version. The SIS card is the social security card of each Belgian resident (Belgian or foreigner)
	tagSISRecordW ( <a href="#">see page 148</a> )	Public information stored on Belgian SIS card - UNICODE version. The SIS card is the social security card of each Belgian resident (Belgian or foreigner)
	EidAddressA ( <a href="#">see page 149</a> )	EID address information, stored on the card - ANSI version
	EidAddressW ( <a href="#">see page 149</a> )	EID address information, stored on the card - UNICODE version
	EidCertificate ( <a href="#">see page 149</a> )	Certificate, stored on EID card
	EidIdentityA ( <a href="#">see page 150</a> )	Identity information stored on EID card - ANSI version
	EidIdentityExA ( <a href="#">see page 151</a> )	Identity information stored on EID card - ANSI version
	EidIdentityExW ( <a href="#">see page 153</a> )	Identity information stored on EID card - UNICODE version
	EidIdentityW ( <a href="#">see page 154</a> )	Identity information stored on EID card - UNICODE version
	EidPicture ( <a href="#">see page 156</a> )	Raw picture data from EID card
	ErrorInformation ( <a href="#">see page 156</a> )	Information about error when operation with the card or certificate is performed
	PEidAddressA ( <a href="#">see page 156</a> )	EID address information, stored on the card - ANSI version
	PEidAddressW ( <a href="#">see page 157</a> )	EID address information, stored on the card - UNICODE version
	PEidCertificate ( <a href="#">see page 157</a> )	Certificate, stored on EID card
	PEidIdentityA ( <a href="#">see page 157</a> )	Identity information stored on EID card - ANSI version
	PEidIdentityExA ( <a href="#">see page 159</a> )	Identity information stored on EID card - ANSI version
	PEidIdentityExW ( <a href="#">see page 160</a> )	Identity information stored on EID card - UNICODE version
	PEidIdentityW ( <a href="#">see page 162</a> )	Identity information stored on EID card - UNICODE version
	PErrorInformation ( <a href="#">see page 163</a> )	Information about error when operation with the card or certificate is performed
	PSISRecordA ( <a href="#">see page 164</a> )	Public information stored on Belgian SIS card - ANSI version. The SIS card is the social security card of each Belgian resident (Belgian or foreigner)
	PSISRecordW ( <a href="#">see page 164</a> )	Public information stored on Belgian SIS card - UNICODE version. The SIS card is the social security card of each Belgian resident (Belgian or foreigner)
	PeidPicture ( <a href="#">see page 165</a> )	Raw picture data from EID card

	SISRecordA (see page 166)	Public information stored on Belgian SIS card - ANSI version. The SIS card is the social security card of each Belgian resident (Belgian or foreigner)
	SISRecordW (see page 166)	Public information stored on Belgian SIS card - UNICODE version. The SIS card is the social security card of each Belgian resident (Belgian or foreigner)

## 1.2.1 structErrorInformation Structure

Information about error when operation with the card or certificate is performed

C++

```
struct structErrorInformation {
    int Code;
    WCHAR Description[ERROR_MAX_DESCRIPTION + 1];
};
```

File

File: CardStructures.h (see page 168)

## 1.2.2 tagCardEventType Enumeration

The type of the reader event

C++

```
enum tagCardEventType {
    ewtUnknownEvent,
    ewtCardInsert,
    ewtCardRemove,
    ewtReadersChange
};
```

File

File: CardEvents.h (see page 168)

Members

Members	Description
ewtUnknownEvent	Unknown event
ewtCardInsert	The card was inserted in the reader
ewtCardRemove	The card was removed from the reader
ewtReadersChange	The readers list changed

## 1.2.3 tagEidAddressA Structure

EID address information, stored on the card - ANSI version

C++

```
struct tagEidAddressA {
    char street[EID_MAX_STREET_LEN+1];
    char zip[EID_MAX_ZIP_LEN+1];
    char municipality[EID_MAX_MUNICIPALITY_LEN+1];
};
```

**File****File:** CardStructures.h (see page 168)**Members**

Members	Description
char street[EID_MAX_STREET_LEN+1];	Street name
char zip[EID_MAX_ZIP_LEN+1];	ZIP code
char municipality[EID_MAX_MUNICIPALITY_LEN+1];	Municipality

## 1.2.4 tagEidAddressW Structure

EID address information, stored on the card - UNICODE version

**C++**

```
struct tagEidAddressW {  
    WCHAR street[EID_MAX_STREET_LEN+1];  
    WCHAR zip[EID_MAX_ZIP_LEN+1];  
    WCHAR municipality[EID_MAX_MUNICIPALITY_LEN+1];  
};
```

**File****File:** CardStructures.h (see page 168)**Members**

Members	Description
WCHAR street[EID_MAX_STREET_LEN+1];	Street name
WCHAR zip[EID_MAX_ZIP_LEN+1];	ZIP code
WCHAR municipality[EID_MAX_MUNICIPALITY_LEN+1];	Municipality

## 1.2.5 tagEidCertificate Structure

Certificate, stored on EID card

**C++**

```
struct tagEidCertificate {  
    BYTE certificate[EID_MAX_CERT_LEN+1];  
    int certificateLength;  
};
```

**File****File:** CardStructures.h (see page 168)**Members**

Members	Description
BYTE certificate[EID_MAX_CERT_LEN+1];	Certificate raw data buffer
int certificateLength;	Certificate data length

## 1.2.6 tagEidIdentityA Structure

Identity information stored on EID card - ANSI version



**C++**

```

struct tagEidIdentityA {
    char cardNumber[EID_MAX_CARD_NUMBER_LEN+1];
    char chipNumber[EID_MAX_CHIP_NUMBER_LEN+1];
    char validityDateBegin[EID_MAX_DATE_BEGIN_LEN+1];
    char validityDateEnd[EID_MAX_DATE_END_LEN +1];
    char municipality[EID_MAX_DELIVERY_MUNICIPALITY_LEN+1];
    char nationalNumber[EID_MAX_NATIONAL_NUMBER_LEN+1];
    char name[EID_MAX_NAME_LEN+1];
    char firstName1[EID_MAX_FIRST_NAME1_LEN+1];
    char firstName2[EID_MAX_FIRST_NAME2_LEN+1];
    char nationality[EID_MAX_NATIONALITY_LEN+1];
    char birthLocation[EID_MAX_BIRTHPLACE_LEN+1];
    char birthDate[EID_MAX_BIRTHDATE_LEN+1];
    char sex[EID_MAX_SEX_LEN+1];
    char nobleCondition[EID_MAX_NOBLE_CONDITION_LEN+1];
    int documentType;
    BOOL whiteCane;
    BOOL yellowCane;
    BOOL extendedMinority;
    char duplicate[EID_MAX_DUPLICATE_LEN + 1];
    char specialOrganization[EID_MAX_SPECIALORGANIZATION_LEN + 1];
    BOOL memberOfFamily;
    char dateAndCountryOfProtection[EID_MAX_DATEANDCOUNTRYOFPROTECTION_LEN + 1];
    char workPermitType[EID_MAX_WORKPERMITTYPE_LEN + 1];
    char vat1[EID_MAX_VAT1_LEN + 1];
    char vat2[EID_MAX_VAT2_LEN + 1];
    char regionalFileNumber[EID_MAX_REGIONALFILENUMBER_LEN + 1];
    char brexitMention1[EID_MAX_BREXITMENTION1_LEN + 1];
    char brexitMention2[EID_MAX_BREXITMENTION2_LEN + 1];
};

```

**File**

File: CardStructures.h (see page 168)

**Members**

Members	Description
char cardNumber[EID_MAX_CARD_NUMBER_LEN+1];	Electronic ID card number
char chipNumber[EID_MAX_CHIP_NUMBER_LEN+1];	Electronic ID card physical chip number
char validityDateBegin[EID_MAX_DATE_BEGIN_LEN+1];	Card validity start date
char validityDateEnd[EID_MAX_DATE_END_LEN +1];	Card validity end date
char municipality[EID_MAX_DELIVERY_MUNICIPALITY_LEN+1];	Card delivery municipality
char nationalNumber[EID_MAX_NATIONAL_NUMBER_LEN+1];	National number
char name[EID_MAX_NAME_LEN+1];	Surname
char firstName1[EID_MAX_FIRST_NAME1_LEN+1];	First name (2 first given names)
char firstName2[EID_MAX_FIRST_NAME2_LEN+1];	First name part 2 (first letter of the 3rd given name).
char nationality[EID_MAX_NATIONALITY_LEN+1];	Nationality
char birthLocation[EID_MAX_BIRTHPLACE_LEN+1];	Birth location
char birthDate[EID_MAX_BIRTHDATE_LEN+1];	Birth date
char sex[EID_MAX_SEX_LEN+1];	Sex
char nobleCondition[EID_MAX_NOBLE_CONDITION_LEN+1];	Noble condition
int documentType;	Document type code (Belgian citizen card, Kids Card, Foreigner card)
BOOL whiteCane;	White cane (blind people)

BOOL yellowCane;	Yellow cane (partially sighted people)
BOOL extendedMinority;	Extended minority
char duplicate[EID_MAX_DUPLICATE_LEN + 1];	Duplicata
char specialOrganization[EID_MAX_SPECIALORGANIZATION_LEN + 1];	Special Organization
BOOL memberOfFamily;	Member of family
char dateAndCountryOfProtection[EID_MAX_DATEANDCOUNTRYOFPROTECTION_LEN + 1];	Date and country of protection
char workPermitType[EID_MAX_WORKPERMITTYPE_LEN + 1];	Work Permit type
char vat1[EID_MAX_VAT1_LEN + 1];	Employer VAT1
char vat2[EID_MAX_VAT2_LEN + 1];	Employer VAT2
char regionalFileNumber[EID_MAX_REGIONALFILENUMBER_LEN + 1];	Regional file number
char brexitMention1[EID_MAX_BREXITMENTION1_LEN + 1];	BREXIT
char brexitMention2[EID_MAX_BREXITMENTION2_LEN + 1];	BREXIT

## 1.2.7 tagEidIdentityExA Structure

Identity information stored on EID card - ANSI version

**C++**

```

struct tagEidIdentityExA {
    WORD nSize;
    WORD nVersion;
    char cardNumber[EID_MAX_CARD_NUMBER_LEN + 1];
    char chipNumber[EID_MAX_CHIP_NUMBER_LEN + 1];
    char validityDateBegin[EID_MAX_DATE_BEGIN_LEN + 1];
    char validityDateEnd[EID_MAX_DATE_END_LEN + 1];
    char municipality[EID_MAX_DELIVERY_MUNICIPALITY_LEN + 1];
    char nationalNumber[EID_MAX_NATIONAL_NUMBER_LEN + 1];
    char name[EID_MAX_NAME_LEN + 1];
    char firstName1[EID_MAX_FIRST_NAME1_LEN + 1];
    char firstName2[EID_MAX_FIRST_NAME2_LEN + 1];
    char nationality[EID_MAX_NATIONALITY_LEN + 1];
    char birthLocation[EID_MAX_BIRTHPLACE_LEN + 1];
    char birthDate[EID_MAX_BIRTHDATE_LEN + 1];
    char sex[EID_MAX_SEX_LEN + 1];
    char nobleCondition[EID_MAX_NOBLE_CONDITION_LEN + 1];
    int documentType;
    BOOL whiteCane;
    BOOL yellowCane;
    BOOL extendedMinority;
    char duplicate[EID_MAX_DUPLICATE_LEN + 1];
    char specialOrganization[EID_MAX_SPECIALORGANIZATION_LEN + 1];
    BOOL memberOfFamily;
    char dateAndCountryOfProtection[EID_MAX_DATEANDCOUNTRYOFPROTECTION_LEN + 1];
    char workPermitType[EID_MAX_WORKPERMITTYPE_LEN + 1];
    char vat1[EID_MAX_VAT1_LEN + 1];
    char vat2[EID_MAX_VAT2_LEN + 1];
    char regionalFileNumber[EID_MAX_REGIONALFILENUMBER_LEN + 1];
    char brexitMention1[EID_MAX_BREXITMENTION1_LEN + 1];
    char brexitMention2[EID_MAX_BREXITMENTION2_LEN + 1];
};

```

**File**

**File:** CardStructures.h (see page 168)

**Members**

Members	Description
WORD nSize;	The size of the structure
WORD nVersion;	The version of the structure
char cardNumber[EID_MAX_CARD_NUMBER_LEN + 1];	Electronic ID card number
char chipNumber[EID_MAX_CHIP_NUMBER_LEN + 1];	Electronic ID card physical chip number
char validityDateBegin[EID_MAX_DATE_BEGIN_LEN + 1];	Card validity start date
char validityDateEnd[EID_MAX_DATE_END_LEN + 1];	Card validity end date
char municipality[EID_MAX_DELIVERY_MUNICIPALITY_LEN + 1];	Card delivery municipality
char nationalNumber[EID_MAX_NATIONAL_NUMBER_LEN + 1];	National number
char name[EID_MAX_NAME_LEN + 1];	Surname
char firstName1[EID_MAX_FIRST_NAME1_LEN + 1];	First name (2 first given names)
char firstName2[EID_MAX_FIRST_NAME2_LEN + 1];	First name part 2 (first letter of the 3rd given name).
char nationality[EID_MAX_NATIONALITY_LEN + 1];	Nationality
char birthLocation[EID_MAX_BIRTHPLACE_LEN + 1];	Birth location
char birthDate[EID_MAX_BIRTHDATE_LEN + 1];	Birth date
char sex[EID_MAX_SEX_LEN + 1];	Sex
char nobleCondition[EID_MAX_NOBLE_CONDITION_LEN + 1];	Noble condition
int documentType;	Document type code (Belgian citizen card, Kids Card, Foreigner card)
BOOL whiteCane;	White cane (blind people)
BOOL yellowCane;	Yellow cane (partially sighted people)
BOOL extendedMinority;	Extended minority
char duplicate[EID_MAX_DUPLICATE_LEN + 1];	Duplicata
char specialOrganization[EID_MAX_SPECIALORGANIZATION_LEN + 1];	Special Organization
BOOL memberOfFamily;	Member of family
char dateAndCountryOfProtection[EID_MAX_DATEANDCOUNTRYOFPROTECTION_LEN + 1];	Date and country of protection
char workPermitType[EID_MAX_WORKPERMITTYPE_LEN + 1];	Work Permit type
char vat1[EID_MAX_VAT1_LEN + 1];	Employer VAT1
char vat2[EID_MAX_VAT2_LEN + 1];	Employer VAT2
char regionalFileNumber[EID_MAX_REGIONALFILENUMBER_LEN + 1];	Regional file number
char brexitMention1[EID_MAX_BREXITMENTION1_LEN + 1];	BREXIT
char brexitMention2[EID_MAX_BREXITMENTION2_LEN + 1];	BREXIT

## 1.2.8 tagEidIdentityExW Structure

Identity information stored on EID card - UNICODE version

**C++**

```

struct tagEidIdentityExW {
    WORD nSize;
    WORD nVersion;
    WCHAR cardNumber[EID_MAX_CARD_NUMBER_LEN + 1];
    WCHAR chipNumber[EID_MAX_CHIP_NUMBER_LEN + 1];
    WCHAR validityDateBegin[EID_MAX_DATE_BEGIN_LEN + 1];
    WCHAR validityDateEnd[EID_MAX_DATE_END_LEN + 1];
    WCHAR municipality[EID_MAX_DELIVERY_MUNICIPALITY_LEN + 1];
    WCHAR nationalNumber[EID_MAX_NATIONAL_NUMBER_LEN + 1];
    WCHAR name[EID_MAX_NAME_LEN + 1];
    WCHAR firstName1[EID_MAX_FIRST_NAME1_LEN + 1];
    WCHAR firstName2[EID_MAX_FIRST_NAME2_LEN + 1];
    WCHAR nationality[EID_MAX_NATIONALITY_LEN + 1];
    WCHAR birthLocation[EID_MAX_BIRTHPLACE_LEN + 1];
    WCHAR birthDate[EID_MAX_BIRTHDATE_LEN + 1];
    WCHAR sex[EID_MAX_SEX_LEN + 1];
    WCHAR nobleCondition[EID_MAX_NOBLE_CONDITION_LEN + 1];
    int documentType;
    BOOL whiteCane;
    BOOL yellowCane;
    BOOL extendedMinority;
    WCHAR duplicate[EID_MAX_DUPLICATE_LEN + 1];
    WCHAR specialOrganization[EID_MAX_SPECIALORGANIZATION_LEN + 1];
    BOOL memberOfFamily;
    WCHAR dateAndCountryOfProtection[EID_MAX_DATEANDCOUNTRYOFPROTECTION_LEN + 1];
    WCHAR workPermitType[EID_MAX_WORKPERMITTYPE_LEN + 1];
    WCHAR vat1[EID_MAX_VAT1_LEN + 1];
    WCHAR vat2[EID_MAX_VAT2_LEN + 1];
    WCHAR regionalFileNumber[EID_MAX_REGIONALFILENUMBER_LEN + 1];
    WCHAR brexitMention1[EID_MAX_BREXITMENTION1_LEN + 1];
    WCHAR brexitMention2[EID_MAX_BREXITMENTION2_LEN + 1];
};

```

**File**

**File:** CardStructures.h (see page 168)

**Members**

Members	Description
WORD nSize;	The size of the structure
WORD nVersion;	The version of the structure
WCHAR cardNumber[EID_MAX_CARD_NUMBER_LEN + 1];	Electronic ID card number
WCHAR chipNumber[EID_MAX_CHIP_NUMBER_LEN + 1];	Electronic ID card physical chip number
WCHAR validityDateBegin[EID_MAX_DATE_BEGIN_LEN + 1];	Card validity start date
WCHAR validityDateEnd[EID_MAX_DATE_END_LEN + 1];	Card validity end date
WCHAR municipality[EID_MAX_DELIVERY_MUNICIPALITY_LEN + 1];	Card delivery municipality
WCHAR nationalNumber[EID_MAX_NATIONAL_NUMBER_LEN + 1];	National number
WCHAR name[EID_MAX_NAME_LEN + 1];	Surname
WCHAR firstName1[EID_MAX_FIRST_NAME1_LEN + 1];	First name (2 first given names)
WCHAR firstName2[EID_MAX_FIRST_NAME2_LEN + 1];	First name part 2 (first letter of the 3rd given name).
WCHAR nationality[EID_MAX_NATIONALITY_LEN + 1];	Nationality
WCHAR birthLocation[EID_MAX_BIRTHPLACE_LEN + 1];	Birth location
WCHAR birthDate[EID_MAX_BIRTHDATE_LEN + 1];	Birth date
WCHAR sex[EID_MAX_SEX_LEN + 1];	Sex

WCHAR nobleCondition[EID_MAX_NOBLE_CONDITION_LEN + 1];	Noble condition
int documentType;	Document type code (Belgian citizen card, Kids Card, Foreigner card)
BOOL whiteCane;	White cane (blind people)
BOOL yellowCane;	Yellow cane (partially sighted people)
BOOL extendedMinority;	Extended minority
WCHAR duplicate[EID_MAX_DUPLICATE_LEN + 1];	Duplicata
WCHAR specialOrganization[EID_MAX_SPECIALORGANIZATION_LEN + 1];	Special Organization
BOOL memberOfFamily;	Member of family
WCHAR dateAndCountryOfProtection[EID_MAX_DATEANDCOUNTRYOFPROTECTION_LEN + 1];	Date and country of protection
WCHAR workPermitType[EID_MAX_WORKPERMITTYPE_LEN + 1];	Work permit type
WCHAR vat1[EID_MAX_VAT1_LEN + 1];	Employer VAT1
WCHAR vat2[EID_MAX_VAT2_LEN + 1];	Employer VAT2
WCHAR regionalFileNumber[EID_MAX_REGIONALFILENUMBER_LEN + 1];	Regional file number
WCHAR brexitMention1[EID_MAX_BREXITMENTION1_LEN + 1];	BREXIT
WCHAR brexitMention2[EID_MAX_BREXITMENTION2_LEN + 1];	BREXIT

## 1.2.9 tagEidIdentityW Structure

Identity information stored on EID card - UNICODE version

C++

```

struct tagEidIdentityW {
    WCHAR cardNumber[EID_MAX_CARD_NUMBER_LEN+1];
    WCHAR chipNumber[EID_MAX_CHIP_NUMBER_LEN+1];
    WCHAR validityDateBegin[EID_MAX_DATE_BEGIN_LEN+1];
    WCHAR validityDateEnd[EID_MAX_DATE_END_LEN + 1];
    WCHAR municipality[EID_MAX_DELIVERY_MUNICIPALITY_LEN+1];
    WCHAR nationalNumber[EID_MAX_NATIONAL_NUMBER_LEN+1];
    WCHAR name[EID_MAX_NAME_LEN+1];
    WCHAR firstName1[EID_MAX_FIRST_NAME1_LEN+1];
    WCHAR firstName2[EID_MAX_FIRST_NAME2_LEN+1];
    WCHAR nationality[EID_MAX_NATIONALITY_LEN+1];
    WCHAR birthLocation[EID_MAX_BIRTHPLACE_LEN+1];
    WCHAR birthDate[EID_MAX_BIRTHDATE_LEN+1];
    WCHAR sex[EID_MAX_SEX_LEN+1];
    WCHAR nobleCondition[EID_MAX_NOBLE_CONDITION_LEN+1];
    int documentType;
    BOOL whiteCane;
    BOOL yellowCane;
    BOOL extendedMinority;
    WCHAR duplicate[EID_MAX_DUPLICATE_LEN + 1];
    WCHAR specialOrganization[EID_MAX_SPECIALORGANIZATION_LEN + 1];
    BOOL memberOfFamily;
    WCHAR dateAndCountryOfProtection[EID_MAX_DATEANDCOUNTRYOFPROTECTION_LEN + 1];
    WCHAR workPermitType[EID_MAX_WORKPERMITTYPE_LEN + 1];
    WCHAR vat1[EID_MAX_VAT1_LEN + 1];
    WCHAR vat2[EID_MAX_VAT2_LEN + 1];
    WCHAR regionalFileNumber[EID_MAX_REGIONALFILENUMBER_LEN + 1];
    WCHAR brexitMention1[EID_MAX_BREXITMENTION1_LEN + 1];
    WCHAR brexitMention2[EID_MAX_BREXITMENTION2_LEN + 1];
};

```

**File**

**File:** CardStructures.h (see page 168)

**Members**

Members	Description
WCHAR cardNumber[EID_MAX_CARD_NUMBER_LEN+1];	Electronic ID card number
WCHAR chipNumber[EID_MAX_CHIP_NUMBER_LEN+1];	Electronic ID card physical chip number
WCHAR validityDateBegin[EID_MAX_DATE_BEGIN_LEN+1];	Card validity start date
WCHAR validityDateEnd[EID_MAX_DATE_END_LEN + 1];	Card validity end date
WCHAR municipality[EID_MAX_DELIVERY_MUNICIPALITY_LEN+1];	Card delivery municipality
WCHAR nationalNumber[EID_MAX_NATIONAL_NUMBER_LEN+1];	National number
WCHAR name[EID_MAX_NAME_LEN+1];	Surname
WCHAR firstName1[EID_MAX_FIRST_NAME1_LEN+1];	First name (2 first given names)
WCHAR firstName2[EID_MAX_FIRST_NAME2_LEN+1];	First name part 2 (first letter of the 3rd given name).
WCHAR nationality[EID_MAX_NATIONALITY_LEN+1];	Nationality
WCHAR birthLocation[EID_MAX_BIRTHPLACE_LEN+1];	Birth location
WCHAR birthDate[EID_MAX_BIRTHDATE_LEN+1];	Birth date
WCHAR sex[EID_MAX_SEX_LEN+1];	Sex
WCHAR nobleCondition[EID_MAX_NOBLE_CONDITION_LEN+1];	Noble condition
int documentType;	Document type code (Belgian citizen card, Kids Card, Foreigner card)
BOOL whiteCane;	White cane (blind people)
BOOL yellowCane;	Yellow cane (partially sighted people)
BOOL extendedMinority;	Extended minority
WCHAR duplicate[EID_MAX_DUPLICATE_LEN + 1];	Duplicata
WCHAR specialOrganization[EID_MAX_SPECIALORGANIZATION_LEN + 1];	Special Organization
BOOL memberOfFamily;	Member of family
WCHAR dateAndCountryOfProtection[EID_MAX_DATEANDCOUNTRYOFPROTECTION_LEN + 1];	Date and country of protection
WCHAR workPermitType[EID_MAX_WORKPERMITTYPE_LEN + 1];	Work permit type
WCHAR vat1[EID_MAX_VAT1_LEN + 1];	Employer VAT1
WCHAR vat2[EID_MAX_VAT2_LEN + 1];	Employer VAT2
WCHAR regionalFileNumber[EID_MAX_REGIONALFILENUMBER_LEN + 1];	Regional file number
WCHAR brexitMention1[EID_MAX_BREXITMENTION1_LEN + 1];	BREXIT
WCHAR brexitMention2[EID_MAX_BREXITMENTION2_LEN + 1];	BREXIT

## 1.2.10 tagEidPicture Structure

Raw picture data from EID card

**C++**

```
struct tagEidPicture {
    BYTE picture[EID_MAX_PICTURE_LEN+1];
    int pictureLength;
};
```

**File**

File: CardStructures.h (see page 168)

**Members**

Members	Description
BYTE picture[EID_MAX_PICTURE_LEN+1];	Picture raw data buffer
int pictureLength;	Picture raw data buffer length

## 1.2.11 tagSISRecordA Structure

Public information stored on Belgian SIS card - ANSI version. The SIS card is the social security card of each Belgian resident (Belgian or foreigner)

**C++**

```
struct tagSISRecordA {
    char Name[SIS_MAX_NAME_LEN + 1];
    char FirstName[SIS_MAX_FIRSTNAMES_LEN + 1];
    char Initial[SIS_MAX_INITIAL_LEN+ 1];
    char Sex[SIS_MAX_SEX_LEN + 1];
    char BirthDate[SIS_FIELD_MAX_BIRTHDATE_LEN + 1];
    char SocialSecurityNumber[SIS_FIELD_MAX_SOCIAL_SECURITY_NUMBER_LEN + 1];
    char CaptureDate[SIS_FIELD_MAX_CAPTUREDATE_LEN + 1];
    char ValidityDateBegin[SIS_FIELD_MAX_VALIDBEGIN_LEN + 1];
    char ValidityDateEnd[SIS_FIELD_MAX_VALIDEND_LEN + 1];
    char CardNumber[SIS_FIELD_MAX_CARDNUMBER_LEN + 1];
    char CardName[SIS_MAX_CARDNAME_LEN +1 ];
};
```

**File**

File: CardStructures.h (see page 168)

**Members**

Members	Description
char Name[SIS_MAX_NAME_LEN + 1];	Family name of the card owner
char FirstName[SIS_MAX_FIRSTNAMES_LEN + 1];	First name of the card owner
char Initial[SIS_MAX_INITIAL_LEN+ 1];	Initial of the card owner
char Sex[SIS_MAX_SEX_LEN + 1];	Sex of the card owner
char BirthDate[SIS_FIELD_MAX_BIRTHDATE_LEN + 1];	Birth date of the card owner
char SocialSecurityNumber[SIS_FIELD_MAX_SOCIAL_SECURITY_NUMBER_LEN + 1];	Social security number of the card owner
char CaptureDate[SIS_FIELD_MAX_CAPTUREDATE_LEN + 1];	Date of issue
char ValidityDateBegin[SIS_FIELD_MAX_VALIDBEGIN_LEN + 1];	Card validity begin date
char ValidityDateEnd[SIS_FIELD_MAX_VALIDEND_LEN + 1];	Card validity end date
char CardNumber[SIS_FIELD_MAX_CARDNUMBER_LEN + 1];	Card number
char CardName[SIS_MAX_CARDNAME_LEN +1 ];	Name of the card

## 1.2.12 tagSISRecordW Structure

Public information stored on Belgian SIS card - UNICODE version. The SIS card is the social security card of each Belgian resident (Belgian or foreigner)

**C++**

```
struct tagSISRecordW {
    WCHAR Name[SIS_MAX_NAME_LEN + 1];
    WCHAR FirstName[SIS_MAX_FIRSTNAMES_LEN + 1];
    WCHAR Initial[SIS_MAX_INITIAL_LEN + 1];
    WCHAR Sex[SIS_MAX_SEX_LEN + 1];
    WCHAR BirthDate[SIS_FIELD_MAX_BIRTHDATE_LEN + 1];
    WCHAR SocialSecurityNumber[SIS_FIELD_MAX_SOCIAL_SECURITY_NUMBER_LEN + 1];
    WCHAR CaptureDate[SIS_FIELD_MAX_CAPTUREDATE_LEN + 1];
    WCHAR ValidityDateBegin[SIS_FIELD_MAX_VALIDBEGIN_LEN + 1];
    WCHAR ValidityDateEnd[SIS_FIELD_MAX_VALIDEND_LEN + 1];
    WCHAR CardNumber[SIS_FIELD_MAX_CARDNUMBER_LEN + 1];
    WCHAR CardName[SIS_MAX_CARDNAME_LEN + 1];
};
```

**File**

File: CardStructures.h (see page 168)

**Members**

Members	Description
WCHAR Name[SIS_MAX_NAME_LEN + 1];	Family name of the card owner
WCHAR FirstName[SIS_MAX_FIRSTNAMES_LEN + 1];	First name of the card owner
WCHAR Initial[SIS_MAX_INITIAL_LEN + 1];	Initial of the card owner
WCHAR Sex[SIS_MAX_SEX_LEN + 1];	Sex of the card owner
WCHAR BirthDate[SIS_FIELD_MAX_BIRTHDATE_LEN + 1];	Birth date of the card owner
WCHAR SocialSecurityNumber[SIS_FIELD_MAX_SOCIAL_SECURITY_NUMBER_LEN + 1];	Social security number of the card owner
WCHAR CaptureDate[SIS_FIELD_MAX_CAPTUREDATE_LEN + 1];	Date of issue
WCHAR ValidityDateBegin[SIS_FIELD_MAX_VALIDBEGIN_LEN + 1];	Card validity begin date
WCHAR ValidityDateEnd[SIS_FIELD_MAX_VALIDEND_LEN + 1];	Card validity end date
WCHAR CardNumber[SIS_FIELD_MAX_CARDNUMBER_LEN + 1];	Card number
WCHAR CardName[SIS_MAX_CARDNAME_LEN + 1];	Name of the card

## 1.2.13 CardEventType Enumeration

The type of the reader event

**C++**

```
typedef enum tagCardEventType {
    ewtUnknownEvent,
    ewtCardInsert,
    ewtCardRemove,
    ewtReadersChange
} CardEventType;
```

**File**

File: CardEvents.h (see page 168)



**Members**

Members	Description
ewtUnknownEvent	Unknown event
ewtCardInsert	The card was inserted in the reader
ewtCardRemove	The card was removed from the reader
ewtReadersChange	The readers list changed

## 1.2.14 EidAddressA Structure

EID address information, stored on the card - ANSI version

**C++**

```
typedef struct tagEidAddressA {  
    char street[EID_MAX_STREET_LEN+1];  
    char zip[EID_MAX_ZIP_LEN+1];  
    char municipality[EID_MAX_MUNICIPALITY_LEN+1];  
} EidAddressA, * PEidAddressA;
```

**File**

File: CardStructures.h (see page 168)

**Members**

Members	Description
char street[EID_MAX_STREET_LEN+1];	Street name
char zip[EID_MAX_ZIP_LEN+1];	ZIP code
char municipality[EID_MAX_MUNICIPALITY_LEN+1];	Municipality

## 1.2.15 EidAddressW Structure

EID address information, stored on the card - UNICODE version

**C++**

```
typedef struct tagEidAddressW {  
    WCHAR street[EID_MAX_STREET_LEN+1];  
    WCHAR zip[EID_MAX_ZIP_LEN+1];  
    WCHAR municipality[EID_MAX_MUNICIPALITY_LEN+1];  
} EidAddressW, * PEidAddressW;
```

**File**

File: CardStructures.h (see page 168)

**Members**

Members	Description
WCHAR street[EID_MAX_STREET_LEN+1];	Street name
WCHAR zip[EID_MAX_ZIP_LEN+1];	ZIP code
WCHAR municipality[EID_MAX_MUNICIPALITY_LEN+1];	Municipality

## 1.2.16 EidCertificate Structure

Certificate, stored on EID card

**C++**

```
typedef struct tagEidCertificate {
    BYTE certificate[EID_MAX_CERT_LEN+1];
    int certificateLength;
} EidCertificate, * PEidCertificate;
```

**File**

File: CardStructures.h (see page 168)

**Members**

Members	Description
BYTE certificate[EID_MAX_CERT_LEN+1];	Certificate raw data buffer
int certificateLength;	Certificate data length

## 1.2.17 EidIdentityA Structure

Identity information stored on EID card - ANSI version

**C++**

```
typedef struct tagEidIdentityA {
    char cardNumber[EID_MAX_CARD_NUMBER_LEN+1];
    char chipNumber[EID_MAX_CHIP_NUMBER_LEN+1];
    char validityDateBegin[EID_MAX_DATE_BEGIN_LEN+1];
    char validityDateEnd[EID_MAX_DATE_END_LEN +1];
    char municipality[EID_MAX_DELIVERY_MUNICIPALITY_LEN+1];
    char nationalNumber[EID_MAX_NATIONAL_NUMBER_LEN+1];
    char name[EID_MAX_NAME_LEN+1];
    char firstName1[EID_MAX_FIRST_NAME1_LEN+1];
    char firstName2[EID_MAX_FIRST_NAME2_LEN+1];
    char nationality[EID_MAX_NATIONALITY_LEN+1];
    char birthLocation[EID_MAX_BIRTHPLACE_LEN+1];
    char birthDate[EID_MAX_BIRTHDATE_LEN+1];
    char sex[EID_MAX_SEX_LEN+1];
    char nobleCondition[EID_MAX_NOBLE_CONDITION_LEN+1];
    int documentType;
    BOOL whiteCane;
    BOOL yellowCane;
    BOOL extendedMinority;
    char duplicate[EID_MAX_DUPLICATE_LEN + 1];
    char specialOrganization[EID_MAX_SPECIALORGANIZATION_LEN + 1];
    BOOL memberOfFamily;
    char dateAndCountryOfProtection[EID_MAX_DATEANDCOUNTRYOFPROTECTION_LEN + 1];
    char workPermitType[EID_MAX_WORKPERMITTYPE_LEN + 1];
    char vat1[EID_MAX_VAT1_LEN + 1];
    char vat2[EID_MAX_VAT2_LEN + 1];
    char regionalFileNumber[EID_MAX_REGIONALFILENUMBER_LEN + 1];
    char brexitMention1[EID_MAX_BREXITMENTION1_LEN + 1];
    char brexitMention2[EID_MAX_BREXITMENTION2_LEN + 1];
} EidIdentityA, * PEidIdentityA;
```

**File**

File: CardStructures.h (see page 168)

**Members**

Members	Description
char cardNumber[EID_MAX_CARD_NUMBER_LEN+1];	Electronic ID card number
char chipNumber[EID_MAX_CHIP_NUMBER_LEN+1];	Electronic ID card physical chip number
char validityDateBegin[EID_MAX_DATE_BEGIN_LEN+1];	Card validity start date
char validityDateEnd[EID_MAX_DATE_END_LEN +1];	Card validity end date

char municipality[EID_MAX_DELIVERY_MUNICIPALITY_LEN+1];	Card delivery municipality
char nationalNumber[EID_MAX_NATIONAL_NUMBER_LEN+1];	National number
char name[EID_MAX_NAME_LEN+1];	Surname
char firstName1[EID_MAX_FIRST_NAME1_LEN+1];	First name (2 first given names)
char firstName2[EID_MAX_FIRST_NAME2_LEN+1];	First name part 2 (first letter of the 3rd given name).
char nationality[EID_MAX_NATIONALITY_LEN+1];	Nationality
char birthLocation[EID_MAX_BIRTHPLACE_LEN+1];	Birth location
char birthDate[EID_MAX_BIRTHDATE_LEN+1];	Birth date
char sex[EID_MAX_SEX_LEN+1];	Sex
char nobleCondition[EID_MAX_NOBLE_CONDITION_LEN+1];	Noble condition
int documentType;	Document type code (Belgian citizen card, Kids Card, Foreigner card)
BOOL whiteCane;	White cane (blind people)
BOOL yellowCane;	Yellow cane (partially sighted people)
BOOL extendedMinority;	Extended minority
char duplicate[EID_MAX_DUPLICATE_LEN + 1];	Duplicata
char specialOrganization[EID_MAX_SPECIALORGANIZATION_LEN + 1];	Special Organization
BOOL memberOfFamily;	Member of family
char dateAndCountryOfProtection[EID_MAX_DATEANDCOUNTRYOFPROTECTION_LEN + 1];	Date and country of protection
char workPermitType[EID_MAX_WORKPERMITTYPE_LEN + 1];	Work Permit type
char vat1[EID_MAX_VAT1_LEN + 1];	Employer VAT1
char vat2[EID_MAX_VAT2_LEN + 1];	Employer VAT2
char regionalFileNumber[EID_MAX_REGIONALFILENUMBER_LEN + 1];	Regional file number
char brexitMention1[EID_MAX_BREXITMENTION1_LEN + 1];	BREXIT
char brexitMention2[EID_MAX_BREXITMENTION2_LEN + 1];	BREXIT

## 1.2.18 EidIdentityExA Structure

Identity information stored on EID card - ANSI version

**C++**

```
typedef struct tagEidIdentityExA {
    WORD nSize;
    WORD nVersion;
    char cardNumber[EID_MAX_CARD_NUMBER_LEN + 1];
    char chipNumber[EID_MAX_CHIP_NUMBER_LEN + 1];
    char validityDateBegin[EID_MAX_DATE_BEGIN_LEN + 1];
    char validityDateEnd[EID_MAX_DATE_END_LEN + 1];
    char municipality[EID_MAX_DELIVERY_MUNICIPALITY_LEN + 1];
    char nationalNumber[EID_MAX_NATIONAL_NUMBER_LEN + 1];
    char name[EID_MAX_NAME_LEN + 1];
    char firstName1[EID_MAX_FIRST_NAME1_LEN + 1];
    char firstName2[EID_MAX_FIRST_NAME2_LEN + 1];
    char nationality[EID_MAX_NATIONALITY_LEN + 1];
    char birthLocation[EID_MAX_BIRTHPLACE_LEN + 1];
```

```

char birthDate[EID_MAX_BIRTHDATE_LEN + 1];
char sex[EID_MAX_SEX_LEN + 1];
char nobleCondition[EID_MAX_NOBLE_CONDITION_LEN + 1];
int documentType;
BOOL whiteCane;
BOOL yellowCane;
BOOL extendedMinority;
char duplicate[EID_MAX_DUPLICATE_LEN + 1];
char specialOrganization[EID_MAX_SPECIALORGANIZATION_LEN + 1];
BOOL memberOfFamily;
char dateAndCountryOfProtection[EID_MAX_DATEANDCOUNTRYOFPROTECTION_LEN + 1];
char workPermitType[EID_MAX_WORKPERMITTYPE_LEN + 1];
char vat1[EID_MAX_VAT1_LEN + 1];
char vat2[EID_MAX_VAT2_LEN + 1];
char regionalFileNumber[EID_MAX_REGIONALFILENUMBER_LEN + 1];
char brexitMention1[EID_MAX_BREXITMENTION1_LEN + 1];
char brexitMention2[EID_MAX_BREXITMENTION2_LEN + 1];
} EidIdentityExA, * PEidIdentityExA;

```

## File

File: CardStructures.h (see page 168)

## Members

Members	Description
WORD nSize;	The size of the structure
WORD nVersion;	The version of the structure
char cardNumber[EID_MAX_CARD_NUMBER_LEN + 1];	Electronic ID card number
char chipNumber[EID_MAX_CHIP_NUMBER_LEN + 1];	Electronic ID card physical chip number
char validityDateBegin[EID_MAX_DATE_BEGIN_LEN + 1];	Card validity start date
char validityDateEnd[EID_MAX_DATE_END_LEN + 1];	Card validity end date
char municipality[EID_MAX_DELIVERY_MUNICIPALITY_LEN + 1];	Card delivery municipality
char nationalNumber[EID_MAX_NATIONAL_NUMBER_LEN + 1];	National number
char name[EID_MAX_NAME_LEN + 1];	Surname
char firstName1[EID_MAX_FIRST_NAME1_LEN + 1];	First name (2 first given names)
char firstName2[EID_MAX_FIRST_NAME2_LEN + 1];	First name part 2 (first letter of the 3rd given name).
char nationality[EID_MAX_NATIONALITY_LEN + 1];	Nationality
char birthLocation[EID_MAX_BIRTHPLACE_LEN + 1];	Birth location
char birthDate[EID_MAX_BIRTHDATE_LEN + 1];	Birth date
char sex[EID_MAX_SEX_LEN + 1];	Sex
char nobleCondition[EID_MAX_NOBLE_CONDITION_LEN + 1];	Noble condition
int documentType;	Document type code (Belgian citizen card, Kids Card, Foreigner card)
BOOL whiteCane;	White cane (blind people)
BOOL yellowCane;	Yellow cane (partially sighted people)
BOOL extendedMinority;	Extended minority
char duplicate[EID_MAX_DUPLICATE_LEN + 1];	Duplicata
char specialOrganization[EID_MAX_SPECIALORGANIZATION_LEN + 1];	Special Organization
BOOL memberOfFamily;	Member of family

char dateAndCountryOfProtection[EID_MAX_DATEANDCOUNTRYOFPROTECTION_LEN + 1];	Date and country of protection
char workPermitType[EID_MAX_WORKPERMITTYPE_LEN + 1];	Work Permit type
char vat1[EID_MAX_VAT1_LEN + 1];	Employer VAT1
char vat2[EID_MAX_VAT2_LEN + 1];	Employer VAT2
char regionalFileNumber[EID_MAX_REGIONALFILENUMBER_LEN + 1];	Regional file number
char brexitMention1[EID_MAX_BREXITMENTION1_LEN + 1];	BREXIT
char brexitMention2[EID_MAX_BREXITMENTION2_LEN + 1];	BREXIT

## 1.2.19 EidIdentityExW Structure

Identity information stored on EID card - UNICODE version

**C++**

```
typedef struct tagEidIdentityExW {
    WORD nSize;
    WORD nVersion;
    WCHAR cardNumber[EID_MAX_CARD_NUMBER_LEN + 1];
    WCHAR chipNumber[EID_MAX_CHIP_NUMBER_LEN + 1];
    WCHAR validityDateBegin[EID_MAX_DATE_BEGIN_LEN + 1];
    WCHAR validityDateEnd[EID_MAX_DATE_END_LEN + 1];
    WCHAR municipality[EID_MAX_DELIVERY_MUNICIPALITY_LEN + 1];
    WCHAR nationalNumber[EID_MAX_NATIONAL_NUMBER_LEN + 1];
    WCHAR name[EID_MAX_NAME_LEN + 1];
    WCHAR firstName1[EID_MAX_FIRST_NAME1_LEN + 1];
    WCHAR firstName2[EID_MAX_FIRST_NAME2_LEN + 1];
    WCHAR nationality[EID_MAX_NATIONALITY_LEN + 1];
    WCHAR birthLocation[EID_MAX_BIRTHPLACE_LEN + 1];
    WCHAR birthDate[EID_MAX_BIRTHDATE_LEN + 1];
    WCHAR sex[EID_MAX_SEX_LEN + 1];
    WCHAR nobleCondition[EID_MAX_NOBLE_CONDITION_LEN + 1];
    int documentType;
    BOOL whiteCane;
    BOOL yellowCane;
    BOOL extendedMinority;
    WCHAR duplicate[EID_MAX_DUPLICATE_LEN + 1];
    WCHAR specialOrganization[EID_MAX_SPECIALORGANIZATION_LEN + 1];
    BOOL memberOfFamily;
    WCHAR dateAndCountryOfProtection[EID_MAX_DATEANDCOUNTRYOFPROTECTION_LEN + 1];
    WCHAR workPermitType[EID_MAX_WORKPERMITTYPE_LEN + 1];
    WCHAR vat1[EID_MAX_VAT1_LEN + 1];
    WCHAR vat2[EID_MAX_VAT2_LEN + 1];
    WCHAR regionalFileNumber[EID_MAX_REGIONALFILENUMBER_LEN + 1];
    WCHAR brexitMention1[EID_MAX_BREXITMENTION1_LEN + 1];
    WCHAR brexitMention2[EID_MAX_BREXITMENTION2_LEN + 1];
} EidIdentityExW, * PEidIdentityExW;
```

**File**

File: CardStructures.h (see page 168)

**Members**

Members	Description
WORD nSize;	The size of the structure
WORD nVersion;	The version of the structure
WCHAR cardNumber[EID_MAX_CARD_NUMBER_LEN + 1];	Electronic ID card number
WCHAR chipNumber[EID_MAX_CHIP_NUMBER_LEN + 1];	Electronic ID card physical chip number
WCHAR validityDateBegin[EID_MAX_DATE_BEGIN_LEN + 1];	Card validity start date

WCHAR validityDateEnd[EID_MAX_DATE_END_LEN + 1];	Card validity end date
WCHAR municipality[EID_MAX_DELIVERY_MUNICIPALITY_LEN + 1];	Card delivery municipality
WCHAR nationalNumber[EID_MAX_NATIONAL_NUMBER_LEN + 1];	National number
WCHAR name[EID_MAX_NAME_LEN + 1];	Surname
WCHAR firstName1[EID_MAX_FIRST_NAME1_LEN + 1];	First name (2 first given names)
WCHAR firstName2[EID_MAX_FIRST_NAME2_LEN + 1];	First name part 2 (first letter of the 3rd given name).
WCHAR nationality[EID_MAX_NATIONALITY_LEN + 1];	Nationality
WCHAR birthLocation[EID_MAX_BIRTHPLACE_LEN + 1];	Birth location
WCHAR birthDate[EID_MAX_BIRTHDATE_LEN + 1];	Birth date
WCHAR sex[EID_MAX_SEX_LEN + 1];	Sex
WCHAR nobleCondition[EID_MAX_NOBLE_CONDITION_LEN + 1];	Noble condition
int documentType;	Document type code (Belgian citizen card, Kids Card, Foreigner card)
BOOL whiteCane;	White cane (blind people)
BOOL yellowCane;	Yellow cane (partially sighted people)
BOOL extendedMinority;	Extended minority
WCHAR duplicate[EID_MAX_DUPLICATE_LEN + 1];	Duplicata
WCHAR specialOrganization[EID_MAX_SPECIALORGANIZATION_LEN + 1];	Special Organization
BOOL memberOfFamily;	Member of family
WCHAR dateAndCountryOfProtection[EID_MAX_DATEANDCOUNTRYOFPROTECTION_LEN + 1];	Date and country of protection
WCHAR workPermitType[EID_MAX_WORKPERMITTYPE_LEN + 1];	Work permit type
WCHAR vat1[EID_MAX_VAT1_LEN + 1];	Employer VAT1
WCHAR vat2[EID_MAX_VAT2_LEN + 1];	Employer VAT2
WCHAR regionalFileNumber[EID_MAX_REGIONALFILENUMBER_LEN + 1];	Regional file number
WCHAR brexitMention1[EID_MAX_BREXITMENTION1_LEN + 1];	BREXIT
WCHAR brexitMention2[EID_MAX_BREXITMENTION2_LEN + 1];	BREXIT

## 1.2.20 EidIdentityW Structure

Identity information stored on EID card - UNICODE version

**C++**

```
typedef struct tagEidIdentityW {
    WCHAR cardNumber[EID_MAX_CARD_NUMBER_LEN+1];
    WCHAR chipNumber[EID_MAX_CHIP_NUMBER_LEN+1];
    WCHAR validityDateBegin[EID_MAX_DATE_BEGIN_LEN+1];
    WCHAR validityDateEnd[EID_MAX_DATE_END_LEN + 1];
    WCHAR municipality[EID_MAX_DELIVERY_MUNICIPALITY_LEN+1];
    WCHAR nationalNumber[EID_MAX_NATIONAL_NUMBER_LEN+1];
    WCHAR name[EID_MAX_NAME_LEN+1];
    WCHAR firstName1[EID_MAX_FIRST_NAME1_LEN+1];
    WCHAR firstName2[EID_MAX_FIRST_NAME2_LEN+1];
    WCHAR nationality[EID_MAX_NATIONALITY_LEN+1];
}
```

```

WCHAR birthLocation[EID_MAX_BIRTHPLACE_LEN+1];
WCHAR birthDate[EID_MAX_BIRTHDATE_LEN+1];
WCHAR sex[EID_MAX_SEX_LEN+1];
WCHAR nobleCondition[EID_MAX_NOBLE_CONDITION_LEN+1];
int documentType;
BOOL whiteCane;
BOOL yellowCane;
BOOL extendedMinority;
WCHAR duplicate[EID_MAX_DUPLICATE_LEN + 1];
WCHAR specialOrganization[EID_MAX_SPECIALORGANIZATION_LEN + 1];
BOOL memberOfFamily;
WCHAR dateAndCountryOfProtection[EID_MAX_DATEANDCOUNTRYOFPROTECTION_LEN + 1];
WCHAR workPermitType[EID_MAX_WORKPERMITTYPE_LEN + 1];
WCHAR vat1[EID_MAX_VAT1_LEN + 1];
WCHAR vat2[EID_MAX_VAT2_LEN + 1];
WCHAR regionalFileNumber[EID_MAX_REGIONALFILENUMBER_LEN + 1];
WCHAR brexitMention1[EID_MAX_BREXITMENTION1_LEN + 1];
WCHAR brexitMention2[EID_MAX_BREXITMENTION2_LEN + 1];
} EidIdentityW, * PEidIdentityW;

```

## File

File: CardStructures.h (see page 168)

## Members

Members	Description
WCHAR cardNumber[EID_MAX_CARD_NUMBER_LEN+1];	Electronic ID card number
WCHAR chipNumber[EID_MAX_CHIP_NUMBER_LEN+1];	Electronic ID card physical chip number
WCHAR validityDateBegin[EID_MAX_DATE_BEGIN_LEN+1];	Card validity start date
WCHAR validityDateEnd[EID_MAX_DATE_END_LEN + 1];	Card validity end date
WCHAR municipality[EID_MAX_DELIVERY_MUNICIPALITY_LEN+1];	Card delivery municipality
WCHAR nationalNumber[EID_MAX_NATIONAL_NUMBER_LEN+1];	National number
WCHAR name[EID_MAX_NAME_LEN+1];	Surname
WCHAR firstName1[EID_MAX_FIRST_NAME1_LEN+1];	First name (2 first given names)
WCHAR firstName2[EID_MAX_FIRST_NAME2_LEN+1];	First name part 2 (first letter of the 3rd given name).
WCHAR nationality[EID_MAX_NATIONALITY_LEN+1];	Nationality
WCHAR birthLocation[EID_MAX_BIRTHPLACE_LEN+1];	Birth location
WCHAR birthDate[EID_MAX_BIRTHDATE_LEN+1];	Birth date
WCHAR sex[EID_MAX_SEX_LEN+1];	Sex
WCHAR nobleCondition[EID_MAX_NOBLE_CONDITION_LEN+1];	Noble condition
int documentType;	Document type code (Belgian citizen card, Kids Card, Foreigner card)
BOOL whiteCane;	White cane (blind people)
BOOL yellowCane;	Yellow cane (partially sighted people)
BOOL extendedMinority;	Extended minority
WCHAR duplicate[EID_MAX_DUPLICATE_LEN + 1];	Duplicata
WCHAR specialOrganization[EID_MAX_SPECIALORGANIZATION_LEN + 1];	Special Organization
BOOL memberOfFamily;	Member of family

WCHAR dateAndCountryOfProtection[EID_MAX_DATEANDCOUNTRYOFPROTECTION_LEN + 1];	Date and country of protection
WCHAR workPermitType[EID_MAX_WORKPERMITTYPE_LEN + 1];	Work permit type
WCHAR vat1[EID_MAX_VAT1_LEN + 1];	Employer VAT1
WCHAR vat2[EID_MAX_VAT2_LEN + 1];	Employer VAT2
WCHAR regionalFileNumber[EID_MAX_REGIONALFILENUMBER_LEN + 1];	Regional file number
WCHAR brexitMention1[EID_MAX_BREXITMENTION1_LEN + 1];	BREXIT
WCHAR brexitMention2[EID_MAX_BREXITMENTION2_LEN + 1];	BREXIT

## 1.2.21 EidPicture Structure

Raw picture data from EID card

**C++**

```
typedef struct tagEidPicture {
    BYTE picture[EID_MAX_PICTURE_LEN+1];
    int pictureLength;
} EidPicture, * PeidPicture;
```

**File**

File: CardStructures.h (see page 168)

**Members**

Members	Description
BYTE picture[EID_MAX_PICTURE_LEN+1];	Picture raw data buffer
int pictureLength;	Picture raw data buffer length

## 1.2.22 ErrorInformation Structure

Information about error when operation with the card or certificate is performed

**C++**

```
typedef struct structErrorInformation {
    int Code;
    WCHAR Description[ERROR_MAX_DESCRIPTION + 1];
} ErrorInformation, * PErrorInformation;
```

**File**

File: CardStructures.h (see page 168)

## 1.2.23 PEidAddressA Structure

EID address information, stored on the card - ANSI version

**C++**

```
typedef struct tagEidAddressA {
    char street[EID_MAX_STREET_LEN+1];
    char zip[EID_MAX_ZIP_LEN+1];
    char municipality[EID_MAX_MUNICIPALITY_LEN+1];
} EidAddressA, * PEidAddressA;
```



**File****File:** CardStructures.h (see page 168)**Members**

Members	Description
char street[EID_MAX_STREET_LEN+1];	Street name
char zip[EID_MAX_ZIP_LEN+1];	ZIP code
char municipality[EID_MAX_MUNICIPALITY_LEN+1];	Municipality

## 1.2.24 PEidAddressW Structure

EID address information, stored on the card - UNICODE version

**C++**

```
typedef struct tagEidAddressW {  
    WCHAR street[EID_MAX_STREET_LEN+1];  
    WCHAR zip[EID_MAX_ZIP_LEN+1];  
    WCHAR municipality[EID_MAX_MUNICIPALITY_LEN+1];  
} EidAddressW, * PEidAddressW;
```

**File****File:** CardStructures.h (see page 168)**Members**

Members	Description
WCHAR street[EID_MAX_STREET_LEN+1];	Street name
WCHAR zip[EID_MAX_ZIP_LEN+1];	ZIP code
WCHAR municipality[EID_MAX_MUNICIPALITY_LEN+1];	Municipality

## 1.2.25 PEidCertificate Structure

Certificate, stored on EID card

**C++**

```
typedef struct tagEidCertificate {  
    BYTE certificate[EID_MAX_CERT_LEN+1];  
    int certificateLength;  
} EidCertificate, * PEidCertificate;
```

**File****File:** CardStructures.h (see page 168)**Members**

Members	Description
BYTE certificate[EID_MAX_CERT_LEN+1];	Certificate raw data buffer
int certificateLength;	Certificate data length

## 1.2.26 PEidIdentityA Structure

Identity information stored on EID card - ANSI version

**C++**

```

typedef struct tagEidIdentityA {
    char cardNumber[EID_MAX_CARD_NUMBER_LEN+1];
    char chipNumber[EID_MAX_CHIP_NUMBER_LEN+1];
    char validityDateBegin[EID_MAX_DATE_BEGIN_LEN+1];
    char validityDateEnd[EID_MAX_DATE_END_LEN +1];
    char municipality[EID_MAX_DELIVERY_MUNICIPALITY_LEN+1];
    char nationalNumber[EID_MAX_NATIONAL_NUMBER_LEN+1];
    char name[EID_MAX_NAME_LEN+1];
    char firstName1[EID_MAX_FIRST_NAME1_LEN+1];
    char firstName2[EID_MAX_FIRST_NAME2_LEN+1];
    char nationality[EID_MAX_NATIONALITY_LEN+1];
    char birthLocation[EID_MAX_BIRTHPLACE_LEN+1];
    char birthDate[EID_MAX_BIRTHDATE_LEN+1];
    char sex[EID_MAX_SEX_LEN+1];
    char nobleCondition[EID_MAX_NOBLE_CONDITION_LEN+1];
    int documentType;
    BOOL whiteCane;
    BOOL yellowCane;
    BOOL extendedMinority;
    char duplicate[EID_MAX_DUPLICATE_LEN + 1];
    char specialOrganization[EID_MAX_SPECIALORGANIZATION_LEN + 1];
    BOOL memberOfFamily;
    char dateAndCountryOfProtection[EID_MAX_DATEANDCOUNTRYOFPROTECTION_LEN + 1];
    char workPermitType[EID_MAX_WORKPERMITTYPE_LEN + 1];
    char vat1[EID_MAX_VAT1_LEN + 1];
    char vat2[EID_MAX_VAT2_LEN + 1];
    char regionalFileNumber[EID_MAX_REGIONALFILENUMBER_LEN + 1];
    char brexitMention1[EID_MAX_BREXITMENTION1_LEN + 1];
    char brexitMention2[EID_MAX_BREXITMENTION2_LEN + 1];
} EidIdentityA, * PEidIdentityA;

```

**File**

File: CardStructures.h (see page 168)

**Members**

Members	Description
char cardNumber[EID_MAX_CARD_NUMBER_LEN+1];	Electronic ID card number
char chipNumber[EID_MAX_CHIP_NUMBER_LEN+1];	Electronic ID card physical chip number
char validityDateBegin[EID_MAX_DATE_BEGIN_LEN+1];	Card validity start date
char validityDateEnd[EID_MAX_DATE_END_LEN +1];	Card validity end date
char municipality[EID_MAX_DELIVERY_MUNICIPALITY_LEN+1];	Card delivery municipality
char nationalNumber[EID_MAX_NATIONAL_NUMBER_LEN+1];	National number
char name[EID_MAX_NAME_LEN+1];	Surname
char firstName1[EID_MAX_FIRST_NAME1_LEN+1];	First name (2 first given names)
char firstName2[EID_MAX_FIRST_NAME2_LEN+1];	First name part 2 (first letter of the 3rd given name).
char nationality[EID_MAX_NATIONALITY_LEN+1];	Nationality
char birthLocation[EID_MAX_BIRTHPLACE_LEN+1];	Birth location
char birthDate[EID_MAX_BIRTHDATE_LEN+1];	Birth date
char sex[EID_MAX_SEX_LEN+1];	Sex
char nobleCondition[EID_MAX_NOBLE_CONDITION_LEN+1];	Noble condition
int documentType;	Document type code (Belgian citizen card, Kids Card, Foreigner card)
BOOL whiteCane;	White cane (blind people)

BOOL yellowCane;	Yellow cane (partially sighted people)
BOOL extendedMinority;	Extended minority
char duplicate[EID_MAX_DUPLICATE_LEN + 1];	Duplicata
char specialOrganization[EID_MAX_SPECIALORGANIZATION_LEN + 1];	Special Organization
BOOL memberOfFamily;	Member of family
char dateAndCountryOfProtection[EID_MAX_DATEANDCOUNTRYOFPROTECTION_LEN + 1];	Date and country of protection
char workPermitType[EID_MAX_WORKPERMITTYPE_LEN + 1];	Work Permit type
char vat1[EID_MAX_VAT1_LEN + 1];	Employer VAT1
char vat2[EID_MAX_VAT2_LEN + 1];	Employer VAT2
char regionalFileNumber[EID_MAX_REGIONALFILENUMBER_LEN + 1];	Regional file number
char brexitMention1[EID_MAX_BREXITMENTION1_LEN + 1];	BREXIT
char brexitMention2[EID_MAX_BREXITMENTION2_LEN + 1];	BREXIT

## 1.2.27 PEidIdentityExA Structure

Identity information stored on EID card - ANSI version

**C++**

```
typedef struct tagEidIdentityExA {
    WORD nSize;
    WORD nVersion;
    char cardNumber[EID_MAX_CARD_NUMBER_LEN + 1];
    char chipNumber[EID_MAX_CHIP_NUMBER_LEN + 1];
    char validityDateBegin[EID_MAX_DATE_BEGIN_LEN + 1];
    char validityDateEnd[EID_MAX_DATE_END_LEN + 1];
    char municipality[EID_MAX_DELIVERY_MUNICIPALITY_LEN + 1];
    char nationalNumber[EID_MAX_NATIONAL_NUMBER_LEN + 1];
    char name[EID_MAX_NAME_LEN + 1];
    char firstName1[EID_MAX_FIRST_NAME1_LEN + 1];
    char firstName2[EID_MAX_FIRST_NAME2_LEN + 1];
    char nationality[EID_MAX_NATIONALITY_LEN + 1];
    char birthLocation[EID_MAX_BIRTHPLACE_LEN + 1];
    char birthDate[EID_MAX_BIRTHDATE_LEN + 1];
    char sex[EID_MAX_SEX_LEN + 1];
    char nobleCondition[EID_MAX_NOBLE_CONDITION_LEN + 1];
    int documentType;
    BOOL whiteCane;
    BOOL yellowCane;
    BOOL extendedMinority;
    char duplicate[EID_MAX_DUPLICATE_LEN + 1];
    char specialOrganization[EID_MAX_SPECIALORGANIZATION_LEN + 1];
    BOOL memberOfFamily;
    char dateAndCountryOfProtection[EID_MAX_DATEANDCOUNTRYOFPROTECTION_LEN + 1];
    char workPermitType[EID_MAX_WORKPERMITTYPE_LEN + 1];
    char vat1[EID_MAX_VAT1_LEN + 1];
    char vat2[EID_MAX_VAT2_LEN + 1];
    char regionalFileNumber[EID_MAX_REGIONALFILENUMBER_LEN + 1];
    char brexitMention1[EID_MAX_BREXITMENTION1_LEN + 1];
    char brexitMention2[EID_MAX_BREXITMENTION2_LEN + 1];
} EidIdentityExA, * PEidIdentityExA;
```

**File**

**File:** CardStructures.h (see page 168)

**Members**

Members	Description
WORD nSize;	The size of the structure
WORD nVersion;	The version of the structure
char cardNumber[EID_MAX_CARD_NUMBER_LEN + 1];	Electronic ID card number
char chipNumber[EID_MAX_CHIP_NUMBER_LEN + 1];	Electronic ID card physical chip number
char validityDateBegin[EID_MAX_DATE_BEGIN_LEN + 1];	Card validity start date
char validityDateEnd[EID_MAX_DATE_END_LEN + 1];	Card validity end date
char municipality[EID_MAX_DELIVERY_MUNICIPALITY_LEN + 1];	Card delivery municipality
char nationalNumber[EID_MAX_NATIONAL_NUMBER_LEN + 1];	National number
char name[EID_MAX_NAME_LEN + 1];	Surname
char firstName1[EID_MAX_FIRST_NAME1_LEN + 1];	First name (2 first given names)
char firstName2[EID_MAX_FIRST_NAME2_LEN + 1];	First name part 2 (first letter of the 3rd given name).
char nationality[EID_MAX_NATIONALITY_LEN + 1];	Nationality
char birthLocation[EID_MAX_BIRTHPLACE_LEN + 1];	Birth location
char birthDate[EID_MAX_BIRTHDATE_LEN + 1];	Birth date
char sex[EID_MAX_SEX_LEN + 1];	Sex
char nobleCondition[EID_MAX_NOBLE_CONDITION_LEN + 1];	Noble condition
int documentType;	Document type code (Belgian citizen card, Kids Card, Foreigner card)
BOOL whiteCane;	White cane (blind people)
BOOL yellowCane;	Yellow cane (partially sighted people)
BOOL extendedMinority;	Extended minority
char duplicate[EID_MAX_DUPLICATE_LEN + 1];	Duplicata
char specialOrganization[EID_MAX_SPECIALORGANIZATION_LEN + 1];	Special Organization
BOOL memberOfFamily;	Member of family
char dateAndCountryOfProtection[EID_MAX_DATEANDCOUNTRYOFPROTECTION_LEN + 1];	Date and country of protection
char workPermitType[EID_MAX_WORKPERMITTYPE_LEN + 1];	Work Permit type
char vat1[EID_MAX_VAT1_LEN + 1];	Employer VAT1
char vat2[EID_MAX_VAT2_LEN + 1];	Employer VAT2
char regionalFileNumber[EID_MAX_REGIONALFILENUMBER_LEN + 1];	Regional file number
char brexitMention1[EID_MAX_BREXITMENTION1_LEN + 1];	BREXIT
char brexitMention2[EID_MAX_BREXITMENTION2_LEN + 1];	BREXIT

## 1.2.28 PEidIdentityExW Structure

Identity information stored on EID card - UNICODE version

**C++**

```

typedef struct tagEidIdentityExW {
    WORD nSize;
    WORD nVersion;
    WCHAR cardNumber[EID_MAX_CARD_NUMBER_LEN + 1];
    WCHAR chipNumber[EID_MAX_CHIP_NUMBER_LEN + 1];
    WCHAR validityDateBegin[EID_MAX_DATE_BEGIN_LEN + 1];
    WCHAR validityDateEnd[EID_MAX_DATE_END_LEN + 1];
    WCHAR municipality[EID_MAX_DELIVERY_MUNICIPALITY_LEN + 1];
    WCHAR nationalNumber[EID_MAX_NATIONAL_NUMBER_LEN + 1];
    WCHAR name[EID_MAX_NAME_LEN + 1];
    WCHAR firstName1[EID_MAX_FIRST_NAME1_LEN + 1];
    WCHAR firstName2[EID_MAX_FIRST_NAME2_LEN + 1];
    WCHAR nationality[EID_MAX_NATIONALITY_LEN + 1];
    WCHAR birthLocation[EID_MAX_BIRTHPLACE_LEN + 1];
    WCHAR birthDate[EID_MAX_BIRTHDATE_LEN + 1];
    WCHAR sex[EID_MAX_SEX_LEN + 1];
    WCHAR nobleCondition[EID_MAX_NOBLE_CONDITION_LEN + 1];
    int documentType;
    BOOL whiteCane;
    BOOL yellowCane;
    BOOL extendedMinority;
    WCHAR duplicate[EID_MAX_DUPLICATE_LEN + 1];
    WCHAR specialOrganization[EID_MAX_SPECIALORGANIZATION_LEN + 1];
    BOOL memberOfFamily;
    WCHAR dateAndCountryOfProtection[EID_MAX_DATEANDCOUNTRYOFPROTECTION_LEN + 1];
    WCHAR workPermitType[EID_MAX_WORKPERMITTYPE_LEN + 1];
    WCHAR vat1[EID_MAX_VAT1_LEN + 1];
    WCHAR vat2[EID_MAX_VAT2_LEN + 1];
    WCHAR regionalFileNumber[EID_MAX_REGIONALFILENUMBER_LEN + 1];
    WCHAR brexitMention1[EID_MAX_BREXITMENTION1_LEN + 1];
    WCHAR brexitMention2[EID_MAX_BREXITMENTION2_LEN + 1];
} EidIdentityExW, * PEidIdentityExW;

```

**File**

**File:** CardStructures.h (see page 168)

**Members**

Members	Description
WORD nSize;	The size of the structure
WORD nVersion;	The version of the structure
WCHAR cardNumber[EID_MAX_CARD_NUMBER_LEN + 1];	Electronic ID card number
WCHAR chipNumber[EID_MAX_CHIP_NUMBER_LEN + 1];	Electronic ID card physical chip number
WCHAR validityDateBegin[EID_MAX_DATE_BEGIN_LEN + 1];	Card validity start date
WCHAR validityDateEnd[EID_MAX_DATE_END_LEN + 1];	Card validity end date
WCHAR municipality[EID_MAX_DELIVERY_MUNICIPALITY_LEN + 1];	Card delivery municipality
WCHAR nationalNumber[EID_MAX_NATIONAL_NUMBER_LEN + 1];	National number
WCHAR name[EID_MAX_NAME_LEN + 1];	Surname
WCHAR firstName1[EID_MAX_FIRST_NAME1_LEN + 1];	First name (2 first given names)
WCHAR firstName2[EID_MAX_FIRST_NAME2_LEN + 1];	First name part 2 (first letter of the 3rd given name).
WCHAR nationality[EID_MAX_NATIONALITY_LEN + 1];	Nationality
WCHAR birthLocation[EID_MAX_BIRTHPLACE_LEN + 1];	Birth location
WCHAR birthDate[EID_MAX_BIRTHDATE_LEN + 1];	Birth date
WCHAR sex[EID_MAX_SEX_LEN + 1];	Sex

WCHAR nobleCondition[EID_MAX_NOBLE_CONDITION_LEN + 1];	Noble condition
int documentType;	Document type code (Belgian citizen card, Kids Card, Foreigner card)
BOOL whiteCane;	White cane (blind people)
BOOL yellowCane;	Yellow cane (partially sighted people)
BOOL extendedMinority;	Extended minority
WCHAR duplicate[EID_MAX_DUPLICATE_LEN + 1];	Duplicata
WCHAR specialOrganization[EID_MAX_SPECIALORGANIZATION_LEN + 1];	Special Organization
BOOL memberOfFamily;	Member of family
WCHAR dateAndCountryOfProtection[EID_MAX_DATEANDCOUNTRYOFPROTECTION_LEN + 1];	Date and country of protection
WCHAR workPermitType[EID_MAX_WORKPERMITTYPE_LEN + 1];	Work permit type
WCHAR vat1[EID_MAX_VAT1_LEN + 1];	Employer VAT1
WCHAR vat2[EID_MAX_VAT2_LEN + 1];	Employer VAT2
WCHAR regionalFileNumber[EID_MAX_REGIONALFILENUMBER_LEN + 1];	Regional file number
WCHAR brexitMention1[EID_MAX_BREXITMENTION1_LEN + 1];	BREXIT
WCHAR brexitMention2[EID_MAX_BREXITMENTION2_LEN + 1];	BREXIT

## 1.2.29 PEidIdentityW Structure

Identity information stored on EID card - UNICODE version

C++

```
typedef struct tagEidIdentityW {
    WCHAR cardNumber[EID_MAX_CARD_NUMBER_LEN+1];
    WCHAR chipNumber[EID_MAX_CHIP_NUMBER_LEN+1];
    WCHAR validityDateBegin[EID_MAX_DATE_BEGIN_LEN+1];
    WCHAR validityDateEnd[EID_MAX_DATE_END_LEN + 1];
    WCHAR municipality[EID_MAX_DELIVERY_MUNICIPALITY_LEN+1];
    WCHAR nationalNumber[EID_MAX_NATIONAL_NUMBER_LEN+1];
    WCHAR name[EID_MAX_NAME_LEN+1];
    WCHAR firstName1[EID_MAX_FIRST_NAME1_LEN+1];
    WCHAR firstName2[EID_MAX_FIRST_NAME2_LEN+1];
    WCHAR nationality[EID_MAX_NATIONALITY_LEN+1];
    WCHAR birthLocation[EID_MAX_BIRTHPLACE_LEN+1];
    WCHAR birthDate[EID_MAX_BIRTHDATE_LEN+1];
    WCHAR sex[EID_MAX_SEX_LEN+1];
    WCHAR nobleCondition[EID_MAX_NOBLE_CONDITION_LEN+1];
    int documentType;
    BOOL whiteCane;
    BOOL yellowCane;
    BOOL extendedMinority;
    WCHAR duplicate[EID_MAX_DUPLICATE_LEN + 1];
    WCHAR specialOrganization[EID_MAX_SPECIALORGANIZATION_LEN + 1];
    BOOL memberOfFamily;
    WCHAR dateAndCountryOfProtection[EID_MAX_DATEANDCOUNTRYOFPROTECTION_LEN + 1];
    WCHAR workPermitType[EID_MAX_WORKPERMITTYPE_LEN + 1];
    WCHAR vat1[EID_MAX_VAT1_LEN + 1];
    WCHAR vat2[EID_MAX_VAT2_LEN + 1];
    WCHAR regionalFileNumber[EID_MAX_REGIONALFILENUMBER_LEN + 1];
    WCHAR brexitMention1[EID_MAX_BREXITMENTION1_LEN + 1];
    WCHAR brexitMention2[EID_MAX_BREXITMENTION2_LEN + 1];
} EidIdentityW, * PEidIdentityW;
```

**File**

**File:** CardStructures.h (see page 168)

**Members**

Members	Description
WCHAR cardNumber[EID_MAX_CARD_NUMBER_LEN+1];	Electronic ID card number
WCHAR chipNumber[EID_MAX_CHIP_NUMBER_LEN+1];	Electronic ID card physical chip number
WCHAR validityDateBegin[EID_MAX_DATE_BEGIN_LEN+1];	Card validity start date
WCHAR validityDateEnd[EID_MAX_DATE_END_LEN + 1];	Card validity end date
WCHAR municipality[EID_MAX_DELIVERY_MUNICIPALITY_LEN+1];	Card delivery municipality
WCHAR nationalNumber[EID_MAX_NATIONAL_NUMBER_LEN+1];	National number
WCHAR name[EID_MAX_NAME_LEN+1];	Surname
WCHAR firstName1[EID_MAX_FIRST_NAME1_LEN+1];	First name (2 first given names)
WCHAR firstName2[EID_MAX_FIRST_NAME2_LEN+1];	First name part 2 (first letter of the 3rd given name).
WCHAR nationality[EID_MAX_NATIONALITY_LEN+1];	Nationality
WCHAR birthLocation[EID_MAX_BIRTHPLACE_LEN+1];	Birth location
WCHAR birthDate[EID_MAX_BIRTHDATE_LEN+1];	Birth date
WCHAR sex[EID_MAX_SEX_LEN+1];	Sex
WCHAR nobleCondition[EID_MAX_NOBLE_CONDITION_LEN+1];	Noble condition
int documentType;	Document type code (Belgian citizen card, Kids Card, Foreigner card)
BOOL whiteCane;	White cane (blind people)
BOOL yellowCane;	Yellow cane (partially sighted people)
BOOL extendedMinority;	Extended minority
WCHAR duplicate[EID_MAX_DUPLICATE_LEN + 1];	Duplicata
WCHAR specialOrganization[EID_MAX_SPECIALORGANIZATION_LEN + 1];	Special Organization
BOOL memberOfFamily;	Member of family
WCHAR dateAndCountryOfProtection[EID_MAX_DATEANDCOUNTRYOFPROTECTION_LEN + 1];	Date and country of protection
WCHAR workPermitType[EID_MAX_WORKPERMITTYPE_LEN + 1];	Work permit type
WCHAR vat1[EID_MAX_VAT1_LEN + 1];	Employer VAT1
WCHAR vat2[EID_MAX_VAT2_LEN + 1];	Employer VAT2
WCHAR regionalFileNumber[EID_MAX_REGIONALFILENUMBER_LEN + 1];	Regional file number
WCHAR brexitMention1[EID_MAX_BREXITMENTION1_LEN + 1];	BREXIT
WCHAR brexitMention2[EID_MAX_BREXITMENTION2_LEN + 1];	BREXIT

## 1.2.30 PErrorInformation Structure

Information about error when operation with the card or certificate is performed

**C++**

```
typedef struct structErrorInformation {
    int Code;
    WCHAR Description[ERROR_MAX_DESCRIPTION + 1];
} ErrorInformation, * PErrorInformation;
```

**File**

File: CardStructures.h (see page 168)

## 1.2.31 PSISRecordA Structure

Public information stored on Belgian SIS card - ANSI version. The SIS card is the social security card of each Belgian resident (Belgian or foreigner)

**C++**

```
typedef struct tagSISRecordA {
    char Name[SIS_MAX_NAME_LEN + 1];
    char FirstName[SIS_MAX_FIRSTNAMES_LEN + 1];
    char Initial[SIS_MAX_INITIAL_LEN + 1];
    char Sex[SIS_MAX_SEX_LEN + 1];
    char BirthDate[SIS_FIELD_MAX_BIRTHDATE_LEN + 1];
    char SocialSecurityNumber[SIS_FIELD_MAX_SOCIAL_SECURITY_NUMBER_LEN + 1];
    char CaptureDate[SIS_FIELD_MAX_CAPTUREDATE_LEN + 1];
    char ValidityDateBegin[SIS_FIELD_MAX_VALIDBEGIN_LEN + 1];
    char ValidityDateEnd[SIS_FIELD_MAX_VALIDEND_LEN + 1];
    char CardNumber[SIS_FIELD_MAX_CARDNUMBER_LEN + 1];
    char CardName[SIS_MAX_CARDNAME_LEN + 1];
} SISRecordA, * PSISRecordA;
```

**File**

File: CardStructures.h (see page 168)

**Members**

Members	Description
char Name[SIS_MAX_NAME_LEN + 1];	Family name of the card owner
char FirstName[SIS_MAX_FIRSTNAMES_LEN + 1];	First name of the card owner
char Initial[SIS_MAX_INITIAL_LEN + 1];	Initial of the card owner
char Sex[SIS_MAX_SEX_LEN + 1];	Sex of the card owner
char BirthDate[SIS_FIELD_MAX_BIRTHDATE_LEN + 1];	Birth date of the card owner
char SocialSecurityNumber[SIS_FIELD_MAX_SOCIAL_SECURITY_NUMBER_LEN + 1];	Social security number of the card owner
char CaptureDate[SIS_FIELD_MAX_CAPTUREDATE_LEN + 1];	Date of issue
char ValidityDateBegin[SIS_FIELD_MAX_VALIDBEGIN_LEN + 1];	Card validity begin date
char ValidityDateEnd[SIS_FIELD_MAX_VALIDEND_LEN + 1];	Card validity end date
char CardNumber[SIS_FIELD_MAX_CARDNUMBER_LEN + 1];	Card number
char CardName[SIS_MAX_CARDNAME_LEN + 1];	Name of the card

## 1.2.32 PSISRecordW Structure

Public information stored on Belgian SIS card - UNICODE version. The SIS card is the social security card of each Belgian resident (Belgian or foreigner)



**C++**

```
typedef struct tagSISRecordW {
    WCHAR Name[SIS_MAX_NAME_LEN + 1];
    WCHAR FirstName[SIS_MAX_FIRSTNAMES_LEN + 1];
    WCHAR Initial[SIS_MAX_INITIAL_LEN+ 1];
    WCHAR Sex[SIS_MAX_SEX_LEN + 1];
    WCHAR BirthDate[SIS_FIELD_MAX_BIRTHDATE_LEN + 1];
    WCHAR SocialSecurityNumber[SIS_FIELD_MAX_SOCIAL_SECURITY_NUMBER_LEN + 1];
    WCHAR CaptureDate[SIS_FIELD_MAX_CAPTUREDATE_LEN + 1];
    WCHAR ValidityDateBegin[SIS_FIELD_MAX_VALIDBEGIN_LEN + 1];
    WCHAR ValidityDateEnd[SIS_FIELD_MAX_VALIDEND_LEN + 1];
    WCHAR CardNumber[SIS_FIELD_MAX_CARDNUMBER_LEN + 1];
    WCHAR CardName[SIS_MAX_CARDNAME_LEN +1 ];
} SISRecordW, * PSISRecordW;
```

**File**

File: CardStructures.h (see page 168)

**Members**

Members	Description
WCHAR Name[SIS_MAX_NAME_LEN + 1];	Family name of the card owner
WCHAR FirstName[SIS_MAX_FIRSTNAMES_LEN + 1];	First name of the card owner
WCHAR Initial[SIS_MAX_INITIAL_LEN+ 1];	Initial of the card owner
WCHAR Sex[SIS_MAX_SEX_LEN + 1];	Sex of the card owner
WCHAR BirthDate[SIS_FIELD_MAX_BIRTHDATE_LEN + 1];	Birth date of the card owner
WCHAR SocialSecurityNumber[SIS_FIELD_MAX_SOCIAL_SECURITY_NUMBER_LEN + 1];	Social security number of the card owner
WCHAR CaptureDate[SIS_FIELD_MAX_CAPTUREDATE_LEN + 1];	Date of issue
WCHAR ValidityDateBegin[SIS_FIELD_MAX_VALIDBEGIN_LEN + 1];	Card validity begin date
WCHAR ValidityDateEnd[SIS_FIELD_MAX_VALIDEND_LEN + 1];	Card validity end date
WCHAR CardNumber[SIS_FIELD_MAX_CARDNUMBER_LEN + 1];	Card number
WCHAR CardName[SIS_MAX_CARDNAME_LEN +1 ];	Name of the card

## 1.2.33 PeidPicture Structure

Raw picture data from EID card

**C++**

```
typedef struct tagEidPicture {
    BYTE picture[EID_MAX_PICTURE_LEN+1];
    int pictureLength;
} EidPicture, * PeidPicture;
```

**File**

File: CardStructures.h (see page 168)

**Members**

Members	Description
BYTE picture[EID_MAX_PICTURE_LEN+1];	Picture raw data buffer
int pictureLength;	Picture raw data buffer length

## 1.2.34 SISRecordA Structure

Public information stored on Belgian SIS card - ANSI version. The SIS card is the social security card of each Belgian resident (Belgian or foreigner)

**C++**

```
typedef struct tagSISRecordA {
    char Name[SIS_MAX_NAME_LEN + 1];
    char FirstName[SIS_MAX_FIRSTNAMES_LEN + 1];
    char Initial[SIS_MAX_INITIAL_LEN+ 1];
    char Sex[SIS_MAX_SEX_LEN + 1];
    char BirthDate[SIS_FIELD_MAX_BIRTHDATE_LEN + 1];
    char SocialSecurityNumber[SIS_FIELD_MAX_SOCIAL_SECURITY_NUMBER_LEN + 1];
    char CaptureDate[SIS_FIELD_MAX_CAPTUREDATE_LEN + 1];
    char ValidityDateBegin[SIS_FIELD_MAX_VALIDBEGIN_LEN + 1];
    char ValidityDateEnd[SIS_FIELD_MAX_VALIDEND_LEN + 1];
    char CardNumber[SIS_FIELD_MAX_CARDNUMBER_LEN + 1];
    char CardName[SIS_MAX_CARDNAME_LEN +1 ];
} SISRecordA, * PSISRecordA;
```

**File**

File: CardStructures.h (see page 168)

**Members**

Members	Description
char Name[SIS_MAX_NAME_LEN + 1];	Family name of the card owner
char FirstName[SIS_MAX_FIRSTNAMES_LEN + 1];	First name of the card owner
char Initial[SIS_MAX_INITIAL_LEN+ 1];	Initial of the card owner
char Sex[SIS_MAX_SEX_LEN + 1];	Sex of the card owner
char BirthDate[SIS_FIELD_MAX_BIRTHDATE_LEN + 1];	Birth date of the card owner
char SocialSecurityNumber[SIS_FIELD_MAX_SOCIAL_SECURITY_NUMBER_LEN + 1];	Social security number of the card owner
char CaptureDate[SIS_FIELD_MAX_CAPTUREDATE_LEN + 1];	Date of issue
char ValidityDateBegin[SIS_FIELD_MAX_VALIDBEGIN_LEN + 1];	Card validity begin date
char ValidityDateEnd[SIS_FIELD_MAX_VALIDEND_LEN + 1];	Card validity end date
char CardNumber[SIS_FIELD_MAX_CARDNUMBER_LEN + 1];	Card number
char CardName[SIS_MAX_CARDNAME_LEN +1 ];	Name of the card

## 1.2.35 SISRecordW Structure

Public information stored on Belgian SIS card - UNICODE version. The SIS card is the social security card of each Belgian resident (Belgian or foreigner)

**C++**

```
typedef struct tagSISRecordW {
    WCHAR Name[SIS_MAX_NAME_LEN + 1];
    WCHAR FirstName[SIS_MAX_FIRSTNAMES_LEN + 1];
    WCHAR Initial[SIS_MAX_INITIAL_LEN+ 1];
    WCHAR Sex[SIS_MAX_SEX_LEN + 1];
    WCHAR BirthDate[SIS_FIELD_MAX_BIRTHDATE_LEN + 1];
    WCHAR SocialSecurityNumber[SIS_FIELD_MAX_SOCIAL_SECURITY_NUMBER_LEN + 1];
    WCHAR CaptureDate[SIS_FIELD_MAX_CAPTUREDATE_LEN + 1];
    WCHAR ValidityDateBegin[SIS_FIELD_MAX_VALIDBEGIN_LEN + 1];
    WCHAR ValidityDateEnd[SIS_FIELD_MAX_VALIDEND_LEN + 1];
```

```

WCHAR CardNumber[SIS_FIELD_MAX_CARDNUMBER_LEN + 1];
WCHAR CardName[SIS_MAX_CARDNAME_LEN + 1 ];
} SISRecordW, * PSISRecordW;

```

## File

**File:** CardStructures.h ([see page 168](#))

## Members

Members	Description
WCHAR Name[SIS_MAX_NAME_LEN + 1];	Family name of the card owner
WCHAR FirstName[SIS_MAX_FIRSTNAMES_LEN + 1];	First name of the card owner
WCHAR Initial[SIS_MAX_INITIAL_LEN + 1];	Initial of the card owner
WCHAR Sex[SIS_MAX_SEX_LEN + 1];	Sex of the card owner
WCHAR BirthDate[SIS_FIELD_MAX_BIRTHDATE_LEN + 1];	Birth date of the card owner
WCHAR SocialSecurityNumber[SIS_FIELD_MAX_SOCIAL_SECURITY_NUMBER_LEN + 1];	Social security number of the card owner
WCHAR CaptureDate[SIS_FIELD_MAX_CAPTUREDATE_LEN + 1];	Date of issue
WCHAR ValidityDateBegin[SIS_FIELD_MAX_VALIDBEGIN_LEN + 1];	Card validity begin date
WCHAR ValidityDateEnd[SIS_FIELD_MAX_VALIDEND_LEN + 1];	Card validity end date
WCHAR CardNumber[SIS_FIELD_MAX_CARDNUMBER_LEN + 1];	Card number
WCHAR CardName[SIS_MAX_CARDNAME_LEN + 1 ];	Name of the card

# 1.3 Files

The following table lists files in this documentation.

## Files

Name	Description
Algorithms.h ( <a href="#">see page 167</a> )	This is file Algorithms.h.
CardEvents.h ( <a href="#">see page 168</a> )	The definition of the possible card events
CardStructures.h ( <a href="#">see page 168</a> )	The definition of the information storage structures
Encryption.h ( <a href="#">see page 169</a> )	Encryption and decryption operations
FileOperations.h ( <a href="#">see page 170</a> )	Files and folders manipulations
Graphics.h ( <a href="#">see page 171</a> )	This is file Graphics.h.
NationalityConverter.h ( <a href="#">see page 172</a> )	Nationality to ISO code cenvter
Network.h ( <a href="#">see page 172</a> )	
Swelio.h ( <a href="#">see page 173</a> )	Belgian electronic Id card access engine
System.h ( <a href="#">see page 178</a> )	Windows related routines
SystemInfo.h ( <a href="#">see page 179</a> )	Routines for querying information about operating system
quicol.h ( <a href="#">see page 179</a> )	QR Code generator


## 1.3.1 Algorithms.h

This is file Algorithms.h.

## 1.3.2 CardEvents.h

The definition of the possible card events












### Enumerations

	Name	Description
	tagCardEventType ( <a href="#">see page 139</a> )	The type of the reader event
	CardEventType ( <a href="#">see page 148</a> )	The type of the reader event

## 1.3.3 CardStructures.h

The definition of the information storage structures

### Structures

	Name	Description
	structErrorInformation ( <a href="#">see page 139</a> )	Information about error when operation with the card or certificate is performed
	tagEidAddressA ( <a href="#">see page 139</a> )	EID address information, stored on the card - ANSI version
	tagEidAddressW ( <a href="#">see page 140</a> )	EID address information, stored on the card - UNICODE version
	tagEidCertificate ( <a href="#">see page 140</a> )	Certificate, stored on EID card
	tagEidIdentityA ( <a href="#">see page 140</a> )	Identity information stored on EID card - ANSI version
	tagEidIdentityExA ( <a href="#">see page 142</a> )	Identity information stored on EID card - ANSI version
	tagEidIdentityExW ( <a href="#">see page 143</a> )	Identity information stored on EID card - UNICODE version
	tagEidIdentityW ( <a href="#">see page 145</a> )	Identity information stored on EID card - UNICODE version
	tagEidPicture ( <a href="#">see page 146</a> )	Raw picture data from EID card
	tagSISRecordA ( <a href="#">see page 147</a> )	Public information stored on Belgian SIS card - ANSI version. The SIS card is the social security card of each Belgian resident (Belgian or foreigner)
	tagSISRecordW ( <a href="#">see page 148</a> )	Public information stored on Belgian SIS card - UNICODE version. The SIS card is the social security card of each Belgian resident (Belgian or foreigner)
	EidAddressA ( <a href="#">see page 149</a> )	EID address information, stored on the card - ANSI version
	EidAddressW ( <a href="#">see page 149</a> )	EID address information, stored on the card - UNICODE version
	EidCertificate ( <a href="#">see page 149</a> )	Certificate, stored on EID card
	EidIdentityA ( <a href="#">see page 150</a> )	Identity information stored on EID card - ANSI version
	EidIdentityExA ( <a href="#">see page 151</a> )	Identity information stored on EID card - ANSI version
	EidIdentityExW ( <a href="#">see page 153</a> )	Identity information stored on EID card - UNICODE version
	EidIdentityW ( <a href="#">see page 154</a> )	Identity information stored on EID card - UNICODE version
	EidPicture ( <a href="#">see page 156</a> )	Raw picture data from EID card
	ErrorInformation ( <a href="#">see page 156</a> )	Information about error when operation with the card or certificate is performed
	PEidAddressA ( <a href="#">see page 156</a> )	EID address information, stored on the card - ANSI version
	PEidAddressW ( <a href="#">see page 157</a> )	EID address information, stored on the card - UNICODE version
	PEidCertificate ( <a href="#">see page 157</a> )	Certificate, stored on EID card
	PEidIdentityA ( <a href="#">see page 157</a> )	Identity information stored on EID card - ANSI version
	PEidIdentityExA ( <a href="#">see page 159</a> )	Identity information stored on EID card - ANSI version
	PEidIdentityExW ( <a href="#">see page 160</a> )	Identity information stored on EID card - UNICODE version

	PEidIdentityW (see page 162)	Identity information stored on EID card - UNICODE version
	PErrorInformation (see page 163)	Information about error when operation with the card or certificate is performed
	PSISRecordA (see page 164)	Public information stored on Belgian SIS card - ANSI version. The SIS card is the social security card of each Belgian resident (Belgian or foreigner)
	PSISRecordW (see page 164)	Public information stored on Belgian SIS card - UNICODE version. The SIS card is the social security card of each Belgian resident (Belgian or foreigner)
	PeidPicture (see page 165)	Raw picture data from EID card
	SISRecordA (see page 166)	Public information stored on Belgian SIS card - ANSI version. The SIS card is the social security card of each Belgian resident (Belgian or foreigner)
	SISRecordW (see page 166)	Public information stored on Belgian SIS card - UNICODE version. The SIS card is the social security card of each Belgian resident (Belgian or foreigner)

## 1.3.4 Encryption.h

Encryption and decryption operations

### Functions

	Name	Description
◆	CardDecryptFileA (see page 16)	Decrypt file using Belgian Id card
◆	CardDecryptFileW (see page 16)	Decrypt file using Belgian Id card
◆	CardEncryptFileA (see page 16)	Encrypt file using Belgian Id card
◆	CardEncryptFileW (see page 17)	Encrypt file using Belgian Id card
◆	CheckMD5 (see page 22)	Checks the MD5 hash value of the memory buffer
◆	CheckSHA1 (see page 23)	Checks the SHA1 hash value of the memory buffer
◆	CheckSHA256 (see page 23)	Checks the SHA256 hash value of the memory buffer
◆	DecryptFileAESA (see page 30)	Decrypts file using AES algorithm.
◆	DecryptFileAESW (see page 31)	Decrypts file using AES algorithm.
◆	EncryptFileAESA (see page 38)	Encrypts file using AES algorithm.
◆	EncryptFileAESW (see page 39)	Encrypts file using AES algorithm.
◆	GetFileMD5A (see page 63)	Gets the MD5 hash value for the file
◆	GetFileMD5W (see page 63)	Gets the MD5 hash value for the file
◆	GetFileSHA1A (see page 64)	Gets the SHA1 hash value for the file
◆	GetFileSHA1W (see page 64)	Gets the SHA1 hash value for the file
◆	GetFileSHA256A (see page 65)	Gets the SHA256 hash value for the file
◆	GetFileSHA256W (see page 65)	Gets the SHA256 hash value for the file
◆	GetFileSHA384A (see page 66)	This is function GetFileSHA384A.
◆	GetFileSHA384W (see page 66)	This is function GetFileSHA384W.
◆	GetFileSHA512A (see page 66)	This is function GetFileSHA512A.
◆	GetFileSHA512W (see page 66)	This is function GetFileSHA512W.
◆	GetMD5 (see page 69)	Gets the MD5 hash value for the content of the memory buffer
◆	GetSHA1 (see page 73)	Gets the SHA1 hash value for the content of the memory buffer
◆	GetSHA256 (see page 73)	Gets the SHA256 hash value for the content of the memory buffer
◆	GetSHA384 (see page 74)	This is function GetSHA384.
◆	GetSHA512 (see page 74)	This is function GetSHA512.

## 1.3.5 FileOperations.h

Files and folders manipulations

### Functions

	Name	Description
✚	AllocateBuffer (see page 12)	Allocates the buffer in memory
✚	ClearFileAttributesA (see page 24)	This function sets the file attributes to normal.
✚	ClearFileAttributesW (see page 24)	This function sets the file attributes to normal.
✚	CreateUnicodeFileA (see page 27)	Creates UNICODE file
✚	CreateUnicodeFileW (see page 27)	Creates UNICODE file
✚	DeallocateBuffer (see page 29)	Deallocates the memory buffer
✚	DeleteToRecycleBinA (see page 31)	Deletes file to the Windows Recycle Bin
✚	DeleteToRecycleBinW (see page 32)	Deletes file to WIndows Recycle Bin
✚	DirectoryExistsA (see page 33)	Determines whether a specified directory exists.
✚	DirectoryExistsW (see page 33)	Determines whether a specified directory exists.
✚	FileCloseA (see page 39)	Concludes input/output (I/O) to a file opened using the FileCreateRewrite function.
✚	FileCloseW (see page 39)	Concludes input/output (I/O) to a file opened using the FileCreateRewrite function.
✚	FileCopyA (see page 40)	The CopyFile function copies an existing file to a new file.
✚	FileCopyW (see page 40)	The CopyFile function copies an existing file to a new file.
✚	FileCreateRewriteA (see page 41)	Creates new or overwrites existing file
✚	FileCreateRewriteW (see page 41)	Creates new or overwrites existing file
✚	FileDeleteA (see page 41)	Deletes a file from disk.
✚	FileDeleteW (see page 42)	Deletes a file from disk.
✚	FileExistsA (see page 42)	Tests whether a specified file exists.
✚	FileExistsW (see page 42)	Tests whether a specified file exists.
✚	FileExtensionIsA (see page 43)	Checks the file extension
✚	FileExtensionIsW (see page 43)	Checks the file extension
✚	FileGetSizeA (see page 43)	Retrieves the size of a specified file.
✚	FileGetSizeW (see page 44)	Retrieves the size of a specified file.
✚	FileIsExeA (see page 44)	Checks if the file is a Windows executable
✚	FileIsExeW (see page 45)	Checks if the file is a Windows executable
✚	FileIsIconA (see page 45)	Checks if the file is a Windows icon (.ico) file
✚	FileIsIconW (see page 45)	Checks if the file is a Windows icon (.ico) file
✚	FileIsImageA (see page 46)	Checks if the file is an image file
✚	FileIsImageW (see page 46)	Checks if the file is an image file
✚	FileIsLink (see page 46)	Checks to see if the file specified by file name is a Microsoft Windows shortcut (.Lnk) file (and is neither a file nor a folder).
✚	FileOrFolderExistsA (see page 47)	Checks if the file or folder with the given name exists

◆	FileOrFolderExistsW ( see page 47)	Checks if the file or folder with the given name exists
◆	FileRenameA ( see page 47)	Renames the file
◆	FileRenameW ( see page 48)	Renames the file
◆	FileWriteA ( see page 48)	Writes string to the file
◆	FileWriteCharA ( see page 48)	Writes one character to the file
◆	FileWriteCharW ( see page 49)	Writes one character to the file
◆	FileWriteNewLineA ( see page 49)	Writes new line sequence to the file
◆	FileWriteNewLineW ( see page 49)	Writes new line sequence to the file
◆	FileWriteW ( see page 50)	Writes string to the file
◆	FullPathA ( see page 50)	Gets the full path to the file based on file name
◆	FullPathW ( see page 51)	Gets the full path to the file based on file name
◆	GetAllFiles ( see page 59)	Returns the names of files in a specified directory.
◆	GetFilesCountA ( see page 66)	Calculates the number of files in the given folder
◆	GetFilesCountW ( see page 67)	Calculates the number of files in the given folder
◆	IsAnimatedGIFA ( see page 78)	Checks if the file is an animated GIF image file
◆	IsAnimatedGIFW ( see page 78)	Checks if the file is an animated GIF image file
◆	IsDirectoryA ( see page 81)	Verifies that a path is a valid directory.
◆	IsDirectoryW ( see page 81)	Verifies that a path is a valid directory.
◆	IsUnicodeFileA ( see page 86)	Checks if the file is UNICODE file
◆	IsUnicodeFileW ( see page 86)	Checks if the file is UNICODE file
◆	IsValidFileNameA ( see page 87)	Checks if provided string is a valid file name
◆	IsValidFileNameW ( see page 87)	Checks if provided string is a valid file name
◆	IsValidPathNameA ( see page 88)	Checks if provided string is a valid file path
◆	IsValidPathNameW ( see page 88)	Checks if provided string is a valid file path
◆	ReadBufferFromFileA ( see page 99)	Reads the content of the file to the memory buffer
◆	ReadBufferFromFileW ( see page 99)	Reads the content of the file to the memory buffer
◆	ShellCopyFileA ( see page 129)	Copies file to the new location
◆	ShellCopyFileW ( see page 130)	Copies file to the new location
◆	StripFileNameA ( see page 132)	Replaces environment variable names with values
◆	StripFileNameW ( see page 133)	Replaces environment variable names with values
◆	WriteBufferToFileA ( see page 136)	Writes the memory buffer to file
◆	WriteBufferToFileW ( see page 137)	Writes the memory buffer to file

## 1.3.6 Graphics.h

This is file Graphics.h.

### Functions

	Name	Description
◆	AlphaBlendBitmap ( see page 15)	This is function AlphaBlendBitmap.
◆	AlphaBlendNative ( see page 15)	This is function AlphaBlendNative.
◆	CloneFont ( see page 25)	This is function CloneFont.
◆	CopyNativeBitmap ( see page 26)	This is function CopyNativeBitmap.



✦	CreateNativeBitmap (see page 27)	This is function CreateNativeBitmap.
✦	CreateWindowsFont (see page 28)	This is function CreateWindowsFont.
✦	DestroyFont (see page 32)	This is function DestroyFont.
✦	DpiY (see page 34)	This is function DpiY.
✦	DrawAlphaText (see page 34)	This is function DrawAlphaText.
✦	DrawAlphaTextRect (see page 34)	This is function DrawAlphaTextRect.
✦	DrawNativeBitmap (see page 35)	This is function DrawNativeBitmap.
✦	DrawTextDirect (see page 35)	This is function DrawTextDirect.
✦	DrawTextDirectEx (see page 36)	This is function DrawTextDirectEx.
✦	DrawTextGlow (see page 36)	This is function DrawTextGlow.
✦	DrawTextLine (see page 36)	This is function DrawTextLine.
✦	DrawTextOutline (see page 36)	This is function DrawTextOutline.
✦	DrawTextRect (see page 37)	This is function DrawTextRect.
✦	EmToPixels (see page 37)	This is function EmToPixels.
✦	GetTextLineSize (see page 76)	This is function GetTextLineSize.
✦	GetTextSize (see page 76)	This is function GetTextSize.
✦	GetTextSizeEx (see page 76)	This is function GetTextSizeEx.
✦	LoadBitmapJPG (see page 91)	This is function LoadBitmapJPG.
✦	LoadBitmapPNG (see page 91)	This is function LoadBitmapPNG.
✦	LoadPNGResource (see page 93)	This is function LoadPNGResource.
✦	MakeCompatibleBitmap (see page 94)	This is function MakeCompatibleBitmap.
✦	PointsToPixels (see page 96)	This is function PointsToPixels.
✦	StretchNativeBitmap (see page 132)	This is function StretchNativeBitmap.

## 1.3.7 NationalityConverter.h

Nationality to ISO code converter

### Functions

	Name	Description
✦	GetISOCodeA (see page 68)	Returns the country ISO code based on the nationality string
✦	GetISOCodeW (see page 69)	Returns the country ISO code based on the nationality string

## 1.3.8 Network.h

### Functions

	Name	Description
✦	ServerAccessible (see page 126)	Check if the specified server is accessible



## 1.3.9 Swelio.h

Belgian electronic Id card access engine

### Functions


	Name	Description
≡	ActivateCard (see page 10)	Established communication between the card and the reader
≡	ActivateCardEx (see page 11)	Established communication between the card and the reader
≡	AddFileToContainer (see page 11)	Add existing file to the container
≡	CardSignCMS (see page 17)	Sign data with eID card according to CMS standard
≡	CardSignCMSEx (see page 18)	Sign data with eID card according to CMS standard
≡	CardSignCadesT (see page 18)	Sign data with eID card according to CADES-T standard
≡	CardSignCadesTEx (see page 19)	Sign data with eID card according to CADES-T standard
≡	CertSignCMS (see page 19)	Sign data with the certificate file according to CMS standard
≡	CertSignCMSData (see page 20)	Sign data with the certificate according to CMS standard
≡	CertSignCMSEx (see page 20)	Sign data with the certificate file according to CMS standard
≡	CertSignCadesT (see page 21)	Sign data with the certificate file according to CADES-T standard
≡	CertSignCadesTData (see page 21)	Sign data with the certificate according to CADES-T standard
≡	CertSignCadesTEx (see page 22)	Sign data with the certificate file according to CADES-T standard
≡	ContainerCertificate (see page 25)	Assign certificate for signing ASIC container
≡	ContainerEidCertificate (see page 25)	Select EID card certificate to sign ASIC container
≡	ContainerPickCertificate (see page 26)	Pick certificate to sign ASIC container
≡	CreateCardBuffer (see page 27)	Creates XML buffer
≡	DeactivateCard (see page 28)	Terminates a connection between a smart card and a reader
≡	DeactivateCardEx (see page 29)	Terminates a connection between a smart card and a reader
≡	DeleteCardBuffer (see page 31)	Deletes XML buffer
≡	DisplayCertificate (see page 34)	Displays the dialog window with certificate information
≡	EncodeCertificate (see page 37)	Performs Base64 encoding of the certificate
≡	EncodePhoto (see page 38)	Performs Base64 encoding of the photo
≡	FreeContainer (see page 50)	Deallocates ASIC container
≡	GenerateAuthenticationSignatureA (see page 51)	Generate authentication signature
≡	GenerateAuthenticationSignatureExA (see page 51)	Generate authentication signature
≡	GenerateAuthenticationSignatureExW (see page 52)	Generate authentication signature
≡	GenerateAuthenticationSignatureW (see page 53)	Generate authentication signature
≡	GenerateNonRepudiationSignatureA (see page 54)	Generate non repudiation signature
≡	GenerateNonRepudiationSignatureExA (see page 55)	Generate non repudiation signature
≡	GenerateNonRepudiationSignatureExW (see page 55)	Generate non repudiation signature

GenerateNonRepudiationSignatureW (see page 56)	Generate non repudiation signature
GenerateQRCodeA (see page 57)	Read eID card and save the identity information and address to PNG QR Code file
GenerateQRCodeExA (see page 57)	Read eID card and save the identity information and address to PNG QR Code file
GenerateQRCodeExW (see page 58)	Read eID card and save the identity information and address to PNG QR Code file
GenerateQRCodeW (see page 58)	Read eID card and save the identity information and address to PNG QR Code file
GetCardBufferA (see page 59)	Gets XML or CSV information from the memory buffer
GetCardBufferSize (see page 60)	This function returns the size of the buffer needed to hold the information from the eID card in the XML or CSV format
GetCardBufferW (see page 60)	Gets XML or CSV information from the memory buffer
GetCardSerialNumber (see page 60)	Get the serial number of EID card
GetCardVersion (see page 61)	Get the applet version number for card in the reader with specified number
GetContainerError (see page 61)	Gets the value of the container validation error message based on the message index
GetContainerErrorsCount (see page 62)	Get the number of error messages available in the container validation report
GetEncodedCertificateSize (see page 62)	Returns the size of the Base64 encoded certificate
GetEncodedPhotoSize (see page 63)	Calculates buffer size for Base64 encoded photo
GetReaderIndexA (see page 70)	Returns the zero-based reader index with specified name
GetReaderIndexW (see page 71)	Returns the zero-based reader index with specified name
GetReaderNameA (see page 71)	Returns the name of the card reader
GetReaderNameLenA (see page 72)	Returns the length of the reader name
GetReaderNameLenW (see page 72)	Returns the length of the reader name
GetReaderNameW (see page 72)	Returns the name of the card reader
GetReadersCount (see page 73)	Get number of card readers connected to PC
GetSelectedReaderIndex (see page 74)	Returns the index of the active smart card reader
GetSupportSIS (see page 75)	Checks if the SIS cards are supported by the engine
IgnoreHardwareEvents (see page 77)	Ignore USB reader insert / remove events
IgnoreServiceEvents (see page 77)	Ignore smartcard service stop events when reporting readers list change
InitializeContainer (see page 77)	Initializes ASIC container
IsCardActivated (see page 78)	Checks the connection between a smart card and a reader
IsCardActivatedEx (see page 79)	Checks the connection between a smart card and a reader
IsCardPresent (see page 79)	Checks if the card is present in the card reader
IsCardPresentEx (see page 79)	Checks if the card is present in the card reader
IsCardStillInserted (see page 80)	Checks if the card is still inserted in the card reader
IsCardStillInsertedEx (see page 80)	Checks if the card is still inserted in the card reader
IsEIDCard (see page 81)	Check if Belgian EID card is inserted into card reader
IsEIDCardEx (see page 82)	Check if Belgian EID card is inserted into card reader
IsEngineActive (see page 82)	Checks if the Swelio Engine is activated
IsFemaleA (see page 82)	Checks if the card owner is female
IsFemaleW (see page 83)	Checks if the card owner is female
IsMaleA (see page 83)	Checks if the card owner is male
IsMaleW (see page 84)	Checks if the card owner is male
IsSISCard (see page 85)	Check if Belgian SIS card is inserted into card reader

IsSISCardEx (see page 85)	Check if Belgian SIS card is inserted into card reader
LoadAddressA (see page 90)	This is function LoadAddressA.
LoadAddressW (see page 91)	This is function LoadAddressW.
LoadCertificateA (see page 91)	Reads the certificate from a file
LoadCertificateW (see page 92)	Reads the certificate from a file
LoadIdentityA (see page 92)	Reads the raw identity information from a file
LoadIdentityW (see page 92)	Reads the raw identity information from a file
LoadPhotoA (see page 93)	Loads photo from a file
LoadPhotoW (see page 93)	Loads photo from a file
NonRepudiationSignatureAlgoA (see page 95)	This is function NonRepudiationSignatureAlgoA.
NonRepudiationSignatureAlgoW (see page 96)	This is function NonRepudiationSignatureAlgoW.
ReadAddressA (see page 96)	Read address information from Belgian eID card
ReadAddressExA (see page 97)	Read address information from Belgian eID card
ReadAddressExW (see page 97)	Read address information from Belgian eID card
ReadAddressW (see page 97)	Read address information from Belgian eID card
ReadAuthenticationCertificate (see page 98)	Read Authentication Certificate to memory
ReadAuthenticationCertificateEx (see page 98)	Read Authentication Certificate to memory
ReadCaCertificate (see page 99)	Read Ca Certificate to memory
ReadCaCertificateEx (see page 100)	Read Ca Certificate to memory
ReadIdentityA (see page 100)	Read identity information from Belgian eID card
ReadIdentityExA (see page 101)	Read identity information from Belgian eID card
ReadIdentityExW (see page 101)	Read identity information from Belgian eID card
ReadIdentityW (see page 101)	Read identity information from Belgian eID card
ReadNonRepudiationCertificate (see page 102)	Read Non Repudiation Certificate to memory
ReadNonRepudiationCertificateEx (see page 102)	Read Non Repudiation Certificate to memory
ReadPhoto (see page 102)	Reads a photo from a card
ReadPhotoAsBitmap (see page 103)	Reads the picture from the card, converts it to bitmap and returns the bitmap handle Description: Reads the photo from the Belgian eID card and returns the bitmap handle Reading the photo from the card is a time consuming operation. Do it only when needed.
ReadPhotoAsBitmapEx (see page 103)	Reads the picture from the card, converts it to bitmap and returns the bitmap handle Description: Reads the photo from the Belgian eID card and returns the Windows bitmap handle Reading the photo from the card is a time consuming operation. Do it only when needed.
ReadPhotoEx (see page 104)	Reads a photo from a card
ReadRootCaCertificate (see page 104)	Read Root Ca Certificate to memory
ReadRootCaCertificateEx (see page 104)	Read Root Ca Certificate to memory
ReadRrnCertificate (see page 105)	Read Rrn Certificate to memory
ReadRrnCertificateEx (see page 105)	Read Rrn Certificate to memory
ReadSISCardA (see page 106)	Read Belgian SIS card.
ReadSISCardExA (see page 106)	Read Belgian SIS card.
ReadSISCardExW (see page 106)	Read Belgian SIS card.
ReadSISCardW (see page 107)	Read Belgian SIS card.
ReloadReadersList (see page 107)	Reloads the list of the available card readers

RemoveCallback (see page 108)	Remove callback procedure for card events
SaveAddressA (see page 109)	This is function SaveAddressA.
SaveAddressW (see page 110)	This is function SaveAddressW.
SaveAuthenticationCertificateA (see page 110)	Save Authentication Certificate to a file
SaveAuthenticationCertificateExW (see page 110)	Save Authentication Certificate to a file
SaveAuthenticationCertificateW (see page 111)	Save Authentication Certificate to a file
SaveCaCertificateA (see page 111)	Save Ca Certificate to a file
SaveCaCertificateExW (see page 111)	Save Ca Certificate to a file
SaveCaCertificateW (see page 112)	Save Ca Certificate to a file
SaveCardToToXMLStreamExA (see page 112)	Read eID card and save the information to XML buffer
SaveCardToToXMLStreamExW (see page 112)	Read eID card and save the information to XML buffer
SaveCardToXmlA (see page 113)	Read eID card and save the information to XML file
SaveCardToXmlExA (see page 113)	Read eID card and save the information to XML file
SaveCardToXmlExW (see page 114)	Read eID card and save the information to XML file
SaveCardToXmlW (see page 114)	Read eID card and save the information to XML file
SaveContainer (see page 114)	Save container to the file
SaveIdentityA (see page 115)	Saves identity information to a file
SaveIdentityW (see page 115)	Saves identity information to a file
SaveNonRepudiationCertificateA (see page 116)	Save Non Repudiation Certificate to a file
SaveNonRepudiationCertificateExW (see page 116)	Save Non Repudiation Certificate to a file
SaveNonRepudiationCertificateW (see page 116)	Save Non Repudiation Certificate to a file
SavePersonCsvToStreamA (see page 117)	Read eID card and save the identity information to CSV memory buffer
SavePersonCsvToStreamW (see page 117)	Read eID card and save the identity information to CSV memory buffer
SavePersonToCsvA (see page 117)	Read eID card and save the identity information and address to CSV file
SavePersonToCsvExA (see page 118)	Read eID card and save the identity information and address to CSV file
SavePersonToCsvExW (see page 118)	Read eID card and save the identity information and address to CSV file
SavePersonToCsvW (see page 119)	Read eID card and save the identity information and address to CSV file
SavePhotoA (see page 119)	Save photo to a file
SavePhotoAsBitmapA (see page 119)	Save the picture from the card to Windows Bitmap file Description: Reads the photo from the Belgian eID card and writes it to the file as bitmap image. Reading the photo from the card is a time consuming operation. Do it only when needed.
SavePhotoAsBitmapExA (see page 120)	Reads the picture from the card and saves it to Windows Bitmap file Description: Reads the photo from the Belgian eID card and writes it to the file as bitmap image. Reading the photo from the card is a time consuming operation. Do it only when needed.
SavePhotoAsBitmapExW (see page 120)	Reads the picture from the card and saves it to Windows Bitmap file Description: Reads the photo from the Belgian eID card and writes it to the file as bitmap image. Reading the photo from the card is a time consuming operation. Do it only when needed.

SavePhotoAsBitmapW (see page 121)	Save the picture from the card to Windows Bitmap file Description: Reads the photo from the Belgian eID card and writes it to the file as bitmap image. Reading the photo from the card is a time consuming operation. Do it only when needed.
SavePhotoAsJpegA (see page 121)	Save the picture from the card to JPG file Description: Reads the photo from the Belgian eID card and writes it to the file as JPG image. Reading the photo from the card is a time consuming operation. Do it only when needed.
SavePhotoAsJpegExA (see page 121)	Save the picture from the card to JPG file Description: Reads the photo from the Belgian eID card and writes it to the file as JPG image. Reading the photo from the card is a time consuming operation. Do it only when needed.
SavePhotoAsJpegExW (see page 122)	Save the picture from the card to JPG file Description: Reads the photo from the Belgian eID card and writes it to the file as JPG image. Reading the photo from the card is a time consuming operation. Do it only when needed.
SavePhotoAsJpegW (see page 122)	Save the picture from the card to JPG file Description: Reads the photo from the Belgian eID card and writes it to the file as JPG image. Reading the photo from the card is a time consuming operation. Do it only when needed.
SavePhotoW (see page 122)	Saves photo to a file
SaveRootCaCertificateA (see page 123)	Save Root Ca Certificate to a file
SaveRootCaCertificateExW (see page 123)	Save Root Ca Certificate to a file
SaveRootCaCertificateW (see page 124)	Save Root Ca Certificate to a file
SaveRrnCertificateA (see page 124)	Save RRN Certificate to a file
SaveRrnCertificateExW (see page 124)	Save RRN Certificate to a file
SaveRrnCertificateW (see page 125)	Save RRN Certificate to a file
SelectReader (see page 125)	When more than 1 reader connected, select the reader with specified number
SelectReaderByNameA (see page 125)	Select active smart card reader by providing the reader name
SelectReaderByNameW (see page 126)	Select active smart card reader by providing the reader name
SendAPDU (see page 126)	This is function SendAPDU.
SetCallback (see page 127)	Activates callback procedure for card status change event
SetContainerCanonization (see page 127)	Set XML canonization standard of the ASIC container
SetContainerTimeServer (see page 128)	Set time server URI for digital signature of the ASIC container
SetMWCompatibility (see page 128)	Set the compatibility mode with the old version of the official EID MiddleWare
SetSupportSIS (see page 129)	Activates or deactivates SIS card support by engine
SignPdfFile (see page 131)	Digitally sign PDF file using Belgian EID card
SignPdfFileEx (see page 131)	Digitally sign PDF file using Belgian EID card
StartEngine (see page 131)	Activates the Swelio Engine.
StopEngine (see page 132)	Deactivates the Swelio Engine
VerifyContainer (see page 134)	Verify the signatures and the integrity of the ASIC container
VerifyPinA (see page 134)	Verify PIN code
VerifyPinExA (see page 135)	Verify PIN code
VerifyPinExW (see page 135)	Verify PIN code
VerifyPinW (see page 136)	Verify PIN code

	VerifySignature ( <a href="#">see page 136</a> )	Verifies the signature from the specified hash value.
---	--	---

## 1.3.10 System.h

Windows related routines

### Functions

	Name	Description
	AddRemoveMessageFilter ( <a href="#">see page 11</a> )	Adds or removes a message from the User Interface Privilege Isolation (UIPI) message filter.
	AllocateDefaultHWNDA ( <a href="#">see page 12</a> )	This function creates the invisible tool window
	AllocateDefaultHWNDA ( <a href="#">see page 13</a> )	This function creates the invisible tool window
	AllocateHWNDA ( <a href="#">see page 13</a> )	This function creates the invisible tool window using the provided window procedure
	AllocateHWNDA ( <a href="#">see page 13</a> )	This function creates the invisible tool window using the provided window procedure
	AllocateLayeredWindowA ( <a href="#">see page 14</a> )	This function creates the layered window using the provided window class name
	AllocateLayeredWindowW ( <a href="#">see page 14</a> )	This function creates the layered window using the provided window class name
	AllocateWindowClassA ( <a href="#">see page 14</a> )	This function creates the standard window using the provided window class name
	AllocateWindowClassW ( <a href="#">see page 15</a> )	This function creates the standard window using the provided window class name
	BringWindowToFront ( <a href="#">see page 15</a> )	This function brings the specified window to the top of the z-order.
	ClearUnusedMemory ( <a href="#">see page 25</a> )	Clears unused memory and minimized the application memory usage
	DeallocateHWNDA ( <a href="#">see page 29</a> )	This function destroys the specified window.
	DeallocateHWNDA ( <a href="#">see page 30</a> )	This function destroys the specified window.
	DrawLayeredWindow ( <a href="#">see page 35</a> )	Repaints the surface of the layered window
	EmptyRecycleBin ( <a href="#">see page 37</a> )	Empties the recycle bin
	GetStartupA ( <a href="#">see page 75</a> )	Checks if the application is registered to run when Windows starts
	GetStartupW ( <a href="#">see page 75</a> )	Checks if the application is registered to run when Windows starts
	HibernateWindows ( <a href="#">see page 76</a> )	Hibernates Windows
	LayeredWndProcA ( <a href="#">see page 90</a> )	The default window procedure for the layered window
	LayeredWndProcW ( <a href="#">see page 90</a> )	The default window procedure for the layered window
	MakeSoundFromFileA ( <a href="#">see page 94</a> )	Plays the wave sound from the file
	MakeSoundFromFileW ( <a href="#">see page 94</a> )	Plays the wave sound from the file
	MakeSoundFromResourceA ( <a href="#">see page 95</a> )	Plays the wave sound from the resource
	MakeSoundFromResourceW ( <a href="#">see page 95</a> )	Plays the wave sound from the resource
	RemoveStartupA ( <a href="#">see page 108</a> )	Removes the application from the list of the automatically started applications



RemoveStartupW (see page 108)	Removes the application from the list of the automatically started applications
RestoreWindowSubclassA (see page 109)	Restores window standard procedure
RestoreWindowSubclassW (see page 109)	Restores window standard procedure
SetStartupA (see page 128)	Register application to run when Windows starts
SetStartupW (see page 129)	Register application to run when Windows starts
ShutdownWindows (see page 130)	Logs off the interactive user, shuts down the system.
SuspendWindows (see page 133)	Suspends Windows
TurnMonitorOff (see page 133)	Turns the monitor off
TurnMonitorOn (see page 133)	Turns the monitor on
UpdateWindowPosition (see page 134)	Updated the window position
fpreset (see page 137)	This is function fpreset.

## 1.3.11 SystemInfo.h

Routines for querying information about operating system

### Functions

Name	Description
CurrentIPAddressA (see page 28)	Returns the IP address
CurrentIPAddressW (see page 28)	Returns the IP address
IsCitrixSession (see page 80)	Checks if application is running in Citrix session
IsConnectedToInternet (see page 80)	Checks if PC is connected to Internet
IsMediaCenter (see page 84)	Checks if the Media Center version of Windows is installed
IsMetroActive (see page 84)	Checks if metro interface is active
IsMultiTouchReady (see page 84)	Checks if the system is multi touch ready
IsNativeWin64 (see page 85)	Checks if the application is native 64 bit executable
IsRemoteSession (see page 85)	Checks if application is running in RDP session
IsTabletPC (see page 86)	Checks if the application is running on the Tablet PC
IsWindows10 (see page 88)	Checks if PC is running Windows 10 or better
IsWindows7 (see page 89)	Checks if PC is running Windows 7 or better
IsWindows8 (see page 89)	Checks if PC is Running Windows 8 or better
IsWindowsVista (see page 89)	Checks if PC is running Windows Vista or better
IsWindowsXP (see page 89)	Checks if PC is running Windows XP
IsWindowsXPSP2 (see page 90)	Checks if PC is running Windows XP with Service Pack 2 installed
IsWow64 (see page 90)	Checks if the 32 bit application runs on 64 bit Windows
PortAvailable (see page 96)	Checks if the port with specified number is available
RecycleBinEmpty (see page 107)	Returns TRUE if Windows Recycle Bin is empty

## 1.3.12 quicol.h

QR Code generator

**Functions**

	<b>Name</b>	<b>Description</b>
≡	DestroyImageBuffer (see page 32)	Destroys the memory buffer
≡	GenerateBMPA (see page 53)	Generates Windows Bitmap file with QR Code image
≡	GenerateBMPW (see page 54)	Generates Windows Bitmap file with QR Code image
≡	GeneratePNGA (see page 56)	Generates PNG file with QR Code image
≡	GeneratePNGW (see page 57)	Generates PNG file with QR Code image
≡	GetHBitmapA (see page 67)	Generates Windows Bitmap in memory with QR Code image
≡	GetHBitmapW (see page 68)	Generates Windows Bitmap in memory with QR Code image
≡	GetPNGA (see page 69)	Writes PNG image to the memory buffer.
≡	GetPNGW (see page 70)	Writes PNG image to the memory buffer.



## Index

### A

ActivateCard 10  
ActivateCard function 10  
ActivateCardEx 11  
ActivateCardEx function 11  
AddFileToContainer 11  
AddFileToContainer function 11  
AddRemoveMessageFilter 11  
AddRemoveMessageFilter function 11  
Algorithms.h 167  
AllocateBuffer 12  
AllocateBuffer function 12  
AllocateDefaultHWND 12  
AllocateDefaultHWND function 12  
AllocateDefaultHWNDW 13  
AllocateDefaultHWNDW function 13  
AllocateHWND 13  
AllocateHWND function 13  
AllocateHWNDW 13  
AllocateHWNDW function 13  
AllocateLayeredWindowA 14  
AllocateLayeredWindowA function 14  
AllocateLayeredWindowW 14  
AllocateLayeredWindowW function 14  
AllocateWindowClassA 14  
AllocateWindowClassA function 14  
AllocateWindowClassW 15  
AllocateWindowClassW function 15  
AlphaBlendBitmap 15  
AlphaBlendBitmap function 15  
AlphaBlendNative 15  
AlphaBlendNative function 15

### B

BringWindowToFront 15  
BringWindowToFront function 15

### C

CardDecryptFileA 16

CardDecryptFileA function 16  
CardDecryptFileW 16  
CardDecryptFileW function 16  
CardEncryptFileA 16  
CardEncryptFileA function 16  
CardEncryptFileW 17  
CardEncryptFileW function 17  
CardEventType 148  
CardEventType enumeration 148  
CardEvents.h 168  
CardSignCMS 17  
CardSignCMS function 17  
CardSignCMSEx 18  
CardSignCMSEx function 18  
CardSignCadesT 18  
CardSignCadesT function 18  
CardSignCadesTEx 19  
CardSignCadesTEx function 19  
CardStructures.h 168  
CertSignCMS 19  
CertSignCMS function 19  
CertSignCMSData 20  
CertSignCMSData function 20  
CertSignCMSEx 20  
CertSignCMSEx function 20  
CertSignCadesT 21  
CertSignCadesT function 21  
CertSignCadesTData 21  
CertSignCadesTData function 21  
CertSignCadesTEx 22  
CertSignCadesTEx function 22  
CheckMD5 22  
CheckMD5 function 22  
CheckSHA1 23  
CheckSHA1 function 23  
CheckSHA256 23  
CheckSHA256 function 23  
ClearFileAttributesA 24  
ClearFileAttributesA function 24  
ClearFileAttributesW 24  
ClearFileAttributesW function 24  
ClearUnusedMemory 25

ClearUnusedMemory function 25  
CloneFont 25  
CloneFont function 25  
ContainerCertificate 25  
ContainerCertificate function 25  
ContainerEidCertificate 25  
ContainerEidCertificate function 25  
ContainerPickCertificate 26  
ContainerPickCertificate function 26  
CopyNativeBitmap 26  
CopyNativeBitmap function 26  
CreateCardBuffer 27  
CreateCardBuffer function 27  
CreateNativeBitmap 27  
CreateNativeBitmap function 27  
CreateUnicodeFileA 27  
CreateUnicodeFileA function 27  
CreateUnicodeFileW 27  
CreateUnicodeFileW function 27  
CreateWindowsFont 28  
CreateWindowsFont function 28  
CurrentIPAddressA 28  
CurrentIPAddressA function 28  
CurrentIPAddressW 28  
CurrentIPAddressW function 28

## D

DeactivateCard 28  
DeactivateCard function 28  
DeactivateCardEx 29  
DeactivateCardEx function 29  
DeallocateBuffer 29  
DeallocateBuffer function 29  
DeallocateHWNDAs 29  
DeallocateHWNDAs function 29  
DeallocateHWNDW 30  
DeallocateHWNDW function 30  
DecryptFileAESA 30  
DecryptFileAESA function 30  
DecryptFileAESW 31  
DecryptFileAESW function 31  
DeleteCardBuffer 31

DeleteCardBuffer function 31  
DeleteToRecycleBinA 31  
DeleteToRecycleBinA function 31  
DeleteToRecycleBinW 32  
DeleteToRecycleBinW function 32  
DestroyFont 32  
DestroyFont function 32  
DestroyImageBuffer 32  
DestroyImageBuffer function 32  
DirectoryExistsA 33  
DirectoryExistsA function 33  
DirectoryExistsW 33  
DirectoryExistsW function 33  
DisplayCertificate 34  
DisplayCertificate function 34  
DpiY 34  
DpiY function 34  
DrawAlphaText 34  
DrawAlphaText function 34  
DrawAlphaTextRect 34  
DrawAlphaTextRect function 34  
DrawLayeredWindow 35  
DrawLayeredWindow function 35  
DrawNativeBitmap 35  
DrawNativeBitmap function 35  
DrawTextDirect 35  
DrawTextDirect function 35  
DrawTextDirectEx 36  
DrawTextDirectEx function 36  
DrawTextGlow 36  
DrawTextGlow function 36  
DrawTextLine 36  
DrawTextLine function 36  
DrawTextOutline 36  
DrawTextOutline function 36  
DrawTextRect 37  
DrawTextRect function 37

## E

EidAddressA 149  
EidAddressA structure 149  
EidAddressW 149

---

EidAddressW structure 149	FileDeleteW 42
EidCertificate 149	FileDeleteW function 42
EidCertificate structure 149	FileExistsA 42
EidIdentityA 150	FileExistsA function 42
EidIdentityA structure 150	FileExistsW 42
EidIdentityExA 151	FileExistsW function 42
EidIdentityExA structure 151	FileExtensionIsA 43
EidIdentityExW 153	FileExtensionIsA function 43
EidIdentityExW structure 153	FileExtensionIsW 43
EidIdentityW 154	FileExtensionIsW function 43
EidIdentityW structure 154	FileGetSizeA 43
EidPicture 156	FileGetSizeA function 43
EidPicture structure 156	FileGetSizeW 44
EmToPixels 37	FileGetSizeW function 44
EmToPixels function 37	FileIsExeA 44
EmptyRecycleBin 37	FileIsExeA function 44
EmptyRecycleBin function 37	FileIsExeW 45
EncodeCertificate 37	FileIsExeW function 45
EncodeCertificate function 37	FileIsIconA 45
EncodePhoto 38	FileIsIconA function 45
EncodePhoto function 38	FileIsIconW 45
EncryptFileAESA 38	FileIsIconW function 45
EncryptFileAESA function 38	FileIsImageA 46
EncryptFileAESW 39	FileIsImageA function 46
EncryptFileAESW function 39	FileIsImageW 46
Encryption.h 169	FileIsImageW function 46
ErrorInformation 156	FileIsLink 46
ErrorInformation structure 156	FileIsLink function 46
FileCloseA 39	FileOperations.h 170
FileCloseA function 39	FileOrFolderExistsA 47
FileCloseW 39	FileOrFolderExistsA function 47
FileCloseW function 39	FileOrFolderExistsW 47
FileCopyA 40	FileOrFolderExistsW function 47
FileCopyA function 40	FileRenameA 47
FileCopyW 40	FileRenameA function 47
FileCopyW function 40	FileRenameW 48
FileCreateRewriteA 41	FileRenameW function 48
FileCreateRewriteA function 41	FileWriteA 48
FileCreateRewriteW 41	FileWriteA function 48
FileCreateRewriteW function 41	FileWriteCharA 48
FileDeleteA 41	FileWriteCharA function 48
FileDeleteA function 41	FileWriteCharW 49

---

FileWriteCharW function 49	GenerateQRCodeExA function 57
FileWriteNewLineA 49	GenerateQRCodeExW 58
FileWriteNewLineA function 49	GenerateQRCodeExW function 58
FileWriteNewLineW 49	GenerateQRCodeW 58
FileWriteNewLineW function 49	GenerateQRCodeW function 58
FileWriteW 50	GetAllFiles 59
FileWriteW function 50	GetAllFiles function 59
Files 167	GetCardBufferA 59
FreeContainer 50	GetCardBufferA function 59
FreeContainer function 50	GetCardBufferSize 60
FullPathA 50	GetCardBufferSize function 60
FullPathA function 50	GetCardBufferW 60
FullPathW 51	GetCardBufferW function 60
FullPathW function 51	GetCardSerialNumber 60
Functions 1	GetCardSerialNumber function 60
GenerateAuthenticationSignatureA 51	GetCardVersion 61
GenerateAuthenticationSignatureA function 51	GetCardVersion function 61
GenerateAuthenticationSignatureExA 51	GetContainerError 61
GenerateAuthenticationSignatureExA function 51	GetContainerError function 61
GenerateAuthenticationSignatureExW 52	GetContainerErrorsCount 62
GenerateAuthenticationSignatureExW function 52	GetContainerErrorsCount function 62
GenerateAuthenticationSignatureW 53	GetEncodedCertificateSize 62
GenerateAuthenticationSignatureW function 53	GetEncodedCertificateSize function 62
GenerateBMPA 53	GetEncodedPhotoSize 63
GenerateBMPA function 53	GetEncodedPhotoSize function 63
GenerateBMPW 54	GetFileMD5A 63
GenerateBMPW function 54	GetFileMD5A function 63
GenerateNonRepudiationSignatureA 54	GetFileMD5W 63
GenerateNonRepudiationSignatureA function 54	GetFileMD5W function 63
GenerateNonRepudiationSignatureExA 55	GetFileSHA1A 64
GenerateNonRepudiationSignatureExA function 55	GetFileSHA1A function 64
GenerateNonRepudiationSignatureExW 55	GetFileSHA1W 64
GenerateNonRepudiationSignatureExW function 55	GetFileSHA1W function 64
GenerateNonRepudiationSignatureW 56	GetFileSHA256A 65
GenerateNonRepudiationSignatureW function 56	GetFileSHA256A function 65
GeneratePNGA 56	GetFileSHA256W 65
GeneratePNGA function 56	GetFileSHA256W function 65
GeneratePNGW 57	GetFileSHA384A 66
GeneratePNGW function 57	GetFileSHA384A function 66
GenerateQRCodeA 57	GetFileSHA384W 66
GenerateQRCodeA function 57	GetFileSHA384W function 66
GenerateQRCodeExA 57	GetFileSHA512A 66

GetFileSHA512A function 66	GetSHA512 function 74
GetFileSHA512W 66	GetSelectedReaderIndex 74
GetFileSHA512W function 66	GetSelectedReaderIndex function 74
GetFilesCountA 66	GetStartupA 75
GetFilesCountA function 66	GetStartupA function 75
GetFilesCountW 67	GetStartupW 75
GetFilesCountW function 67	GetStartupW function 75
GetHBitmapA 67	GetSupportSIS 75
GetHBitmapA function 67	GetSupportSIS function 75
GetHBitmapW 68	GetTextLineSize 76
GetHBitmapW function 68	GetTextLineSize function 76
GetISOCODEA 68	GetTextSize 76
GetISOCODEA function 68	GetTextSize function 76
GetISOCODEW 69	GetTextSizeEx 76
GetISOCODEW function 69	GetTextSizeEx function 76
GetMD5 69	Graphics.h 171
GetMD5 function 69	HibernateWindows 76
GetPNGA 69	HibernateWindows function 76
GetPNGA function 69	IgnoreHardwareEvents 77
GetPNGW 70	IgnoreHardwareEvents function 77
GetPNGW function 70	IgnoreServiceEvents 77
GetReaderIndexA 70	IgnoreServiceEvents function 77
GetReaderIndexA function 70	InitializeContainer 77
GetReaderIndexW 71	InitializeContainer function 77
GetReaderIndexW function 71	IsAnimatedGIFA 78
GetReaderNameA 71	IsAnimatedGIFA function 78
GetReaderNameA function 71	IsAnimatedGIFW 78
GetReaderNameLenA 72	IsAnimatedGIFW function 78
GetReaderNameLenA function 72	IsCardActivated 78
GetReaderNameLenW 72	IsCardActivated function 78
GetReaderNameLenW function 72	IsCardActivatedEx 79
GetReaderNameW 72	IsCardActivatedEx function 79
GetReaderNameW function 72	IsCardPresent 79
GetReadersCount 73	IsCardPresent function 79
GetReadersCount function 73	IsCardPresentEx 79
GetSHA1 73	IsCardPresentEx function 79
GetSHA1 function 73	IsCardStillInserted 80
GetSHA256 73	IsCardStillInserted function 80
GetSHA256 function 73	IsCardStillInsertedEx 80
GetSHA384 74	IsCardStillInsertedEx function 80
GetSHA384 function 74	IsCitrixSession 80
GetSHA512 74	IsCitrixSession function 80

IsConnectedToInternet 80	IsValidFileNameW 87
IsConnectedToInternet function 80	IsValidFileNameW function 87
IsDirectoryA 81	IsValidPathNameA 88
IsDirectoryA function 81	IsValidPathNameA function 88
IsDirectoryW 81	IsValidPathNameW 88
IsDirectoryW function 81	IsValidPathNameW function 88
IsEIDCard 81	IsWindows10 88
IsEIDCard function 81	IsWindows10 function 88
IsEIDCardEx 82	IsWindows7 89
IsEIDCardEx function 82	IsWindows7 function 89
IsEngineActive 82	IsWindows8 89
IsEngineActive function 82	IsWindows8 function 89
IsFemaleA 82	IsWindowsVista 89
IsFemaleA function 82	IsWindowsVista function 89
IsFemaleW 83	IsWindowsXP 89
IsFemaleW function 83	IsWindowsXP function 89
IsMaleA 83	IsWindowsXPSP2 90
IsMaleA function 83	IsWindowsXPSP2 function 90
IsMaleW 84	IsWow64 90
IsMaleW function 84	IsWow64 function 90
IsMediaCenter 84	LayeredWndProcA 90
IsMediaCenter function 84	LayeredWndProcA function 90
IsMetroActive 84	LayeredWndProcW 90
IsMetroActive function 84	LayeredWndProcW function 90
IsMultiTouchReady 84	LoadAddressA 90
IsMultiTouchReady function 84	LoadAddressA function 90
IsNativeWin64 85	LoadAddressW 91
IsNativeWin64 function 85	LoadAddressW function 91
IsRemoteSession 85	LoadBitmapJPG 91
IsRemoteSession function 85	LoadBitmapJPG function 91
IsSISCard 85	LoadBitmapPNG 91
IsSISCard function 85	LoadBitmapPNG function 91
IsSISCardEx 85	LoadCertificateA 91
IsSISCardEx function 85	LoadCertificateA function 91
IsTabletPC 86	LoadCertificateW 92
IsTabletPC function 86	LoadCertificateW function 92
IsUnicodeFileA 86	LoadIdentityA 92
IsUnicodeFileA function 86	LoadIdentityA function 92
IsUnicodeFileW 86	LoadIdentityW 92
IsUnicodeFileW function 86	LoadIdentityW function 92
IsValidFileNameA 87	LoadPNGResource 93
IsValidFileNameA function 87	LoadPNGResource function 93

---

LoadPhotoA 93	PointsToPixels 96
LoadPhotoA function 93	PointsToPixels function 96
LoadPhotoW 93	PortAvailable 96
LoadPhotoW function 93	PortAvailable function 96
MakeCompatibleBitmap 94	ReadAddressA 96
MakeCompatibleBitmap function 94	ReadAddressA function 96
MakeSoundFromFileA 94	ReadAddressExA 97
MakeSoundFromFileA function 94	ReadAddressExA function 97
MakeSoundFromFileW 94	ReadAddressExW 97
MakeSoundFromFileW function 94	ReadAddressExW function 97
MakeSoundFromResourceA 95	ReadAddressW 97
MakeSoundFromResourceA function 95	ReadAddressW function 97
MakeSoundFromResourceW 95	ReadAuthenticationCertificate 98
MakeSoundFromResourceW function 95	ReadAuthenticationCertificate function 98
NationalityConverter.h 172	ReadAuthenticationCertificateEx 98
Network.h 172	ReadAuthenticationCertificateEx function 98
NonRepudiationSignatureAlgoA 95	ReadBufferFromFileA 99
NonRepudiationSignatureAlgoA function 95	ReadBufferFromFileA function 99
NonRepudiationSignatureAlgoW 96	ReadBufferFromFileW 99
NonRepudiationSignatureAlgoW function 96	ReadBufferFromFileW function 99
PEidAddressA 156	ReadCaCertificate 99
PEidAddressA structure 156	ReadCaCertificate function 99
PEidAddressW 157	ReadCaCertificateEx 100
PEidAddressW structure 157	ReadCaCertificateEx function 100
PEidCertificate 157	ReadIdentityA 100
PEidCertificate structure 157	ReadIdentityA function 100
PEidIdentityA 157	ReadIdentityExA 101
PEidIdentityA structure 157	ReadIdentityExA function 101
PEidIdentityExA 159	ReadIdentityExW 101
PEidIdentityExA structure 159	ReadIdentityExW function 101
PEidIdentityExW 160	ReadIdentityW 101
PEidIdentityExW structure 160	ReadIdentityW function 101
PEidIdentityW 162	ReadNonRepudiationCertificate 102
PEidIdentityW structure 162	ReadNonRepudiationCertificate function 102
PErrorInformation 163	ReadNonRepudiationCertificateEx 102
PErrorInformation structure 163	ReadNonRepudiationCertificateEx function 102
PSISRecordA 164	ReadPhoto 102
PSISRecordA structure 164	ReadPhoto function 102
PSISRecordW 164	ReadPhotoAsBitmap 103
PSISRecordW structure 164	ReadPhotoAsBitmap function 103
PeidPicture 165	ReadPhotoAsBitmapEx 103
PeidPicture structure 165	ReadPhotoAsBitmapEx function 103

---

ReadPhotoEx 104	SaveAuthenticationCertificateExW 110
ReadPhotoEx function 104	SaveAuthenticationCertificateExW function 110
ReadRootCaCertificate 104	SaveAuthenticationCertificateW 111
ReadRootCaCertificate function 104	SaveAuthenticationCertificateW function 111
ReadRootCaCertificateEx 104	SaveCaCertificateA 111
ReadRootCaCertificateEx function 104	SaveCaCertificateA function 111
ReadRrnCertificate 105	SaveCaCertificateExW 111
ReadRrnCertificate function 105	SaveCaCertificateExW function 111
ReadRrnCertificateEx 105	SaveCaCertificateW 112
ReadRrnCertificateEx function 105	SaveCaCertificateW function 112
ReadSISCardA 106	SaveCardToToXMLStreamExA 112
ReadSISCardA function 106	SaveCardToToXMLStreamExA function 112
ReadSISCardExA 106	SaveCardToToXMLStreamExW 112
ReadSISCardExA function 106	SaveCardToToXMLStreamExW function 112
ReadSISCardExW 106	SaveCardToXmlA 113
ReadSISCardExW function 106	SaveCardToXmlA function 113
ReadSISCardW 107	SaveCardToXmlExA 113
ReadSISCardW function 107	SaveCardToXmlExA function 113
RecycleBinEmpty 107	SaveCardToXmlExW 114
RecycleBinEmpty function 107	SaveCardToXmlExW function 114
ReloadReadersList 107	SaveCardToXmlW 114
ReloadReadersList function 107	SaveCardToXmlW function 114
RemoveCallback 108	SaveContainer 114
RemoveCallback function 108	SaveContainer function 114
RemoveStartupA 108	SaveIdentityA 115
RemoveStartupA function 108	SaveIdentityA function 115
RemoveStartupW 108	SaveIdentityW 115
RemoveStartupW function 108	SaveIdentityW function 115
RestoreWindowSubclassA 109	SaveNonRepudiationCertificateA 116
RestoreWindowSubclassA function 109	SaveNonRepudiationCertificateA function 116
RestoreWindowSubclassW 109	SaveNonRepudiationCertificateExW 116
RestoreWindowSubclassW function 109	SaveNonRepudiationCertificateExW function 116
SISRecordA 166	SaveNonRepudiationCertificateW 116
SISRecordA structure 166	SaveNonRepudiationCertificateW function 116
SISRecordW 166	SavePersonCsvToStreamA 117
SISRecordW structure 166	SavePersonCsvToStreamA function 117
SaveAddressA 109	SavePersonCsvToStreamW 117
SaveAddressA function 109	SavePersonCsvToStreamW function 117
SaveAddressW 110	SavePersonToCsvA 117
SaveAddressW function 110	SavePersonToCsvA function 117
SaveAuthenticationCertificateA 110	SavePersonToCsvExA 118
SaveAuthenticationCertificateA function 110	SavePersonToCsvExA function 118



SavePersonToCsvExW 118	SendAPDU 126
SavePersonToCsvExW function 118	SendAPDU function 126
SavePersonToCsvW 119	ServerAccessible 126
SavePersonToCsvW function 119	ServerAccessible function 126
SavePhotoA 119	SetCallback 127
SavePhotoA function 119	SetCallback function 127
SavePhotoAsBitmapA 119	SetContainerCanonization 127
SavePhotoAsBitmapA function 119	SetContainerCanonization function 127
SavePhotoAsBitmapExA 120	SetContainerTimeServer 128
SavePhotoAsBitmapExA function 120	SetContainerTimeServer function 128
SavePhotoAsBitmapExW 120	SetMWCompatibility 128
SavePhotoAsBitmapExW function 120	SetMWCompatibility function 128
SavePhotoAsBitmapW 121	SetStartupA 128
SavePhotoAsBitmapW function 121	SetStartupA function 128
SavePhotoAsJpegA 121	SetStartupW 129
SavePhotoAsJpegA function 121	SetStartupW function 129
SavePhotoAsJpegExA 121	SetSupportSIS 129
SavePhotoAsJpegExA function 121	SetSupportSIS function 129
SavePhotoAsJpegExW 122	ShellCopyFileA 129
SavePhotoAsJpegExW function 122	ShellCopyFileA function 129
SavePhotoAsJpegW 122	ShellCopyFileW 130
SavePhotoAsJpegW function 122	ShellCopyFileW function 130
SavePhotoW 122	ShutdownWindows 130
SavePhotoW function 122	ShutdownWindows function 130
SaveRootCaCertificateA 123	SignPdfFile 131
SaveRootCaCertificateA function 123	SignPdfFile function 131
SaveRootCaCertificateExW 123	SignPdfFileEx 131
SaveRootCaCertificateExW function 123	SignPdfFileEx function 131
SaveRootCaCertificateW 124	StartEngine 131
SaveRootCaCertificateW function 124	StartEngine function 131
SaveRrnCertificateA 124	StopEngine 132
SaveRrnCertificateA function 124	StopEngine function 132
SaveRrnCertificateExW 124	StretchNativeBitmap 132
SaveRrnCertificateExW function 124	StretchNativeBitmap function 132
SaveRrnCertificateW 125	StripFileNameA 132
SaveRrnCertificateW function 125	StripFileNameA function 132
SelectReader 125	StripFileNameW 133
SelectReader function 125	StripFileNameW function 133
SelectReaderByNameA 125	Structs, Records, Enums 137
SelectReaderByNameA function 125	SuspendWindows 133
SelectReaderByNameW 126	SuspendWindows function 133
SelectReaderByNameW function 126	Swelio.h 173

System.h 178  
SystemInfo.h 179  
TurnMonitorOff 133  
TurnMonitorOff function 133  
TurnMonitorOn 133  
TurnMonitorOn function 133  
UpdateWindowPosition 134  
UpdateWindowPosition function 134  
VerifyContainer 134  
VerifyContainer function 134  
VerifyPinA 134  
VerifyPinA function 134  
VerifyPinExA 135  
VerifyPinExA function 135  
VerifyPinExW 135  
VerifyPinExW function 135  
VerifyPinW 136  
VerifyPinW function 136  
VerifySignature 136  
VerifySignature function 136  
WriteBufferToFileA 136  
WriteBufferToFileA function 136  
WriteBufferToFileW 137  
WriteBufferToFileW function 137  
ewtCardInsert enumeration member 139  
ewtCardRemove enumeration member 139  
ewtReadersChange enumeration member 139  
ewtUnknownEvent enumeration member 139

## F

fpreset 137  
fpreset function 137

## Q

quicol.h 179

## S

structErrorInformation 139  
structErrorInformation structure 139

## T

tagCardEventType 139  
tagCardEventType enumeration 139  
tagEidAddressA 139  
tagEidAddressA structure 139  
tagEidAddressW 140  
tagEidAddressW structure 140  
tagEidCertificate 140  
tagEidCertificate structure 140  
tagEidIdentityA 140  
tagEidIdentityA structure 140  
tagEidIdentityExA 142  
tagEidIdentityExA structure 142  
tagEidIdentityExW 143  
tagEidIdentityExW structure 143  
tagEidIdentityW 145  
tagEidIdentityW structure 145  
tagEidPicture 146  
tagEidPicture structure 146  
tagSISRecordA 147  
tagSISRecordA structure 147  
tagSISRecordW 148  
tagSISRecordW structure 148