

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

CloneFont Function	24
ContainerCertificate Function	25
ContainerEidCertificate Function	25
ContainerPickCertificate Function	25
CopyNativeBitmap Function	26
CreateCardBuffer Function	26
CreateNativeBitmap Function	26
CreateUnicodeFileA Function	27
CreateUnicodeFileW Function	27
CreateWindowsFont Function	27
CurrentIPAddressA Function	27
CurrentIPAddressW Function	28
DeactivateCard Function	28
DeactivateCardEx Function	28
DeallocateBuffer Function	29
DeallocateHWNDAs Function	29
DeallocateHWNDA Function	29
DecryptFileAESA Function	30
DecryptFileAESW Function	30
DeleteCardBuffer Function	31
DeleteToRecycleBinA Function	31
DeleteToRecycleBinW Function	31
DestroyFont Function	32
DestroyImageBuffer Function	32
DirectoryExistsA Function	32
DirectoryExistsW Function	33
DisplayCertificate Function	33
DpiY Function	33
DrawAlphaText Function	34
DrawAlphaTextRect Function	34
DrawLayeredWindow Function	34
DrawNativeBitmap Function	35
DrawTextDirect Function	35
DrawTextDirectEx Function	35
DrawTextGlow Function	35
DrawTextLine Function	36
DrawTextOutline Function	36
DrawTextRect Function	36
EmptyRecycleBin Function	36
EmToPixels Function	37
EncodeCertificate Function	37
EncodePhoto Function	37

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

GenerateBMPW Function	53
GenerateNonRepudiationSignatureA Function	54
GenerateNonRepudiationSignatureExA Function	54
GenerateNonRepudiationSignatureExW Function	55
GenerateNonRepudiationSignatureW Function	55
GeneratePNGA Function	56
GeneratePNGW Function	56
GenerateQRCodeA Function	57
GenerateQRCodeExA Function	57
GenerateQRCodeExW Function	57
GenerateQRCodeW Function	58
GetAllFiles Function	58
GetCardBufferA Function	59
GetCardBufferSize Function	59
GetCardBufferW Function	59
GetCardSerialNumber Function	60
GetCardVersion Function	60
GetEncodedCertificateSize Function	61
GetEncodedPhotoSize Function	61
GetFileMD5A Function	62
GetFileMD5W Function	62
GetFilesCountA Function	62
GetFilesCountW Function	63
GetFileSHA1A Function	63
GetFileSHA1W Function	64
GetFileSHA256A Function	64
GetFileSHA256W Function	64
GetHBitmapA Function	65
GetHBitmapW Function	65
GetISOCODEA Function	66
GetISOCODEW Function	66
GetMD5 Function	67
GetPNGA Function	67
GetPNGW Function	68
GetReaderIndexA Function	68
GetReaderIndexW Function	68
GetReaderNameA Function	69
GetReaderNameLenA Function	69
GetReaderNameLenW Function	70
GetReaderNameW Function	70
GetReadersCount Function	70
GetSelectedReaderIndex Function	71

GetSHA1 Function	71
GetSHA256 Function	71
GetStartupA Function	72
GetStartupW Function	72
GetSupportSIS Function	73
GetTextLineSize Function	73
GetTextSize Function	73
GetTextSizeEx Function	73
HibernateWindows Function	74
IgnoreHardwareEvents Function	74
IgnoreServiceEvents Function	74
InitializeContainer Function	75
IsAnimatedGIFA Function	75
IsAnimatedGIFW Function	75
IsCardActivated Function	76
IsCardActivatedEx Function	76
IsCardPresent Function	76
IsCardPresentEx Function	77
IsCardStillInserted Function	77
IsCardStillInsertedEx Function	77
IsCitrixSession Function	78
IsConnectedToInternet Function	78
IsDirectoryA Function	78
IsDirectoryW Function	78
IsEIDCard Function	79
IsEIDCardEx Function	79
IsEngineActive Function	79
IsFemaleA Function	80
IsFemaleW Function	80
IsMaleA Function	80
IsMaleW Function	81
IsMediaCenter Function	81
IsMetroActive Function	81
IsMultiTouchReady Function	82
IsNativeWin64 Function	82
IsRemoteSession Function	82
IsSISCard Function	82
IsSISCardEx Function	83
IsTabletPC Function	83
IsUnicodeFileA Function	83
IsUnicodeFileW Function	84
IsValidFileNameA Function	84

IsValidFileNameW Function	84
IsValidPathNameA Function	85
IsValidPathNameW Function	85
IsWindows10 Function	86
IsWindows7 Function	86
IsWindows8 Function	86
IsWindowsVista Function	86
IsWindowsXP Function	87
IsWindowsXPSP2 Function	87
IsWow64 Function	87
LayeredWndProcA Function	87
LayeredWndProcW Function	87
LoadBitmapJPG Function	88
LoadBitmapPNG Function	88
LoadCertificateA Function	88
LoadCertificateW Function	89
LoadIdentityA Function	89
LoadIdentityW Function	89
LoadPhotoA Function	90
LoadPhotoW Function	90
LoadPNGResource Function	90
MakeCompatibleBitmap Function	91
MakeSoundFromFileA Function	91
MakeSoundFromFileW Function	91
MakeSoundFromResourceA Function	92
MakeSoundFromResourceW Function	92
PointsToPixels Function	92
PortAvailable Function	93
ReadAddressA Function	93
ReadAddressExA Function	93
ReadAddressExW Function	94
ReadAddressW Function	94
ReadAuthenticationCertificate Function	94
ReadAuthenticationCertificateEx Function	95
ReadBufferFromFileA Function	95
ReadBufferFromFileW Function	95
ReadCaCertificate Function	96
ReadCaCertificateEx Function	96
ReadIdentityA Function	97
ReadIdentityExA Function	97
ReadIdentityExW Function	97
ReadIdentityW Function	98



ReadNonRepudiationCertificate Function	98
ReadNonRepudiationCertificateEx Function	99
ReadPhoto Function	99
ReadPhotoAsBitmap Function	99
ReadPhotoAsBitmapEx Function	100
ReadPhotoEx Function	100
ReadRootCaCertificate Function	101
ReadRootCaCertificateEx Function	101
ReadRrnCertificate Function	101
ReadRrnCertificateEx Function	102
ReadSISCardA Function	102
ReadSISCardExA Function	103
ReadSISCardExW Function	103
ReadSISCardW Function	103
RecycleBinEmpty Function	104
ReloadReadersList Function	104
RemoveCallback Function	104
RemoveStartupA Function	105
RemoveStartupW Function	105
RestoreWindowSubclassA Function	105
RestoreWindowSubclassW Function	106
SaveAuthenticationCertificateA Function	106
SaveAuthenticationCertificateExW Function	106
SaveAuthenticationCertificateW Function	107
SaveCaCertificateA Function	107
SaveCaCertificateExW Function	107
SaveCaCertificateW Function	108
SaveCardToToXMLStreamExA Function	108
SaveCardToToXMLStreamExW Function	108
SaveCardToXmlA Function	109
SaveCardToXmlExA Function	109
SaveCardToXmlExW Function	110
SaveCardToXmlW Function	110
SaveContainer Function	111
SaveIdentityA Function	111
SaveIdentityW Function	111
SaveNonRepudiationCertificateA Function	112
SaveNonRepudiationCertificateExW Function	112
SaveNonRepudiationCertificateW Function	112
SavePersonCsvToStreamA Function	113
SavePersonCsvToStreamW Function	113
SavePersonToCsvA Function	114

SavePersonToCsvExA Function	114
SavePersonToCsvExW Function	114
SavePersonToCsvW Function	115
SavePhotoA Function	115
SavePhotoAsBitmapA Function	116
SavePhotoAsBitmapExA Function	116
SavePhotoAsBitmapExW Function	116
SavePhotoAsBitmapW Function	117
SavePhotoAsJpegA Function	117
SavePhotoAsJpegExA Function	117
SavePhotoAsJpegExW Function	118
SavePhotoAsJpegW Function	118
SavePhotoW Function	119
SaveRootCaCertificateA Function	119
SaveRootCaCertificateExW Function	119
SaveRootCaCertificateW Function	120
SaveRrnCertificateA Function	120
SaveRrnCertificateExW Function	120
SaveRrnCertificateW Function	121
SelectReader Function	121
SelectReaderByNameA Function	121
SelectReaderByNameW Function	122
SendAPDU Function	122
SetCallback Function	122
SetMWCompatibility Function	123
SetStartupA Function	123
SetStartupW Function	123
SetSupportSIS Function	124
ShellCopyFileA Function	124
ShellCopyFileW Function	125
ShutdownWindows Function	125
StartEngine Function	126
StopEngine Function	126
StretchNativeBitmap Function	126
StripFileNameA Function	126
StripFileNameW Function	127
SuspendWindows Function	127
TurnMonitorOff Function	127
TurnMonitorOn Function	128
UpdateWindowPosition Function	128
VerifyPinA Function	128
VerifyPinExA Function	128

VerifyPinExW Function	129
VerifyPinW Function	129
VerifySignature Function	130
WriteBufferToFileA Function	130
WriteBufferToFileW Function	130
<b>Structs, Records, Enums</b>	<b>131</b>
structErrorInformation Structure	132
tagCardEventType Enumeration	132
tagEidAddressA Structure	133
tagEidAddressW Structure	133
tagEidCertificate Structure	133
tagEidIdentityA Structure	134
tagEidIdentityExA Structure	135
tagEidIdentityExW Structure	137
tagEidIdentityW Structure	138
tagEidPicture Structure	140
tagSISRecordA Structure	140
tagSISRecordW Structure	141
CardEventType Enumeration	142
EidAddressA Structure	142
EidAddressW Structure	142
EidCertificate Structure	143
EidIdentityA Structure	143
EidIdentityExA Structure	145
EidIdentityExW Structure	146
EidIdentityW Structure	148
EidPicture Structure	149
ErrorInformation Structure	149
PEidAddressA Structure	150
PEidAddressW Structure	150
PEidCertificate Structure	150
PEidIdentityA Structure	151
PEidIdentityExA Structure	152
PEidIdentityExW Structure	154
PEidIdentityW Structure	155
PeidPicture Structure	157
PErrorInformation Structure	157
PSISRecordA Structure	157
PSISRecordW Structure	158
SISRecordA Structure	159
SISRecordW Structure	160

<b>Files</b>	<b>177</b>
CardEvents.h	177
CardStructures.h	178
Encryption.h	180
FileOperations.h	181
Graphics.h	183
NationalityConverter.h	184
quicol.h	184
Swelio.h	184
System.h	189
SystemInfo.h	190

## **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 10)	Established communication between the card and the reader
◆	AddFileToContainer (see page 10)	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 11)	Allocates the buffer in memory
◆	AllocateDefaultHWND (see page 12)	This function creates the invisible tool window
◆	AllocateDefaultHWN (see page 12)	This function creates the invisible tool window
◆	AllocateHWN (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 13)	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 14)	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 15)	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 16)	Encrypt file using Belgian Id card
◆	CardSignCadesT (see page 17)	Sign data with eID card according to CADES-T standard
◆	CardSignCadesTEx (see page 17)	Sign data with eID card according to CADES-T standard
◆	CardSignCMS (see page 18)	Sign data with eID card according to CMS standard
◆	CardSignCMSEx (see page 18)	Sign data with eID card according to CMS standard
◆	CertSignCadesT (see page 19)	Sign data with the certificate file according to CADES-T standard
◆	CertSignCadesTData (see page 19)	Sign data with the certificate according to CADES-T standard

⇒	CertSignCadesTEx (see page 20)	Sign data with the certificate file according to CADES-T standard
⇒	CertSignCMS (see page 21)	Sign data with the certificate file according to CMS standard
⇒	CertSignCMSData (see page 21)	Sign data with the certificate according to CMS standard
⇒	CertSignCMSEx (see page 22)	Sign data with the certificate file according to CMS 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 23)	This function sets the file attributes to normal.
⇒	ClearFileAttributesW (see page 24)	This function sets the file attributes to normal.
⇒	ClearUnusedMemory (see page 24)	Clears unused memory and minimized the application memory usage
⇒	CloneFont (see page 24)	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 25)	Pick certificate to sign ASIC container
⇒	CopyNativeBitmap (see page 26)	This is function CopyNativeBitmap.
⇒	CreateCardBuffer (see page 26)	Creates XML buffer
⇒	CreateNativeBitmap (see page 26)	This is function CreateNativeBitmap.
⇒	CreateUnicodeFileA (see page 27)	Creates UNICODE file
⇒	CreateUnicodeFileW (see page 27)	Creates UNICODE file
⇒	CreateWindowsFont (see page 27)	This is function CreateWindowsFont.
⇒	CurrentIPAddressA (see page 27)	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 28)	Terminates a connection between a smart card and a reader
⇒	DeallocateBuffer (see page 29)	Deallocates the memory buffer
⇒	DeallocateHWNDA (see page 29)	This function destroys the specified window.
⇒	DeallocateHWNDA (see page 29)	This function destroys the specified window.
⇒	DecryptFileAESA (see page 30)	Decrypts file using AES algorithm.
⇒	DecryptFileAESW (see page 30)	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 31)	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 32)	Determines whether a specified directory exists.
⇒	DirectoryExistsW (see page 33)	Determines whether a specified directory exists.
⇒	DisplayCertificate (see page 33)	Displays the dialog window with certificate information
⇒	DpiY (see page 33)	This is function DpiY.

◆	DrawAlphaText (see page 34)	This is function DrawAlphaText.
◆	DrawAlphaTextRect (see page 34)	This is function DrawAlphaTextRect.
◆	DrawLayeredWindow (see page 34)	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 35)	This is function DrawTextDirectEx.
◆	DrawTextGlow (see page 35)	This is function DrawTextGlow.
◆	DrawTextLine (see page 36)	This is function DrawTextLine.
◆	DrawTextOutline (see page 36)	This is function DrawTextOutline.
◆	DrawTextRect (see page 36)	This is function DrawTextRect.
◆	EmptyRecycleBin (see page 36)	Empties the recycle bin
◆	EmToPixels (see page 37)	This is function EmToPixels.
◆	EncodeCertificate (see page 37)	Performs Base64 encoding of the certificate
◆	EncodePhoto (see page 37)	Performs Base64 encoding of the photo
◆	EncryptFileAESA (see page 38)	Encrypts file using AES algorithm.
◆	EncryptFileAESW (see page 38)	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 39)	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 40)	Creates new or overwrites existing file
◆	FileCreateRewriteW (see page 40)	Creates new or overwrites existing file
◆	FileDeleteA (see page 41)	Deletes a file from disk.
◆	FileDeleteW (see page 41)	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 42)	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 43)	Retrieves the size of a specified file.
◆	FileIsExeA (see page 44)	Checks if the file is a Windows executable
◆	FileIsExeW (see page 44)	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 45)	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 46)	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 47)	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 48)	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 49)	Writes string to the file
fpreset (see page 50)	This is function fpreset.
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 52)	Generate authentication signature
GenerateBMPA (see page 53)	Generates Windows Bitmap file with QR Code image
GenerateBMPW (see page 53)	Generates Windows Bitmap file with QR Code image
GenerateNonRepudiationSignatureA (see page 54)	Generate non repudiation signature
GenerateNonRepudiationSignatureExA (see page 54)	Generate non repudiation signature
GenerateNonRepudiationSignatureExW (see page 55)	Generate non repudiation signature
GenerateNonRepudiationSignatureW (see page 55)	Generate non repudiation signature
GeneratePNGA (see page 56)	Generates PNG file with QR Code image
GeneratePNGW (see page 56)	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 57)	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 58)	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 59)	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 59)	Gets XML or CSV information from the memory buffer
GetCardSerialNumber (see page 60)	Get the serial number of EID card
GetCardVersion (see page 60)	Get the applet version number for card in the reader with specified number
GetEncodedCertificateSize (see page 61)	Returns the size of the Base64 encoded certificate
GetEncodedPhotoSize (see page 61)	Calculates buffer size for Base64 encoded photo
GetFileMD5A (see page 62)	Gets the MD5 hash value for the file
GetFileMD5W (see page 62)	Gets the MD5 hash value for the file
GetFilesCountA (see page 62)	Calculates the number of files in the given folder
GetFilesCountW (see page 63)	Calculates the number of files in the given folder
GetFileSHA1A (see page 63)	Gets the SHA1 hash value for the file



◆	GetFileSHA1W (see page 64)	Gets the SHA1 hash value for the file
◆	GetFileSHA256A (see page 64)	Gets the SHA256 hash value for the file
◆	GetFileSHA256W (see page 64)	Gets the SHA256 hash value for the file
◆	GetHBitmapA (see page 65)	Generates Windows Bitmap in memory with QR Code image
◆	GetHBitmapW (see page 65)	Generates Windows Bitmap in memory with QR Code image
◆	GetISOCodeA (see page 66)	Returns the country ISO code based on the nationality string
◆	GetISOCodeW (see page 66)	Returns the country ISO code based on the nationality string
◆	GetMD5 (see page 67)	Gets the MD5 hash value for the content of the memory buffer
◆	GetPNGA (see page 67)	Writes PNG image to the memory buffer.
◆	GetPNGW (see page 68)	Writes PNG image to the memory buffer.
◆	GetReaderIndexA (see page 68)	Returns the zero-based reader index with specified name
◆	GetReaderIndexW (see page 68)	Returns the zero-based reader index with specified name
◆	GetReaderNameA (see page 69)	Returns the name of the card reader
◆	GetReaderNameLenA (see page 69)	Returns the length of the reader name
◆	GetReaderNameLenW (see page 70)	Returns the length of the reader name
◆	GetReaderNameW (see page 70)	Returns the name of the card reader
◆	GetReadersCount (see page 70)	Get number of card readers connected to PC
◆	GetSelectedReaderIndex (see page 71)	Returns the index of the active smart card reader
◆	GetSHA1 (see page 71)	Gets the SHA1 hash value for the content of the memory buffer
◆	GetSHA256 (see page 71)	Gets the SHA256 hash value for the content of the memory buffer
◆	GetStartupA (see page 72)	Checks if the application is registered to run when Windows starts
◆	GetStartupW (see page 72)	Checks if the application is registered to run when Windows starts
◆	GetSupportSIS (see page 73)	Checks if the SIS cards are supported by the engine
◆	GetTextLineSize (see page 73)	This is function GetTextLineSize.
◆	GetTextSize (see page 73)	This is function GetTextSize.
◆	GetTextSizeEx (see page 73)	This is function GetTextSizeEx.
◆	HibernateWindows (see page 74)	Hibernates Windows
◆	IgnoreHardwareEvents (see page 74)	Ignore USB reader insert / remove events
◆	IgnoreServiceEvents (see page 74)	Ignore smartcard service stop events when reporting readers list change
◆	InitializeContainer (see page 75)	Initializes ASIC container
◆	IsAnimatedGIFA (see page 75)	Checks if the file is an animated GIF image file
◆	IsAnimatedGIFW (see page 75)	Checks if the file is an animated GIF image file
◆	IsCardActivated (see page 76)	Checks the connection between a smart card and a reader
◆	IsCardActivatedEx (see page 76)	Checks the connection between a smart card and a reader
◆	IsCardPresent (see page 76)	Checks if the card is present in the card reader
◆	IsCardPresentEx (see page 77)	Checks if the card is present in the card reader
◆	IsCardStillInserted (see page 77)	Checks if the card is still inserted in the card reader
◆	IsCardStillInsertedEx (see page 77)	Checks if the card is still inserted in the card reader
◆	IsCitrixSession (see page 78)	Checks if application is running in Citrix session
◆	IsConnectedToInternet (see page 78)	Checks if PC is connected to Internet
◆	IsDirectoryA (see page 78)	Verifies that a path is a valid directory.
◆	IsDirectoryW (see page 78)	Verifies that a path is a valid directory.
◆	IsEIDCard (see page 79)	Check if Belgian EID card is inserted into card reader
◆	IsEIDCardEx (see page 79)	Check if Belgian EID card is inserted into card reader
◆	IsEngineActive (see page 79)	Checks if the Swelio Engine is activated
◆	IsFemaleA (see page 80)	Checks if the card owner is female
◆	IsFemaleW (see page 80)	Checks if the card owner is female

IsMaleA (see page 80)	Checks if the card owner is male
IsMaleW (see page 81)	Checks if the card owner is male
IsMediaCenter (see page 81)	Checks if the Media Center version of Windows is installed
IsMetroActive (see page 81)	Checks if metro interface is active
IsMultiTouchReady (see page 82)	Checks if the system is multi touch ready
IsNativeWin64 (see page 82)	Checks if the application is native 64 bit executable
IsRemoteSession (see page 82)	Checks if application is running in RDP session
IsSISCard (see page 82)	Check if Belgian SIS card is inserted into card reader
IsSISCardEx (see page 83)	Check if Belgian SIS card is inserted into card reader
IsTabletPC (see page 83)	Checks if the application is running on the Tablet PC
IsUnicodeFileA (see page 83)	Checks if the file is UNICODE file
IsUnicodeFileW (see page 84)	Checks if the file is UNICODE file
IsValidFileNameA (see page 84)	Checks if provided string is a valid file name
IsValidFileNameW (see page 84)	Checks if provided string is a valid file name
IsValidPathNameA (see page 85)	Checks if provided string is a valid file path
IsValidPathNameW (see page 85)	Checks if provided string is a valid file path
IsWindows10 (see page 86)	Checks if PC is running Windows 10 or better
IsWindows7 (see page 86)	Checks if PC is running Windows 7 or better
IsWindows8 (see page 86)	Checks if PC is Running Windows 8 or better
IsWindowsVista (see page 86)	Checks if PC is running Windows Vista or better
IsWindowsXP (see page 87)	Checks if PC is running Windows XP
IsWindowsXPSP2 (see page 87)	Checks if PC is running Windows XP with Service Pack 2 installed
IsWow64 (see page 87)	Checks if the 32 bit application runs on 64 bit Windows
LayeredWndProcA (see page 87)	The default window procedure for the layered window
LayeredWndProcW (see page 87)	The default window procedure for the layered window
LoadBitmapJPG (see page 88)	This is function LoadBitmapJPG.
LoadBitmapPNG (see page 88)	This is function LoadBitmapPNG.
LoadCertificateA (see page 88)	Reads the certificate from a file
LoadCertificateW (see page 89)	Reads the certificate from a file
LoadIdentityA (see page 89)	Reads the raw identity information from a file
LoadIdentityW (see page 89)	Reads the raw identity information from a file
LoadPhotoA (see page 90)	Loads photo from a file
LoadPhotoW (see page 90)	Loads photo from a file
LoadPNGResource (see page 90)	This is function LoadPNGResource.
MakeCompatibleBitmap (see page 91)	This is function MakeCompatibleBitmap.
MakeSoundFromFileA (see page 91)	Plays the wave sound from the file
MakeSoundFromFileW (see page 91)	Plays the wave sound from the file
MakeSoundFromResourceA (see page 92)	Plays the wave sound from the resource
MakeSoundFromResourceW (see page 92)	Plays the wave sound from the resource
PointsToPixels (see page 92)	This is function PointsToPixels.
PortAvailable (see page 93)	Checks if the port with specified number is available
ReadAddressA (see page 93)	Read address information from Belgian eID card
ReadAddressExA (see page 93)	Read address information from Belgian eID card
ReadAddressExW (see page 94)	Read address information from Belgian eID card
ReadAddressW (see page 94)	Read address information from Belgian eID card
ReadAuthenticationCertificate (see page 94)	Read Authentication Certificate to memory

ReadAuthenticationCertificateEx (see page 95)	Read Authentication Certificate to memory
ReadBufferFromFileA (see page 95)	Reads the content of the file to the memory buffer
ReadBufferFromFileW (see page 95)	Reads the content of the file to the memory buffer
ReadCaCertificate (see page 96)	Read Ca Certificate to memory
ReadCaCertificateEx (see page 96)	Read Ca Certificate to memory
ReadIdentityA (see page 97)	Read identity information from Belgian eID card
ReadIdentityExA (see page 97)	Read identity information from Belgian eID card
ReadIdentityExW (see page 97)	Read identity information from Belgian eID card
ReadIdentityW (see page 98)	Read identity information from Belgian eID card
ReadNonRepudiationCertificate (see page 98)	Read Non Repudiation Certificate to memory
ReadNonRepudiationCertificateEx (see page 99)	Read Non Repudiation Certificate to memory
ReadPhoto (see page 99)	Reads a photo from a card
ReadPhotoAsBitmap (see page 99)	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 100)	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 100)	Reads a photo from a card
ReadRootCaCertificate (see page 101)	Read Root Ca Certificate to memory
ReadRootCaCertificateEx (see page 101)	Read Root Ca Certificate to memory
ReadRrnCertificate (see page 101)	Read Rrn Certificate to memory
ReadRrnCertificateEx (see page 102)	Read Rrn Certificate to memory
ReadSISCardA (see page 102)	Read Belgian SIS card.
ReadSISCardExA (see page 103)	Read Belgian SIS card.
ReadSISCardExW (see page 103)	Read Belgian SIS card.
ReadSISCardW (see page 103)	Read Belgian SIS card.
RecycleBinEmpty (see page 104)	Returns TRUE if Windows Recycle Bin is empty
ReloadReadersList (see page 104)	Reloads the list of the available card readers
RemoveCallback (see page 104)	Remove callback procedure for card events
RemoveStartupA (see page 105)	Removes the application from the list of the automatically started applications
RemoveStartupW (see page 105)	Removes the application from the list of the automatically started applications
RestoreWindowSubclassA (see page 105)	Restores window standard procedure
RestoreWindowSubclassW (see page 106)	Restores window standard procedure
SaveAuthenticationCertificateA (see page 106)	Save Authentication Certificate to a file
SaveAuthenticationCertificateExW (see page 106)	Save Authentication Certificate to a file
SaveAuthenticationCertificateW (see page 107)	Save Authentication Certificate to a file
SaveCaCertificateA (see page 107)	Save Ca Certificate to a file
SaveCaCertificateExW (see page 107)	Save Ca Certificate to a file

SaveCaCertificateW (see page 108)	Save Ca Certificate to a file
SaveCardToToXMLStreamExA (see page 108)	Read eID card and save the information to XML buffer
SaveCardToToXMLStreamExW (see page 108)	Read eID card and save the information to XML buffer
SaveCardToXmlA (see page 109)	Read eID card and save the information to XML file
SaveCardToXmlExA (see page 109)	Read eID card and save the information to XML file
SaveCardToXmlExW (see page 110)	Read eID card and save the information to XML file
SaveCardToXmlW (see page 110)	Read eID card and save the information to XML file
SaveContainer (see page 111)	Save container to the file
SaveIdentityA (see page 111)	Saves indentity information to a file
SaveIdentityW (see page 111)	Saves indentity information to a file
SaveNonRepudiationCertificateA (see page 112)	Save Non Repudiation Certificate to a file
SaveNonRepudiationCertificateExW (see page 112)	Save Non Repudiation Certificate to a file
SaveNonRepudiationCertificateW (see page 112)	Save Non Repudiation Certificate to a file
SavePersonCsvToStreamA (see page 113)	Read eID card and save the identity information to CSV memory buffer
SavePersonCsvToStreamW (see page 113)	Read eID card and save the identity information to CSV memory buffer
SavePersonToCsvA (see page 114)	Read eID card and save the identity information and address to CSV file
SavePersonToCsvExA (see page 114)	Read eID card and save the identity information and address to CSV file
SavePersonToCsvExW (see page 114)	Read eID card and save the identity information and address to CSV file
SavePersonToCsvW (see page 115)	Read eID card and save the identity information and address to CSV file
SavePhotoA (see page 115)	Save photo to a file
SavePhotoAsBitmapA (see page 116)	Save the picture from the card to Windows Bitmap file Decription: 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 116)	Reads the picture from the card and saves it to Windows Bitmap file Decription: 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 116)	Reads the picture from the card and saves it to Windows Bitmap file Decription: 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 117)	Save the picture from the card to Windows Bitmap file Decription: 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 117)	Save the picture from the card to JPG file Decription: 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 117)	Save the picture from the card to JPG file Decription: 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 118)	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 118)	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 119)	Saves photo to a file
SaveRootCaCertificateA (see page 119)	Save Root Ca Certificate to a file
SaveRootCaCertificateExW (see page 119)	Save Root Ca Certificate to a file
SaveRootCaCertificateW (see page 120)	Save Root Ca Certificate to a file
SaveRrnCertificateA (see page 120)	Save RRN Certificate to a file
SaveRrnCertificateExW (see page 120)	Save RRN Certificate to a file
SaveRrnCertificateW (see page 121)	Save RRN Certificate to a file
SelectReader (see page 121)	When more than 1 reader connected, select the reader with specified number
SelectReaderByNameA (see page 121)	Select active smart card reader by providing the reader name
SelectReaderByNameW (see page 122)	Select active smart card reader by providing the reader name
SendAPDU (see page 122)	This is function SendAPDU.
SetCallback (see page 122)	Activates callback procedure for card status change event
SetMWCompatibility (see page 123)	Set the compatibility mode with the old version of the official EID MiddleWare
SetStartupA (see page 123)	Register application to run when Windows starts
SetStartupW (see page 123)	Register application to run when Windows starts
SetSupportSIS (see page 124)	Activates or deactivates SIS card support by engine
ShellCopyFileA (see page 124)	Copies file to the new location
ShellCopyFileW (see page 125)	Copies file to the new location
ShutdownWindows (see page 125)	Logs off the interactive user, shuts down the system.
StartEngine (see page 126)	Activates the Swelio Engine.
StopEngine (see page 126)	Deactivates the Swelio Engine
StretchNativeBitmap (see page 126)	This is function StretchNativeBitmap.
StripFileNameA (see page 126)	Replaces environment variable names with values
StripFileNameW (see page 127)	Replaces environment variable names with values
SuspendWindows (see page 127)	Suspends Windows
TurnMonitorOff (see page 127)	Turns the monitor off
TurnMonitorOn (see page 128)	Turns the monitor on
UpdateWindowPosition (see page 128)	Updated the window position
VerifyPinA (see page 128)	Verify PIN code
VerifyPinExA (see page 128)	Verify PIN code
VerifyPinExW (see page 129)	Verify PIN code
VerifyPinW (see page 129)	Verify PIN code
VerifySignature (see page 130)	Verifies the signature from the specified hash value.
WriteBufferToFileA (see page 130)	Writes the memory buffer to file
WriteBufferToFileW (see page 130)	Writes the memory buffer to file

---

## 1.1.1 ActivateCard Function

Established communication between the card and the reader

### C++

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

### File

Swelio.h (see page 184)

### 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

Swelio.h (see page 184)

### 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**

Swelio.h (see page 184)

**Parameters**

Parameters	Description
LPVOID container	Container handle, which is allocated by calling InitializeContainer (see page 75) 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**

System.h (see page 189)

**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. <ul style="list-style-type: none"><li>MSGFLT_ADD - Adds the message to the filter. This has the effect of allowing the message to be received.</li><li>MSGFLT_REMOVE - Removes the message from the filter. This has the effect of blocking the message.</li></ul>

**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**

FileOperations.h (see page 181)

**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**

System.h (see page 189)

**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**

System.h (see page 189)

**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 AllocateHWND A Function

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

### C++

```
HWND WINAPI AllocateHWND A(LONG_PTR method) ;
```

### File

System.h (see page 189)

### 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

System.h (see page 189)

### 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

System.h (see page 189)

### 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**

System.h (see page 189)

**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**

System.h (see page 189)

**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**

System.h (see page 189)

**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**

Graphics.h (see page 183)

---

## 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**

Graphics.h (see page 183)

---

## 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**

System.h (see page 189)

**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**

Encryption.h (see page 180)

**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**

Encryption.h (see page 180)

**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**

Encryption.h (see page 180)

**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**

Encryption.h (see page 180)

**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 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**

Swelio.h (see page 184)

**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

## 1.1.22 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**

Swelio.h (see page 184)

**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

---

## 1.1.23 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**

Swelio.h (see page 184)

**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

---

## 1.1.24 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**

Swelio.h (see page 184)

**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

---

## 1.1.25 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**

Swelio.h (see page 184)

**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

---

## 1.1.26 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**

Swelio.h (see page 184)

**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 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**

Swelio.h (see page 184)

**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



## 1.1.28 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

Swelio.h (see page 184)

### 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

## 1.1.29 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

Swelio.h (see page 184)

### 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 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

Swelio.h (see page 184)

### 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

## 1.1.31 CheckMD5 Function

Checks the MD5 hash value of the memory buffer

### C++

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

### File

Encryption.h (see page 180)

### 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

Encryption.h (see page 180)

### 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

Encryption.h (see page 180)

### 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**

FileOperations.h (see page 181)

**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**

FileOperations.h (see page 181)

**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**

System.h (see page 189)

---

## 1.1.37 CloneFont Function

This is function CloneFont.

**C++**

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

**File**

Graphics.h (see page 183)

## 1.1.38 ContainerCertificate Function

Assign certificate for signing ASIC container

### C++

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

### File

Swelio.h ([see page 184](#))

### Parameters

Parameters	Description
LPVOID container	Container handle, which is allocated by calling InitializeContainer ( <a href="#">see page 75</a> ) 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

Swelio.h ([see page 184](#))

### Parameters

Parameters	Description
LPVOID container	Container handle, which is allocated by calling InitializeContainer ( <a href="#">see page 75</a> ) 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

Swelio.h (🔗 see page 184)

Parameters

Parameters	Description
LPVOID container	Container handle, which is allocated by calling InitializeContainer (🔗 see page 75) 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

Graphics.h (🔗 see page 183)

---

# 1.1.42 CreateCardBuffer Function

Creates XML buffer

C++

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

File

Swelio.h (🔗 see page 184)

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**

Graphics.h (🔗 see page 183)

---

## 1.1.44 CreateUnicodeFileA Function

Creates UNICODE file

**C++**

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

**File**

FileOperations.h (🔗 see page 181)

**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**

FileOperations.h (🔗 see page 181)

**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**

Graphics.h (🔗 see page 183)

---

## 1.1.47 CurrentIPAddressA Function

Returns the IP address

**C++**

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

**File**

SystemInfo.h ([see page 190](#))

---

## 1.1.48 CurrentIPAddressW Function

Returns the IP address

**C++**

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

**File**

SystemInfo.h ([see page 190](#))

---

## 1.1.49 DeactivateCard Function

Terminates a connection between a smart card and a reader

**C++**

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

**File**

Swelio.h ([see page 184](#))

**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**

Swelio.h ([see page 184](#))

**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

FileOperations.h (see page 181)

### Parameters

Parameters	Description
void* buffer	The pointer to the memory buffer

### Description

DeallocateBuffer frees a memory block previously allocated with AllocateBuffer (see page 11). Use this procedure to dispose of a memory block obtained with AllocateBuffer (see page 11).

## 1.1.52 DeallocateHWND A Function

This function destroys the specified window.

### C++

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

### File

System.h (see page 189)

### 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

System.h (see page 189)

### 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**

Encryption.h (🔗 see page 180)

**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**

Encryption.h (🔗 see page 180)

**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**

Swelio.h (see page 184)

**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**

FileOperations.h (see page 181)

**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**

FileOperations.h (see page 181)

**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**

Graphics.h (🔗 see page 183)

---

## 1.1.60 DestroyImageBuffer Function

Destroys the memory buffer

**C++**

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

**File**

quricol.h (🔗 see page 184)

**Parameters**

Parameters	Description
void* buffer	The memory buffer

**Description**

Destroys the memory buffer created to hold the image returned by GetPNGA (🔗 see page 67) (Ansi) or GetPNGW (🔗 see page 68) (Unicode) functions

---

## 1.1.61 DirectoryExistsA Function

Determines whether a specified directory exists.

**C++**

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

**File**

FileOperations.h (🔗 see page 181)

**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**

FileOperations.h (🔗 see page 181)

**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**

Swelio.h (🔗 see page 184)

**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**

Graphics.h (🔗 see page 183)

---

## 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**

Graphics.h (🔗 see page 183)

---

## 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**

Graphics.h (🔗 see page 183)

---

## 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**

System.h (🔗 see page 189)

**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

Graphics.h (🔗 see page 183)

## 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

Graphics.h (🔗 see page 183)

## 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

Graphics.h (🔗 see page 183)

## 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

Graphics.h (🔗 see page 183)

---

## 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**

Graphics.h (📄 see page 183)

---

## 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**

Graphics.h (📄 see page 183)

---

## 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**

Graphics.h (📄 see page 183)

---

## 1.1.75 EmptyRecycleBin Function

Empties the recycle bin

**C++**

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

**File**

System.h (📄 see page 189)

**Description**

Removes all files from the Windows recycle bin



# 1.1.76 EmToPixels Function

This is function EmToPixels.

C++

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

File

Graphics.h (🔗 see page 183)

# 1.1.77 EncodeCertificate Function

Performs Base64 encoding of the certificate

C++

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

File

Swelio.h (🔗 see page 184)

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

Swelio.h (🔗 see page 184)

Parameters

Parameters	Description
PEidPicture photo	The pointer to EidPicture (🔗 see page 149) 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**

Encryption.h (📄 see page 180)

**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**

Encryption.h (📄 see page 180)

**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

FileOperations.h (see page 181)

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

FileOperations.h (see page 181)

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

FileOperations.h (see page 181)

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**

FileOperations.h (see page 181)

**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**

FileOperations.h (see page 181)

**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**

FileOperations.h (see page 181)

**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**

FileOperations.h (see page 181)

**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**

FileOperations.h (see page 181)

**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**

FileOperations.h (see page 181)

**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**

FileOperations.h (see page 181)

**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**

FileOperations.h (see page 181)

**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**

FileOperations.h (see page 181)

**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**

FileOperations.h (see page 181)

**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**

FileOperations.h (see page 181)

**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**

FileOperations.h (see page 181)

**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**

FileOperations.h (see page 181)

**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

FileOperations.h (see page 181)

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

FileOperations.h (see page 181)

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

FileOperations.h (see page 181)

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

FileOperations.h (see page 181)

### 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

FileOperations.h (see page 181)

### 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

FileOperations.h (see page 181)

**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**

FileOperations.h (🔗 see page 181)

**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**

FileOperations.h (🔗 see page 181)

**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

FileOperations.h (🔗 see page 181)

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

FileOperations.h (🔗 see page 181)

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

FileOperations.h (🔗 see page 181)

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**

FileOperations.h (see page 181)

**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**

FileOperations.h (see page 181)

**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**

FileOperations.h (see page 181)

**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**

FileOperations.h (see page 181)

**Parameters**

Parameters	Description
HANDLE handle	The handle of the file

LPWSTR text	The text to write
-------------	-------------------

### 1.1.112 fpreset Function

This is function fpreset.

**C++**

```
void fpreset() ;
```

**File**

System.h (🔗 see page 189)

### 1.1.113 FreeContainer Function

Deallocates ASIC container

**C++**

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

**File**

Swelio.h (🔗 see page 184)

**Parameters**

Parameters	Description
LPVOID container	Container handle, which is allocated by calling InitializeContainer (🔗 see page 75) function

**Returns**

None

**Description**

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

### 1.1.114 FullPathA Function

Gets the full path to the file based on file name

**C++**

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

**File**

FileOperations.h (🔗 see page 181)

**Parameters**

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

## 1.1.115 FullPathW Function

Gets the full path to the file based on file name

### C++

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

### File

FileOperations.h (see page 181)

### Parameters

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

## 1.1.116 GenerateAuthenticationSignatureA Function

Generate authentication signature

### C++

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

### File

Swelio.h (see page 184)

### 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

## 1.1.117 GenerateAuthenticationSignatureExA Function

Generate authentication signature

### C++

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

### File

Swelio.h (see page 184)

**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

---

## 1.1.118 GenerateAuthenticationSignatureExW Function

Generate authentication signature

**C++**

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

**File**

Swelio.h (see page 184)

**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

---

## 1.1.119 GenerateAuthenticationSignatureW Function

Generate authentication signature

**C++**

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

**File**

Swelio.h (see page 184)



**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

---

## 1.1.120 GenerateBMPA Function

Generates Windows Bitmap file with QR Code image

**C++**

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

**File**

quicol.h (see page 184)

**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 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**

quicol.h (see page 184)

**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

## 1.1.122 GenerateNonRepudiationSignatureA Function

Generate non repudiation signature

**C++**

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

**File**

Swelio.h (see page 184)

**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

## 1.1.123 GenerateNonRepudiationSignatureExA Function

Generate non repudiation signature

**C++**

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

**File**

Swelio.h (see page 184)

**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

---

## 1.1.124 GenerateNonRepudiationSignatureExW Function

Generate non repudiation signature

**C++**

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

**File**

Swelio.h (see page 184)

**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

---

## 1.1.125 GenerateNonRepudiationSignatureW Function

Generate non repudiation signature

**C++**

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

**File**

Swelio.h (see page 184)

**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

---

## 1.1.126 GeneratePNGA Function

Generates PNG file with QR Code image

**C++**

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

**File**

quicol.h (see page 184)

**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

---

## 1.1.127 GeneratePNGW Function

Generates PNG file with QR Code image

**C++**

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

**File**

quicol.h (see page 184)

**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

# 1.1.128 GenerateQRCodeA Function

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

C++

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

File

Swelio.h (see page 184)

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

# 1.1.129 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

Swelio.h (see page 184)

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

# 1.1.130 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

Swelio.h (see page 184)

**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

---

## 1.1.131 GenerateQRCodeW Function

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

**C++**

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

**File**

Swelio.h (see page 184)

**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

---

## 1.1.132 GetAllFiles Function

Returns the names of files in a specified directory.

**C++**

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

**File**

FileOperations.h (see page 181)

**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.

---

## 1.1.133 GetCardBufferA Function

Gets XML or CSV information from the memory buffer

**C++**

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

**File**

Swelio.h (see page 184)

**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

---

## 1.1.134 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**

Swelio.h (see page 184)

**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

---

## 1.1.135 GetCardBufferW Function

Gets XML or CSV information from the memory buffer

**C++**

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

**File**

Swelio.h (📄 see page 184)

**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

---

## 1.1.136 GetCardSerialNumber Function

Get the serial number of EID card

**C++**

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

**File**

Swelio.h (📄 see page 184)

**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

---

## 1.1.137 GetCardVersion Function

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

**C++**

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

**File**

Swelio.h (📄 see page 184)

**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.

---

## 1.1.138 GetEncodedCertificateSize Function

Returns the size of the Base64 encoded certificate

**C++**

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

**File**

Swelio.h ([see page 184](#))

**Parameters**

Parameters	Description
PEidCertificate 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

---

## 1.1.139 GetEncodedPhotoSize Function

Calculates buffer size for Base64 encoded photo

**C++**

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

**File**

Swelio.h ([see page 184](#))

**Parameters**

Parameters	Description
PeidPicture photo	The pointer to EidPicture ( <a href="#">see page 149</a> ) 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

---

## 1.1.140 GetFileMD5A Function

Gets the MD5 hash value for the file

### C++

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

### File

Encryption.h (see page 180)

### 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

---

## 1.1.141 GetFileMD5W Function

Gets the MD5 hash value for the file

### C++

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

### File

Encryption.h (see page 180)

### 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

---

## 1.1.142 GetFilesCountA Function

Calculates the number of files in the given folder

**C++**

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

**File**

FileOperations.h (see page 181)

**Parameters**

Parameters	Description
LPSTR folderName	The name of the folder

**Returns**

The number of files in the given folder

## 1.1.143 GetFilesCountW Function

Calculates the number of files in the given folder

**C++**

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

**File**

FileOperations.h (see page 181)

**Parameters**

Parameters	Description
LPWSTR folderName	The name of the folder

**Returns**

The number of files in the given folder

## 1.1.144 GetFileSHA1A Function

Gets the SHA1 hash value for the file

**C++**

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

**File**

Encryption.h (see page 180)

**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 SHA1 hash value for the given file

---

## 1.1.145 GetFileSHA1W Function

Gets the SHA1 hash value for the file

### C++

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

### File

Encryption.h (see page 180)

### 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

---

## 1.1.146 GetFileSHA256A Function

Gets the SHA256 hash value for the file

### C++

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

### File

Encryption.h (see page 180)

### 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

---

## 1.1.147 GetFileSHA256W Function

Gets the SHA256 hash value for the file

**C++**

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

**File**

Encryption.h (🔗 see page 180)

**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

---

## 1.1.148 GetHBitmapA Function

Generates Windows Bitmap in memory with QR Code image

**C++**

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

**File**

quricol.h (🔗 see page 184)

**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

---

## 1.1.149 GetHBitmapW Function

Generates Windows Bitmap in memory with QR Code image

**C++**

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

**File**

quricol.h (🔗 see page 184)

**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

---

## 1.1.150 GetISOCodeA Function

Returns the country ISO code based on the nationality string

**C++**

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

**File**

NationalityConverter.h (see page 184)

**Parameters**

Parameters	Description
LPCSTR nationality	The nationality string
LPSTR 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

---

## 1.1.151 GetISOCodeW Function

Returns the country ISO code based on the nationality string

**C++**

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

**File**

NationalityConverter.h (see page 184)

**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

---

## 1.1.152 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**

Encryption.h (see page 180)

**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

---

## 1.1.153 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**

quicol.h (see page 184)

**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.

# 1.1.154 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

quricol.h (see page 184)

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.

# 1.1.155 GetReaderIndexA Function

Returns the zero-based reader index with specified name

C++

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

File

Swelio.h (see page 184)

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

# 1.1.156 GetReaderIndexW Function

Returns the zero-based reader index with specified name

C++

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



File

Swelio.h (📄 see page 184)

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

---

# 1.1.157 GetReaderNameA Function

Returns the name of the card reader

C++

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

File

Swelio.h (📄 see page 184)

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

---

# 1.1.158 GetReaderNameLenA Function

Returns the length of the reader name

C++

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

File

Swelio.h (📄 see page 184)

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

# 1.1.159 GetReaderNameLenW Function

Returns the length of the reader name

**C++**

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

**File**

Swelio.h (see page 184)

**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

# 1.1.160 GetReaderNameW Function

Returns the name of the card reader

**C++**

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

**File**

Swelio.h (see page 184)

**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

# 1.1.161 GetReadersCount Function

Get number of card readers connected to PC

**C++**

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

**File**

Swelio.h (see page 184)

**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

---

## 1.1.162 GetSelectedReaderIndex Function

Returns the index of the active smart card reader

**C++**

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

**File**

Swelio.h (see page 184)

**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.

---

## 1.1.163 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**

Encryption.h (see page 180)

**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

---

## 1.1.164 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**

Encryption.h (see page 180)

**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

---

## 1.1.165 GetStartupA Function

Checks if the application is registered to run when Windows starts

**C++**

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

**File**

System.h (see page 189)

**Parameters**

Parameters	Description
LPCSTR appName	The name of the application

---

## 1.1.166 GetStartupW Function

Checks if the application is registered to run when Windows starts

**C++**

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

**File**

System.h (see page 189)

**Parameters**

Parameters	Description
LPCWSTR appName	The name of the application

---

## 1.1.167 GetSupportSIS Function

Checks if the SIS cards are supported by the engine

### C++

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

### File

Swelio.h (see page 184)

### 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 124) function. Use GetSupportSIS function to check if the SIS card support is activated.

---

## 1.1.168 GetTextLineSize Function

This is function GetTextLineSize.

### C++

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

### File

Graphics.h (see page 183)

---

## 1.1.169 GetTextSize Function

This is function GetTextSize.

### C++

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

### File

Graphics.h (see page 183)

---

## 1.1.170 GetTextSizeEx Function

This is function GetTextSizeEx.

### C++

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

File

Graphics.h (🔗 see page 183)

---

# 1.1.171 HibernateWindows Function

Hibernates Windows

C++

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

File

System.h (🔗 see page 189)

---

# 1.1.172 IgnoreHardwareEvents Function

Ignore USB reader insert / remove events

C++

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

File

Swelio.h (🔗 see page 184)

Parameters

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

Returns

None

---

# 1.1.173 IgnoreServiceEvents Function

Ignore smartcard service stop events when reporting readers list change

C++

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

File

Swelio.h (🔗 see page 184)

Parameters

Parameters	Description
BOOL value	To ignore or not the service event

Returns

None

# 1.1.174 InitializeContainer Function

Initializes ASIC container

**C++**

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

**File**

Swelio.h (🔗 see page 184)

**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

# 1.1.175 IsAnimatedGIFA Function

Checks if the file is an animated GIF image file

**C++**

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

**File**

FileOperations.h (🔗 see page 181)

**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.

# 1.1.176 IsAnimatedGIFW Function

Checks if the file is an animated GIF image file

**C++**

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

**File**

FileOperations.h (🔗 see page 181)

**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.

---

## 1.1.177 IsCardActivated Function

Checks the connection between a smart card and a reader

### C++

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

### File

Swelio.h (see page 184)

### Description

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

---

## 1.1.178 IsCardActivatedEx Function

Checks the connection between a smart card and a reader

### C++

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

### File

Swelio.h (see page 184)

### 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.

---

## 1.1.179 IsCardPresent Function

Checks if the card is present in the card reader

### C++

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

### File

Swelio.h (see page 184)

### 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



# 1.1.180 IsCardPresentEx Function

Checks if the card is present in the card reader

C++

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

File

Swelio.h (see page 184)

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

# 1.1.181 IsCardStillInserted Function

Checks if the card is still inserted in the card reader

C++

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

File

Swelio.h (see page 184)

Description

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

# 1.1.182 IsCardStillInsertedEx Function

Checks if the card is still inserted in the card reader

C++

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

File

Swelio.h (see page 184)

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

---

## 1.1.183 IsCitrixSession Function

Checks if application is running in Citrix session

### C++

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

### File

SystemInfo.h (see page 190)

---

## 1.1.184 IsConnectedToInternet Function

Checks if PC is connected to Internet

### C++

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

### File

SystemInfo.h (see page 190)

---

## 1.1.185 IsDirectoryA Function

Verifies that a path is a valid directory.

### C++

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

### File

FileOperations.h (see page 181)

### 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

---

## 1.1.186 IsDirectoryW Function

Verifies that a path is a valid directory.

### C++

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

### File

FileOperations.h (see page 181)

### 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

---

## 1.1.187 IsEIDCard Function

Check if Belgian EID card is inserted into card reader

**C++**

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

**File**

Swelio.h ([see page 184](#))

**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.

---

## 1.1.188 IsEIDCardEx Function

Check if Belgian EID card is inserted into card reader

**C++**

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

**File**

Swelio.h ([see page 184](#))

**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.

---

## 1.1.189 IsEngineActive Function

Checks if the Swelio Engine is activated

**C++**

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

**File**

Swelio.h ([see page 184](#))

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 126) function.

# 1.1.190 IsFemaleA Function

Checks if the card owner is female

C++

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

File

Swelio.h ( see page 184)

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

# 1.1.191 IsFemaleW Function

Checks if the card owner is female

C++

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

File

Swelio.h ( see page 184)

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

# 1.1.192 IsMaleA Function

Checks if the card owner is male

C++

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

File

Swelio.h (🔗 see page 184)

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

---

## 1.1.193 IsMaleW Function

Checks if the card owner is male

C++

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

File

Swelio.h (🔗 see page 184)

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

---

## 1.1.194 IsMediaCenter Function

Checks if the Media Center version of Windows is installed

C++

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

File

SystemInfo.h (🔗 see page 190)

---

## 1.1.195 IsMetroActive Function

Checks if metro interface is active

**C++**

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

**File**

SystemInfo.h ([see page 190](#))

---

## 1.1.196 IsMultiTouchReady Function

Checks if the system is multi touch ready

**C++**

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

**File**

SystemInfo.h ([see page 190](#))

---

## 1.1.197 IsNativeWin64 Function

Checks if the application is native 64 bit executable

**C++**

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

**File**

SystemInfo.h ([see page 190](#))

---

## 1.1.198 IsRemoteSession Function

Checks if application is running in RDP session

**C++**

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

**File**

SystemInfo.h ([see page 190](#))

---

## 1.1.199 IsSISCard Function

Check if Belgian SIS card is inserted into card reader

**C++**

```
BOOL WINAPI IsSISCard();
```

**File**

Swelio.h ([see page 184](#))

**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.

---

# 1.1.200 IsSISCardEx Function

Check if Belgian SIS card is inserted into card reader

**C++**

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

**File**

Swelio.h (🔗 see page 184)

**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.

---

# 1.1.201 IsTabletPC Function

Checks if the application is running on the Tablet PC

**C++**

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

**File**

SystemInfo.h (🔗 see page 190)

---

# 1.1.202 IsUnicodeFileA Function

Checks if the file is UNICODE file

**C++**

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

**File**

FileOperations.h (🔗 see page 181)

**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).

---

## 1.1.203 IsUnicodeFileW Function

Checks if the file is UNICODE file

**C++**

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

**File**

FileOperations.h (see page 181)

**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).

---

## 1.1.204 IsValidFileNameA Function

Checks if provided string is a valid file name

**C++**

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

**File**

FileOperations.h (see page 181)

**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

---

## 1.1.205 IsValidFileNameW Function

Checks if provided string is a valid file name



C++

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

File

FileOperations.h (see page 181)

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

# 1.1.206 IsValidPathNameA Function

Checks if provided string is a valid file path

C++

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

File

FileOperations.h (see page 181)

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

# 1.1.207 IsValidPathNameW Function

Checks if provided string is a valid file path

C++

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

File

FileOperations.h (see page 181)

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

---

## 1.1.208 IsWindows10 Function

Checks if PC is running Windows 10 or better

**C++**

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

**File**

SystemInfo.h ([🔗](#) see page 190)

---

## 1.1.209 IsWindows7 Function

Checks if PC is running Windows 7 or better

**C++**

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

**File**

SystemInfo.h ([🔗](#) see page 190)

---

## 1.1.210 IsWindows8 Function

Checks if PC is Running Windows 8 or better

**C++**

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

**File**

SystemInfo.h ([🔗](#) see page 190)

---

## 1.1.211 IsWindowsVista Function

Checks if PC is running Windows Vista or better

**C++**

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

**File**

SystemInfo.h ([🔗](#) see page 190)

---

## 1.1.212 IsWindowsXP Function

Checks if PC is running Windows XP

### C++

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

### File

SystemInfo.h (see page 190)

---

## 1.1.213 IsWindowsXPSP2 Function

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

### C++

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

### File

SystemInfo.h (see page 190)

---

## 1.1.214 IsWow64 Function

Checks if the 32 bit application runs on 64 bit Windows

### C++

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

### File

SystemInfo.h (see page 190)

---

## 1.1.215 LayeredWndProcA Function

The default window procedure for the layered window

### C++

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

### File

System.h (see page 189)

---

## 1.1.216 LayeredWndProcW Function

The default window procedure for the layered window

C++

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

File

System.h (🔗 see page 189)

# 1.1.217 LoadBitmapJPG Function

This is function LoadBitmapJPG.

C++

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

File

Graphics.h (🔗 see page 183)

# 1.1.218 LoadBitmapPNG Function

This is function LoadBitmapPNG.

C++

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

File

Graphics.h (🔗 see page 183)

# 1.1.219 LoadCertificateA Function

Reads the certificate from a file

C++

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

File

Swelio.h (🔗 see page 184)

Parameters

Parameters	Description
LPSTR fileName	The source file name
PEidCertificate certificate	The pointer to EidCertificate (🔗 see page 143) structure

Description

Use this function to read the certificate from the file

# 1.1.220 LoadCertificateW Function

Reads the certificate from a file

C++

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

File

Swelio.h (see page 184)

Parameters

Parameters	Description
LPWSTR fileName	The source file name
PEidCertificate certificate	The pointer to EidCertificate (see page 143) structure

Description

Use this function to read the certificate from the file

# 1.1.221 LoadIdentityA Function

Reads the raw identity information from a file

C++

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

File

Swelio.h (see page 184)

Parameters

Parameters	Description
LPSTR fileName	The name of the source file
PEidIdentityA identity	The pointer to EidIdentityA (see page 143) structure

Description

Use this function to read back the identity information stored to the file using SavIdentityA (see page 111) function

# 1.1.222 LoadIdentityW Function

Reads the raw identity information from a file

C++

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

File

Swelio.h (see page 184)

Parameters

Parameters	Description
LPWSTR fileName	The name of the source file

PEidIdentityW identity	The pointer to EidIdentityW (see page 148) structure
------------------------	--

Description

Use this function to read back the identity information stored to the file using SaveldentityW (see page 111) function

# 1.1.223 LoadPhotoA Function

Loads photo from a file

C++

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

File

Swelio.h (see page 184)

Parameters

Parameters	Description
PeidPicture photo	The pointer to EidPicture (see page 149) structure
LPSTR fileName	Destination file name

Description

Loads raw picture data from a file

# 1.1.224 LoadPhotoW Function

Loads photo from a file

C++

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

File

Swelio.h (see page 184)

Parameters

Parameters	Description
PeidPicture photo	The pointer to EidPicture (see page 149) structure
LPWSTR fileName	Destination file name

Description

Loads raw picture data from a file

# 1.1.225 LoadPNGResource Function

This is function LoadPNGResource.

C++

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

**File**

Graphics.h ([see page 183](#))

---

## 1.1.226 MakeCompatibleBitmap Function

This is function MakeCompatibleBitmap.

**C++**

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

**File**

Graphics.h ([see page 183](#))

---

## 1.1.227 MakeSoundFromFileA Function

Plays the wave sound from the file

**C++**

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

**File**

System.h ([see page 189](#))

**Parameters**

Parameters	Description
LPCSTR soundName	The name of the file

**Description**

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

---

## 1.1.228 MakeSoundFromFileW Function

Plays the wave sound from the file

**C++**

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

**File**

System.h ([see page 189](#))

**Parameters**

Parameters	Description
LPCWSTR soundName	The name of the file

**Description**

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

---

## 1.1.229 MakeSoundFromResourceA Function

Plays the wave sound from the resource

### C++

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

### File

System.h (🔗 see page 189)

### 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.

---

## 1.1.230 MakeSoundFromResourceW Function

Plays the wave sound from the resource

### C++

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

### File

System.h (🔗 see page 189)

### 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.

---

## 1.1.231 PointsToPixels Function

This is function PointsToPixels.

### C++

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

### File

Graphics.h (🔗 see page 183)



# 1.1.232 PortAvailable Function

Checks if the port with specified number is available

**C++**

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

**File**

SystemInfo.h (see page 190)

# 1.1.233 ReadAddressA Function

Read address information from Belgian eID card

**C++**

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

**File**

Swelio.h (see page 184)

**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

# 1.1.234 ReadAddressExA Function

Read address information from Belgian eID card

**C++**

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

**File**

Swelio.h (see page 184)

**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

## 1.1.235 ReadAddressExW Function

Read address information from Belgian eID card

C++

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

File

Swelio.h (see page 184)

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

## 1.1.236 ReadAddressW Function

Read address information from Belgian eID card

C++

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

File

Swelio.h (see page 184)

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

## 1.1.237 ReadAuthenticationCertificate Function

Read Authentication Certificate to memory

C++

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

File

Swelio.h (see page 184)

Parameters

Parameters	Description
certificate	The pointer to EidCertificate (see page 143) 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 143) structure

1.1.238 ReadAuthenticationCertificateEx Function

Read Authentication Certificate to memory

C++

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

File

Swelio.h (see page 184)

Parameters

Parameters	Description
int readerNumber	The zero-based index of the card reader
certificate	The pointer to EidCertificate (see page 143) 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 143) structure

1.1.239 ReadBufferFromFileA Function

Reads the content of the file to the memory buffer

C++

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

File

FileOperations.h (see page 181)

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

1.1.240 ReadBufferFromFileW Function

Reads the content of the file to the memory buffer

**C++**

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

**File**

FileOperations.h (see page 181)

**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

---

## 1.1.241 ReadCaCertificate Function

Read Ca Certificate to memory

**C++**

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

**File**

Swelio.h (see page 184)

**Parameters**

Parameters	Description
certificate	The pointer to EidCertificate (see page 143) structure

**Returns**

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

**Description**

Read Ca Certificate to EidCertificate (see page 143) structure

---

## 1.1.242 ReadCaCertificateEx Function

Read Ca Certificate to memory

**C++**

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

**File**

Swelio.h (see page 184)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader
certificate	The pointer to EidCertificate (see page 143) structure

Returns

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

Description

Read Ca Certificate to EidCertificate (see page 143) structure

# 1.1.243 ReadIdentityA Function

Read identity information from Belgian eID card

C++

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

File

Swelio.h (see page 184)

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

# 1.1.244 ReadIdentityExA Function

Read identity information from Belgian eID card

C++

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

File

Swelio.h (see page 184)

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

# 1.1.245 ReadIdentityExW Function

Read identity information from Belgian eID card

C++

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

File

Swelio.h (🔗 see page 184)

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

# 1.1.246 ReadIdentityW Function

Read identity information from Belgian eID card

C++

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

File

Swelio.h (🔗 see page 184)

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

# 1.1.247 ReadNonRepudiationCertificate Function

Read Non Repudiation Certificate to memory

C++

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

File

Swelio.h (🔗 see page 184)

Parameters

Parameters	Description
certificate	The pointer to EidCertificate (🔗 see page 143) structure

Returns

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

Description

Read Non Repudiation Certificate to EidCertificate (🔗 see page 143) structure

# 1.1.248 ReadNonRepudiationCertificateEx Function

Read Non Repudiation Certificate to memory

**C++**

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

**File**

Swelio.h (see page 184)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader
certificate	The pointer to EidCertificate (see page 143) structure

**Returns**

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

**Description**

Read Non Repudiation Certificate to EidCertificate (see page 143) structure

# 1.1.249 ReadPhoto Function

Reads a photo from a card

**C++**

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

**File**

Swelio.h (see page 184)

**Parameters**

Parameters	Description
PEidPicture photo	The pointer to EidPicture (see page 149) 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 149) structure. This structure holds the raw image bytes and the length of the image bytes array

# 1.1.250 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

Swelio.h (see page 184)

Returns

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

# 1.1.251 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

Swelio.h (see page 184)

Parameters

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

Returns

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

# 1.1.252 ReadPhotoEx Function

Reads a photo from a card

C++

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

File

Swelio.h (see page 184)

Parameters

Parameters	Description
int readerNumber	The zero-based index of the card reader.
PeidPicture photo	The pointer to EidPicture (see page 149) 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 149) structure



# 1.1.253 ReadRootCaCertificate Function

Read Root Ca Certificate to memory

C++

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

File

Swelio.h (see page 184)

Parameters

Parameters	Description
certificate	The pointer to EidCertificate (see page 143) structure

Returns

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

Description

Read Root Ca Certificate to EidCertificate (see page 143) structure

# 1.1.254 ReadRootCaCertificateEx Function

Read Root Ca Certificate to memory

C++

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

File

Swelio.h (see page 184)

Parameters

Parameters	Description
int readerNumber	The zero-based index of the card reader
certificate	The pointer to EidCertificate (see page 143) structure

Returns

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

Description

Read Root Ca Certificate to EidCertificate (see page 143) structure

# 1.1.255 ReadRrnCertificate Function

Read Rrn Certificate to memory

C++

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

File

Swelio.h (see page 184)

**Parameters**

Parameters	Description
certificate	The pointer to EidCertificate (see page 143) structure

**Returns**

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

**Description**

Read Rrn Certificate to EidCertificate (see page 143) structure

---

## 1.1.256 ReadRrnCertificateEx Function

Read Rrn Certificate to memory

**C++**

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

**File**

Swelio.h (see page 184)

**Parameters**

Parameters	Description
int readerNumber	The zero-based index of the card reader
certificate	The pointer to EidCertificate (see page 143) structure

**Returns**

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

**Description**

Read Rrn Certificate to EidCertificate (see page 143) structure

---

## 1.1.257 ReadSISCardA Function

Read Belgian SIS card.

**C++**

```
BOOL WINAPI ReadSISCardA(PSISRecordA identity);
```

**File**

Swelio.h (see page 184)

**Parameters**

Parameters	Description
PSISRecordA	The pointer to SISRecordA (see page 159) 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)

# 1.1.258 ReadSISCardExA Function

Read Belgian SIS card.

C++  
BOOL WINAPI ReadSISCardExA(int readerNumber, PSISRecordA identity);

File  
Swelio.h (see page 184)

Parameters

Parameters	Description
int readerNumber	The zero-based index of the card reader.
PSISRecordA	The pointer to SISRecordA (see page 159) 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)

# 1.1.259 ReadSISCardExW Function

Read Belgian SIS card.

C++  
BOOL WINAPI ReadSISCardExW(int readerNumber, PSISRecordW identity);

File  
Swelio.h (see page 184)

Parameters

Parameters	Description
int readerNumber	The zero-based index of the card reader.
PSISRecordW	The pointer to SISRecordW (see page 160) 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)

# 1.1.260 ReadSISCardW Function

Read Belgian SIS card.

**C++**

```
BOOL WINAPI ReadSISCardW(P SISRecordW identity);
```

**File**

Swelio.h (see page 184)

**Parameters**

Parameters	Description
P SISRecordW	The pointer to SISRecordW (see page 160) 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)

---

## 1.1.261 RecycleBinEmpty Function

Returns TRUE if Windows Recycle Bin is empty

**C++**

```
BOOL WINAPI RecycleBinEmpty();
```

**File**

SystemInfo.h (see page 190)

---

## 1.1.262 ReloadReadersList Function

Reloads the list of the available card readers

**C++**

```
void WINAPI ReloadReadersList();
```

**File**

Swelio.h (see page 184)

**Description**

When the card reader is inserted or removed you may need to reload the list of the available card readers

---

## 1.1.263 RemoveCallback Function

Remove callback procedure for card events

**C++**

```
void WINAPI RemoveCallback();
```

**File**

Swelio.h (see page 184)

**Description**

Use this function to deactivate card events callback procedure

---

## 1.1.264 RemoveStartupA Function

Removes the application from the list of the automatically started applications

**C++**

```
void WINAPI RemoveStartupA(LPCSTR appName);
```

**File**

System.h (🔗 see page 189)

**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

---

## 1.1.265 RemoveStartupW Function

Removes the application from the list of the automatically started applications

**C++**

```
void WINAPI RemoveStartupW(LPCWSTR appName);
```

**File**

System.h (🔗 see page 189)

**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

---

## 1.1.266 RestoreWindowSubclassA Function

Restores window standard procedure

**C++**

```
void WINAPI RestoreWindowSubclassA(HWND hwnd);
```

**File**

System.h (🔗 see page 189)

**Parameters**

Parameters	Description
HWND hwnd	The window handle

---

## 1.1.267 RestoreWindowSubclassW Function

Restores window standard procedure

**C++**

```
void WINAPI RestoreWindowSubclassW(HWND hwnd);
```

**File**

System.h (🔗 see page 189)

**Parameters**

Parameters	Description
HWND hwnd	The window handle

---

## 1.1.268 SaveAuthenticationCertificateA Function

Save Authentication Certificate to a file

**C++**

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

**File**

Swelio.h (🔗 see page 184)

**Parameters**

Parameters	Description
LPSTR fileName	File name to store the certificate

**Description**

Read Authentication Certificate from the card and save it to a file.

---

## 1.1.269 SaveAuthenticationCertificateExW Function

Save Authentication Certificate to a file

**C++**

```
void WINAPI SaveAuthenticationCertificateExW(LPWSTR fileName, int readerNumber);
```

**File**

Swelio.h (🔗 see page 184)

**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.

---

## 1.1.270 SaveAuthenticationCertificateW Function

Save Authentication Certificate to a file

**C++**

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

**File**

Swelio.h (see page 184)

**Parameters**

Parameters	Description
LPWSTR fileName	File name to store the certificate

**Description**

Read Authentication Certificate from the card and save it to a file.

---

## 1.1.271 SaveCaCertificateA Function

Save Ca Certificate to a file

**C++**

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

**File**

Swelio.h (see page 184)

**Parameters**

Parameters	Description
LPSTR fileName	File name to store the certificate

**Description**

Read Ca Certificate from the card and save it to a file

---

## 1.1.272 SaveCaCertificateExW Function

Save Ca Certificate to a file

**C++**

```
void WINAPI SaveCaCertificateExW(LPWSTR fileName, int readerNumber);
```

**File**

Swelio.h (see page 184)

**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

---

## 1.1.273 SaveCaCertificateW Function

Save Ca Certificate to a file

**C++**

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

**File**

Swelio.h (see page 184)

**Parameters**

Parameters	Description
LPWSTR fileName	File name to store the certificate

**Description**

Read Ca Certificate from the card and save it to a file

---

## 1.1.274 SaveCardToToXMLStreamExA Function

Read eID card and save the information to XML buffer

**C++**

```
BOOL WINAPI SaveCardToToXMLStreamExA(int readerNumber, void* buffer);
```

**File**

Swelio.h (see page 184)

**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.

---

## 1.1.275 SaveCardToToXMLStreamExW Function

Read eID card and save the information to XML buffer



**C++**

```
BOOL WINAPI SaveCardToToXMLStreamExW(int readerNumber, void* buffer);
```

**File**

Swelio.h (see page 184)

**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.

---

## 1.1.276 SaveCardToXmlA Function

Read eID card and save the information to XML file

**C++**

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

**File**

Swelio.h (see page 184)

**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.

---

## 1.1.277 SaveCardToXmlExA Function

Read eID card and save the information to XML file

**C++**

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

**File**

Swelio.h (see page 184)

**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.

---

# 1.1.278 SaveCardToXmlExW Function

Read eID card and save the information to XML file

C++

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

File

Swelio.h (🔗 see page 184)

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.

---

# 1.1.279 SaveCardToXmlW Function

Read eID card and save the information to XML file

C++

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

File

Swelio.h (🔗 see page 184)

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.

# 1.1.280 SaveContainer Function

Save container to the file

C++

```
BOOL WINAPI SaveContainer(LPVOID container, LPWSTR fileName);
```

File

Swelio.h (see page 184)

Parameters

Parameters	Description
LPVOID container	Container handle, which is allocated by calling InitializeContainer (see page 75) 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

# 1.1.281 SaveIdentityA Function

Saves identity information to a file

C++

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

File

Swelio.h (see page 184)

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 89) to read this information from the file to EidIdentityA (see page 143) structure

# 1.1.282 SaveIdentityW Function

Saves identity information to a file

C++

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

File

Swelio.h (see page 184)

**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 89) to read this information from the file to EidIdentityW (see page 148) structure

---

## 1.1.283 SaveNonRepudiationCertificateA Function

Save Non Repudiation Certificate to a file

**C++**

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

**File**

Swelio.h (see page 184)

**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

---

## 1.1.284 SaveNonRepudiationCertificateExW Function

Save Non Repudiation Certificate to a file

**C++**

```
void WINAPI SaveNonRepudiationCertificateExW(LPWSTR fileName, int readerNumber);
```

**File**

Swelio.h (see page 184)

**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

---

## 1.1.285 SaveNonRepudiationCertificateW Function

Save Non Repudiation Certificate to a file

**C++**

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

**File**

Swelio.h (see page 184)

**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

---

## 1.1.286 SavePersonCsvToStreamA Function

Read eID card and save the identity information to CSV memory buffer

**C++**

```
BOOL WINAPI SavePersonCsvToStreamA(int readerNumber, void* buffer);
```

**File**

Swelio.h (see page 184)

**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.

---

## 1.1.287 SavePersonCsvToStreamW Function

Read eID card and save the identity information to CSV memory buffer

**C++**

```
BOOL WINAPI SavePersonCsvToStreamW(int readerNumber, void* buffer);
```

**File**

Swelio.h (see page 184)

**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.

# 1.1.288 SavePersonToCsvA Function

Read eID card and save the identity information and address to CSV file

C++

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

File

Swelio.h (see page 184)

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.

# 1.1.289 SavePersonToCsvExA Function

Read eID card and save the identity information and address to CSV file

C++

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

File

Swelio.h (see page 184)

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.

# 1.1.290 SavePersonToCsvExW Function

Read eID card and save the identity information and address to CSV file

C++

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

File

Swelio.h (see page 184)

**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.

---

## 1.1.291 SavePersonToCsvW Function

Read eID card and save the identity information and address to CSV file

**C++**

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

**File**

Swelio.h (see page 184)

**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.

---

## 1.1.292 SavePhotoA Function

Save photo to a file

**C++**

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

**File**

Swelio.h (see page 184)

**Parameters**

Parameters	Description
PeidPicture photo	The pointer to EidPicture (see page 149) structure
LPSTR fileName	Destination file name

**Description**

Save the raw picture data to a file

## 1.1.293 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

Swelio.h (see page 184)

### 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

## 1.1.294 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

Swelio.h (see page 184)

### 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

## 1.1.295 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

Swelio.h (see page 184)



**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

---

## 1.1.296 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**

Swelio.h (see page 184)

**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

---

## 1.1.297 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**

Swelio.h (see page 184)

**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

---

## 1.1.298 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**

Swelio.h (see page 184)

**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

---

## 1.1.299 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**

Swelio.h (see page 184)

**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

---

## 1.1.300 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**

Swelio.h (see page 184)

**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

## 1.1.301 SavePhotoW Function

Saves photo to a file

**C++**

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

**File**

Swelio.h (see page 184)

**Parameters**

Parameters	Description
PeidPicture photo	The pointer to EidPicture (see page 149) structure
LPWSTR fileName	Destination file name

**Description**

Saves the raw picture data to a file

## 1.1.302 SaveRootCaCertificateA Function

Save Root Ca Certificate to a file

**C++**

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

**File**

Swelio.h (see page 184)

**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

## 1.1.303 SaveRootCaCertificateExW Function

Save Root Ca Certificate to a file

**C++**

```
void WINAPI SaveRootCaCertificateExW(LPWSTR fileName, int readerNumber);
```

**File**

Swelio.h (see page 184)

**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

---

## 1.1.304 SaveRootCaCertificateW Function

Save Root Ca Certificate to a file

**C++**

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

**File**

Swelio.h (see page 184)

**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

---

## 1.1.305 SaveRrnCertificateA Function

Save RRN Certificate to a file

**C++**

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

**File**

Swelio.h (see page 184)

**Parameters**

Parameters	Description
LPSTR fileName	File name to store the certificate

**Description**

Read RRN certificate from the card and save it to a file

---

## 1.1.306 SaveRrnCertificateExW Function

Save RRN Certificate to a file

**C++**

```
void WINAPI SaveRrnCertificateExW(LPWSTR fileName, int readerNumber);
```

**File**

Swelio.h (see page 184)

**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

---

## 1.1.307 SaveRrnCertificateW Function

Save RRN Certificate to a file

**C++**

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

**File**

Swelio.h (see page 184)

**Parameters**

Parameters	Description
LPWSTR fileName	File name to store the certificate

**Description**

Read RRN certificate from the card and save it to a file

---

## 1.1.308 SelectReader Function

When more than 1 reader connected, select the reader with specified number

**C++**

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

**File**

Swelio.h (see page 184)

**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.

---

## 1.1.309 SelectReaderByNameA Function

Select active smart card reader by providing the reader name

**C++**

```
BOOL WINAPI SelectReaderByNameA(LPSTR readerName) ;
```

**File**

Swelio.h (see page 184)

**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

---

## 1.1.310 SelectReaderByNameW Function

Select active smart card reader by providing the reader name

**C++**

```
BOOL WINAPI SelectReaderByNameW(LPWSTR readerName) ;
```

**File**

Swelio.h (see page 184)

**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

---

## 1.1.311 SendAPDU Function

This is function SendAPDU.

**C++**

```
BOOL WINAPI SendAPDU(int readerNumber, LPCBYTE apdu, DWORD apduLen, PCHAR result, LPDWORD len) ;
```

**File**

Swelio.h (see page 184)

---

## 1.1.312 SetCallback Function

Activates callback procedure for card status change event

C++

```
void WINAPI SetCallback(CALLBACK_HANDLER callback, LPVOID userContext);
```

File

Swelio.h (see page 184)

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

### 1.1.313 SetMWCompatibility Function

Set the compatibility mode with the old version of the official EID MiddleWare

C++

```
void WINAPI SetMWCompatibility();
```

File

Swelio.h (see page 184)

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

### 1.1.314 SetStartupA Function

Register application to run when Windows starts

C++

```
void WINAPI SetStartupA(LPCSTR appName, LPCWSTR appPath);
```

File

System.h (see page 189)

Parameters

Parameters	Description
LPCSTR appName	The name of the application
LPCWSTR appPath	The path to the application executable

### 1.1.315 SetStartupW Function

Register application to run when Windows starts

**C++**

```
void WINAPI SetStartupW(LPCWSTR appName, LPCWSTR appPath);
```

**File**

System.h (🔗 see page 189)

**Parameters**

Parameters	Description
LPCWSTR appName	The name of the application
LPCWSTR appPath	The path to the application executable

---

## 1.1.316 SetSupportSIS Function

Activates or deactivates SIS card support by engine

**C++**

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

**File**

Swelio.h (🔗 see page 184)

**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.

---

## 1.1.317 ShellCopyFileA Function

Copies file to the new location

**C++**

```
void WINAPI ShellCopyFileA(LPSTR oldName, LPSTR newName);
```

**File**

FileOperations.h (🔗 see page 181)

**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



## 1.1.318 ShellCopyFileW Function

Copies file to the new location

### C++

```
void WINAPI ShellCopyFileW(LPWSTR oldName, LPWSTR newName);
```

### File

FileOperations.h (see page 181)

### 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

## 1.1.319 ShutdownWindows Function

Logs off the interactive user, shuts down the system.

### C++

```
BOOL WINAPI ShutdownWindows(UINT flags);
```

### File

System.h (see page 189)

### 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.

---

## 1.1.320 StartEngine Function

Activates the Swelio Engine.

### C++

```
BOOL WINAPI StartEngine();
```

### File

Swelio.h (see page 184)

### 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.

---

## 1.1.321 StopEngine Function

Deactivates the Swelio Engine

### C++

```
void WINAPI StopEngine();
```

### File

Swelio.h (see page 184)

### 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.

---

## 1.1.322 StretchNativeBitmap Function

This is function StretchNativeBitmap.

### C++

```
BOOL WINAPI StretchNativeBitmap(HBITMAP src, HBITMAP dst, int srcWidth, int srcHeight, int dstWidth, int dstHeight);
```

### File

Graphics.h (see page 183)

---

## 1.1.323 StripFileNameA Function

Replaces environment variable names with values

### C++

```
void WINAPI StripFileNameA(LPCSTR fileName, LPSTR fullName);
```

File

FileOperations.h (🔗 see page 181)

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.

---

# 1.1.324 StripFileNameW Function

Replaces environment variable names with values

C++

```
void WINAPI StripFileNameW(LPCWSTR fileName, LPWSTR fullName);
```

File

FileOperations.h (🔗 see page 181)

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.

---

# 1.1.325 SuspendWindows Function

Suspends Windows

C++

```
BOOL WINAPI SuspendWindows();
```

File

System.h (🔗 see page 189)

---

# 1.1.326 TurnMonitorOff Function

Turns the monitor off

C++

```
void WINAPI TurnMonitorOff();
```

File

System.h (🔗 see page 189)

# 1.1.327 TurnMonitorOn Function

Turns the monitor on

**C++**

```
void WINAPI TurnMonitorOn();
```

**File**

System.h (see page 189)

# 1.1.328 UpdateWindowPosition Function

Updated the window position

**C++**

```
void WINAPI UpdateWindowPosition(HWND handle, int x, int y);
```

**File**

System.h (see page 189)

**Parameters**

Parameters	Description
HWND handle	The handle of the window
int x	New horizontal coordinate
int y	New vertical coordinate

# 1.1.329 VerifyPinA Function

Verify PIN code

**C++**

```
BOOL WINAPI VerifyPinA(LPSTR value);
```

**File**

Swelio.h (see page 184)

**Parameters**

Parameters	Description
LPSTR value	PIN code to verify

**Returns**

TRUE when the correct PIN code is provided; otherwise returns FALSE

# 1.1.330 VerifyPinExA Function

Verify PIN code

**C++**

```
BOOL WINAPI VerifyPinExA(int readerNumber, LPSTR value);
```

**File**

Swelio.h ([see page 184](#))

**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

---

## 1.1.331 VerifyPinExW Function

Verify PIN code

**C++**

```
BOOL WINAPI VerifyPinExW(int readerNumber, LPWSTR value);
```

**File**

Swelio.h ([see page 184](#))

**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

---

## 1.1.332 VerifyPinW Function

Verify PIN code

**C++**

```
BOOL WINAPI VerifyPinW(LPWSTR value);
```

**File**

Swelio.h ([see page 184](#))

**Parameters**

Parameters	Description
LPWSTR value	PIN code to verify

**Returns**

TRUE when the correct PIN code is provided; otherwise returns FALSE

## 1.1.333 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

Swelio.h (see page 184)

### 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

## 1.1.334 WriteBufferToFileA Function

Writes the memory buffer to file

### C++

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

### File

FileOperations.h (see page 181)

### 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.

## 1.1.335 WriteBufferToFileW Function

Writes the memory buffer to file

**C++**

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

**File**

FileOperations.h (see page 181)

**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.












## 1.2 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

**Enumerations**

	Name	Description
	tagCardEventType (see page 132)	The type of the reader event
	CardEventType (see page 142)	The type of the reader event

**Structures**

	Name	Description
	structErrorInformation (see page 132)	Information about error when operation with the card or certificate is performed
	tagEidAddressA (see page 133)	EID address information, stored on the card - ANSI version
	tagEidAddressW (see page 133)	EID address information, stored on the card - UNICODE version
	tagEidCertificate (see page 133)	Certificate, stored on EID card
	tagEidIdentityA (see page 134)	Identity information stored on EID card - ANSI version
	tagEidIdentityExA (see page 135)	Identity information stored on EID card - ANSI version
	tagEidIdentityExW (see page 137)	Identity information stored on EID card - UNICODE version
	tagEidIdentityW (see page 138)	Identity information stored on EID card - UNICODE version
	tagEidPicture (see page 140)	Raw picture data from EID card
	tagSISRecordA (see page 140)	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 (see page 141)	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 (see page 142)	EID address information, stored on the card - ANSI version
	EidAddressW (see page 142)	EID address information, stored on the card - UNICODE version
	EidCertificate (see page 143)	Certificate, stored on EID card
	EidIdentityA (see page 143)	Identity information stored on EID card - ANSI version
	EidIdentityExA (see page 145)	Identity information stored on EID card - ANSI version
	EidIdentityExW (see page 146)	Identity information stored on EID card - UNICODE version
	EidIdentityW (see page 148)	Identity information stored on EID card - UNICODE version

EidPicture (see page 149)	Raw picture data from EID card
ErrorInformation (see page 149)	Information about error when operation with the card or certificate is performed
PEidAddressA (see page 150)	EID address information, stored on the card - ANSI version
PEidAddressW (see page 150)	EID address information, stored on the card - UNICODE version
PEidCertificate (see page 150)	Certificate, stored on EID card
PEidIdentityA (see page 151)	Identity information stored on EID card - ANSI version
PEidIdentityExA (see page 152)	Identity information stored on EID card - ANSI version
PEidIdentityExW (see page 154)	Identity information stored on EID card - UNICODE version
PEidIdentityW (see page 155)	Identity information stored on EID card - UNICODE version
PeidPicture (see page 157)	Raw picture data from EID card
PErrorInformation (see page 157)	Information about error when operation with the card or certificate is performed
PSISRecordA (see page 157)	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 158)	Public information stored on Belgian SIS card - UNICODE version. The SIS card is the social security card of each Belgian resident (Belgian or foreigner)
SISRecordA (see page 159)	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 160)	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

CardStructures.h (see page 178)

## 1.2.2 tagCardEventType Enumeration

The type of the reader event

### C++

```
enum tagCardEventType {
    ewtUnknownEvent,
    ewtCardInsert,
    ewtCardRemove,
    ewtReadersChange
};
```

### File

CardEvents.h (see page 177)



**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**

CardStructures.h (see page 178)

**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**

CardStructures.h (see page 178)

**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**

CardStructures.h (see page 178)

**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**

CardStructures.h (see page 178)

**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**

CardStructures.h (see page 178)

**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**

CardStructures.h (see page 178)

**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

CardStructures.h (see page 178)

## 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

CardStructures.h (see page 178)

### 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_VALIDDBEGIN_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

CardStructures.h (see page 178)

### 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

CardStructures.h (see page 178)

### 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

CardEvents.h (🔗 see page 177)

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

CardStructures.h (🔗 see page 178)

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**

CardStructures.h (see page 178)

**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**

CardStructures.h (see page 178)

**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**

CardStructures.h (see page 178)

**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**

CardStructures.h (see page 178)

**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**

CardStructures.h (see page 178)

**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**

CardStructures.h (see page 178)

**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

CardStructures.h (see page 178)

### 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

CardStructures.h (see page 178)

# 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

CardStructures.h (see page 178)

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

CardStructures.h (see page 178)

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**

CardStructures.h (see page 178)

**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**

CardStructures.h (see page 178)

**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

CardStructures.h (see page 178)

## 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**

CardStructures.h (see page 178)

**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

CardStructures.h (see page 178)

## 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 PeidPicture Structure

Raw picture data from EID card

**C++**

```
typedef struct tagEidPicture {
    BYTE picture[EID_MAX_PICTURE_LEN+1];
    int pictureLength;
} EidPicture, * PeidPicture;
```

**File**

CardStructures.h (see page 178)

**Members**

Members	Description
BYTE picture[EID_MAX_PICTURE_LEN+1];	Picture raw data buffer
int pictureLength;	Picture raw data buffer length

## 1.2.31 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**

CardStructures.h (see page 178)

## 1.2.32 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**

CardStructures.h (see page 178)

**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

## 1.2.33 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**

CardStructures.h (see page 178)

**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

## 1.2.34 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

CardStructures.h (see page 178)

### 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

CardStructures.h (🔗 see page 178)

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 Macros

The following table lists macros in this documentation.

Macros

Name	Description
EID_MAX_BIRTHDATE_LEN (🔗 see page 162)	Maximum length of the birthdate
EID_MAX_BIRTHPLACE_LEN (🔗 see page 162)	Maximum length of the birthplace
EID_MAX_BREXITMENTION1_LEN (🔗 see page 163)	Maximum length of the BREXIT mention field
EID_MAX_BREXITMENTION2_LEN (🔗 see page 163)	Maximum length of the BREXIT mention field

EID_MAX_CARD_NUMBER_LEN (see page 163)	Maximum length of the card number field
EID_MAX_CERT_LEN (see page 163)	Maximum length of the certificate data
EID_MAX_CHIP_NUMBER_LEN (see page 163)	Maximum length of the chip number field
EID_MAX_DATE_BEGIN_LEN (see page 164)	Maximum length of the begin date field
EID_MAX_DATE_END_LEN (see page 164)	Maximum length of the end date field
EID_MAX_DATEANDCOUNTRYOFPROTECTION_LEN (see page 164)	Maximum length of the Date and country of protection field
EID_MAX_DELIVERY_MUNICIPALITY_LEN (see page 164)	Maximum length of the name of the delivery municipality
EID_MAX_DOCUMENT_TYPE_LEN (see page 165)	Maximum length of the document type field
EID_MAX_DUPLICATE_LEN (see page 165)	Maximum length of the Duplicate field
EID_MAX_FIRST_NAME1_LEN (see page 165)	Maximum length of the first name
EID_MAX_FIRST_NAME2_LEN (see page 165)	Maximum length of the first name
EID_MAX_MEMBEROFFAMILY_LEN (see page 166)	Maximum length of the Member of family field
EID_MAX_MUNICIPALITY_LEN (see page 166)	Maximum length of the municipality name field
EID_MAX_NAME_LEN (see page 166)	Maximum length of the surname
EID_MAX_NATIONAL_NUMBER_LEN (see page 166)	Maximum length of the national number
EID_MAX_NATIONALITY_LEN (see page 166)	Maximum length of the nationality
EID_MAX_NOBLE_CONDITION_LEN (see page 167)	Maximum length of the noble condition field
EID_MAX_PICTURE_LEN (see page 167)	Maximum length of the picture data
EID_MAX_REGIONALFILENUMBER_LEN (see page 167)	Maximum length of regional file number field
EID_MAX_SEX_LEN (see page 167)	Maximum length of the sex field
EID_MAX_SPECIAL_STATUS_LEN (see page 168)	Maximum length of the special status field
EID_MAX_SPECIALORGANIZATION_LEN (see page 168)	Maximum length of the Special organization field
EID_MAX_STREET_LEN (see page 168)	Maximum length of the street name field
EID_MAX_VAT1_LEN (see page 168)	Maximum length of the VAT1 field
EID_MAX_VAT2_LEN (see page 168)	Maximum length of the VAT2 field
EID_MAX_WORKPERMITTYPE_LEN (see page 169)	Maximum length of the type of the workpermit
EID_MAX_ZIP_LEN (see page 169)	Maximum length of the ZIP code field
ERR_CARD_ERROR (see page 169)	This is macro ERR_CARD_ERROR.
ERR_CERTIFICATE_ATTR (see page 169)	This is macro ERR_CERTIFICATE_ATTR.
ERR_MESSAGE_DECODE (see page 170)	This is macro ERR_MESSAGE_DECODE.
ERR_MESSAGE_ENCODE (see page 170)	This is macro ERR_MESSAGE_ENCODE.
ERR_MESSAGE_PARAM (see page 170)	This is macro ERR_MESSAGE_PARAM.
ERR_MESSAGE_UPDATE (see page 170)	This is macro ERR_MESSAGE_UPDATE.
ERR_NO_CERTIFICATE (see page 170)	This is macro ERR_NO_CERTIFICATE.
ERR_NO_DATA (see page 171)	This is macro ERR_NO_DATA.
ERR_NO_PRIVATE_KEY (see page 171)	This is macro ERR_NO_PRIVATE_KEY.
ERR_OPEN_CERTIFICATE (see page 171)	This is macro ERR_OPEN_CERTIFICATE.
ERR_SIGN_ERROR (see page 171)	This is macro ERR_SIGN_ERROR.
ERR_TIMESTAMP (see page 172)	This is macro ERR_TIMESTAMP.
ERROR_MAX_DESCRIPTION (see page 172)	Maximum error description length
FONT_BOLD (see page 172)	This is macro FONT_BOLD.
FONT_ITALIC (see page 172)	This is macro FONT_ITALIC.
FONT_NORMAL (see page 172)	This is macro FONT_NORMAL.
FONT_STRIKEOUT (see page 173)	This is macro FONT_STRIKEOUT.

FONT_UNDERLINE (see page 173)	This is macro FONT_UNDERLINE.
GetFileSHA256 (see page 173)	This is macro GetFileSHA256.
IID_PPV_ARG (see page 173)	IID_PPV_ARG(IType, ppType) IType is the type of pType ppType is the variable of type IType that will be filled RESULTS in: IID_IType, ppvType will create a compiler error if wrong level of indirection is used. macro for QueryInterface and related functions that require a IID and a (void **) this will insure that the cast is safe and appropriate on C
SaveCardToToXMLStreamEx (see page 174)	This is macro SaveCardToToXMLStreamEx.
SavePersonCsvToStream (see page 174)	This is macro SavePersonCsvToStream.
SIS_FIELD_MAX_BIRTHDATE_LEN (see page 174)	Maximum length of the birth date field
SIS_FIELD_MAX_CAPTUREDATE_LEN (see page 174)	Maximum length of the capture date field
SIS_FIELD_MAX_CARDNUMBER_LEN (see page 175)	Maximum length of the car number field
SIS_FIELD_MAX_SOCIAL_SECURITY_NUMBER_LEN (see page 175)	Maximum length of the social security number field
SIS_FIELD_MAX_VALIDBEGIN_LEN (see page 175)	Maximum length of the start validity date field
SIS_FIELD_MAX_VALIDEND_LEN (see page 175)	Maximum length of the end validity date field
SIS_MAX_CARDNAME_LEN (see page 176)	Maximum length of the card name field
SIS_MAX_FIRSTNAMES_LEN (see page 176)	Maximum length of the first name field
SIS_MAX_INITIAL_LEN (see page 176)	Maximum length of the initial field
SIS_MAX_NAME_LEN (see page 176)	Maximum length of the surname field
SIS_MAX_SEX_LEN (see page 176)	Maximum length of the sex field
WIDTHBYTES (see page 177)	This is macro WIDTHBYTES.

## 1.3.1 EID\_MAX\_BIRTHDATE\_LEN Macro

Maximum length of the birthdate

**C++**

```
#define EID_MAX_BIRTHDATE_LEN 0xc
```

**File**

CardStructures.h (see page 178)

## 1.3.2 EID\_MAX\_BIRTHPLACE\_LEN Macro

Maximum length of the birthplace

**C++**

```
#define EID_MAX_BIRTHPLACE_LEN 0x50
```

**File**

CardStructures.h (see page 178)

---

### 1.3.3 EID\_MAX\_BREXITMENTION1\_LEN Macro

Maximum length of the BREXIT mention field

**C++**

```
#define EID_MAX_BREXITMENTION1_LEN 1
```

**File**

CardStructures.h (see page 178)

---

### 1.3.4 EID\_MAX\_BREXITMENTION2\_LEN Macro

Maximum length of the BREXIT mention field

**C++**

```
#define EID_MAX_BREXITMENTION2_LEN 1
```

**File**

CardStructures.h (see page 178)

---

### 1.3.5 EID\_MAX\_CARD\_NUMBER\_LEN Macro

Maximum length of the card number field

**C++**

```
#define EID_MAX_CARD_NUMBER_LEN 0xc
```

**File**

CardStructures.h (see page 178)

---

### 1.3.6 EID\_MAX\_CERT\_LEN Macro

Maximum length of the certificate data

**C++**

```
#define EID_MAX_CERT_LEN 0x800
```

**File**

CardStructures.h (see page 178)

---

### 1.3.7 EID\_MAX\_CHIP\_NUMBER\_LEN Macro

Maximum length of the chip number field

**C++**

```
#define EID_MAX_CHIP_NUMBER_LEN 0x20
```

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

---

## 1.3.8 EID\_MAX\_DATE\_BEGIN\_LEN Macro

Maximum length of the begin date field

**C++**

```
#define EID_MAX_DATE_BEGIN_LEN 0xa
```

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

---

## 1.3.9 EID\_MAX\_DATE\_END\_LEN Macro

Maximum length of the end date field

**C++**

```
#define EID_MAX_DATE_END_LEN 0xa
```

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

---

## 1.3.10 EID\_MAX\_DATEANDCOUNTRYOFPROTECTION\_LEN Macro

Maximum length of theDate and country of protection field

**C++**

```
#define EID_MAX_DATEANDCOUNTRYOFPROTECTION_LEN 13
```

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

---

## 1.3.11 EID\_MAX\_DELIVERY\_MUNICIPALITY\_LEN Macro

Maximum length of the name of the devivery municipality

**C++**

```
#define EID_MAX_DELIVERY_MUNICIPALITY_LEN 0x50
```



**File**

CardStructures.h ([see page 178](#))

---

## 1.3.12 EID\_MAX\_DOCUMENT\_TYPE\_LEN Macro

Maximum length of the document type field

**C++**

```
#define EID_MAX_DOCUMENT_TYPE_LEN 0x2
```

**File**

CardStructures.h ([see page 178](#))

---

## 1.3.13 EID\_MAX\_DUPLICATE\_LEN Macro

Maximum length of the Duplicate field

**C++**

```
#define EID_MAX_DUPLICATE_LEN 2
```

**File**

CardStructures.h ([see page 178](#))

---

## 1.3.14 EID\_MAX\_FIRST\_NAME1\_LEN Macro

Maximum length of the first name

**C++**

```
#define EID_MAX_FIRST_NAME1_LEN 0x5f
```

**File**

CardStructures.h ([see page 178](#))

---

## 1.3.15 EID\_MAX\_FIRST\_NAME2\_LEN Macro

Maximum length of the first name

**C++**

```
#define EID_MAX_FIRST_NAME2_LEN 0x3
```

**File**

CardStructures.h ([see page 178](#))

---

## 1.3.16 EID\_MAX\_MEMBEROFFAMILY\_LEN Macro

Maximum length of the Member of family field

**C++**

```
#define EID_MAX_MEMBEROFFAMILY_LEN 1
```

**File**

CardStructures.h (see page 178)

---

## 1.3.17 EID\_MAX\_MUNICIPALITY\_LEN Macro

Maximum length of the municipality name field

**C++**

```
#define EID_MAX_MUNICIPALITY_LEN 0x43
```

**File**

CardStructures.h (see page 178)

---

## 1.3.18 EID\_MAX\_NAME\_LEN Macro

Maximum length of the surname

**C++**

```
#define EID_MAX_NAME_LEN 0x6e
```

**File**

CardStructures.h (see page 178)

---

## 1.3.19 EID\_MAX\_NATIONAL\_NUMBER\_LEN Macro

Maximum length of the national number

**C++**

```
#define EID_MAX_NATIONAL_NUMBER_LEN 0xb
```

**File**

CardStructures.h (see page 178)

---

## 1.3.20 EID\_MAX\_NATIONALITY\_LEN Macro

Maximum length of the nationality

**C++**

```
#define EID_MAX_NATIONALITY_LEN 0x55
```

**File**

CardStructures.h ([see page 178](#))

---

## 1.3.21 EID\_MAX\_NOBLE\_CONDITION\_LEN Macro

Maximum length of the noble condition field

**C++**

```
#define EID_MAX_NOBLE_CONDITION_LEN 0x32
```

**File**

CardStructures.h ([see page 178](#))

---

## 1.3.22 EID\_MAX\_PICTURE\_LEN Macro

Maximum length of the picture data

**C++**

```
#define EID_MAX_PICTURE_LEN 0x1000
```

**File**

CardStructures.h ([see page 178](#))

---

## 1.3.23 EID\_MAX\_REGIONALFILENUMBER\_LEN Macro

Maximum length of regional file number field

**C++**

```
#define EID_MAX_REGIONALFILENUMBER_LEN 18
```

**File**

CardStructures.h ([see page 178](#))

---

## 1.3.24 EID\_MAX\_SEX\_LEN Macro

Maximum length of the sex field

**C++**

```
#define EID_MAX_SEX_LEN 0x1
```

**File**

CardStructures.h ([see page 178](#))

---

## 1.3.25 EID\_MAX\_SPECIAL\_STATUS\_LEN Macro

Maximum length of the special status field

**C++**

```
#define EID_MAX_SPECIAL_STATUS_LEN 0x2
```

**File**

CardStructures.h (see page 178)

---

## 1.3.26 EID\_MAX\_SPECIALORGANIZATION\_LEN Macro

Maximum length of the Special organization field

**C++**

```
#define EID_MAX_SPECIALORGANIZATION_LEN 1
```

**File**

CardStructures.h (see page 178)

---

## 1.3.27 EID\_MAX\_STREET\_LEN Macro

Maximum length of the street name field

**C++**

```
#define EID_MAX_STREET_LEN 0x50
```

**File**

CardStructures.h (see page 178)

---

## 1.3.28 EID\_MAX\_VAT1\_LEN Macro

Maximum length of the VAT1 field

**C++**

```
#define EID_MAX_VAT1_LEN 13
```

**File**

CardStructures.h (see page 178)

---

## 1.3.29 EID\_MAX\_VAT2\_LEN Macro

Maximum length of the VAT2 field

**C++**

```
#define EID_MAX_VAT2_LEN 13
```

**File**

CardStructures.h ([see page 178](#))

---

## 1.3.30 EID\_MAX\_WORKPERMITTYPE\_LEN Macro

Maximum length of the type of the workpermit

**C++**

```
#define EID_MAX_WORKPERMITTYPE_LEN 1
```

**File**

CardStructures.h ([see page 178](#))

---

## 1.3.31 EID\_MAX\_ZIP\_LEN Macro

Maximum length of the ZIP code field

**C++**

```
#define EID_MAX_ZIP_LEN 0x4
```

**File**

CardStructures.h ([see page 178](#))

---

## 1.3.32 ERR\_CARD\_ERROR Macro

This is macro ERR\_CARD\_ERROR.

**C++**

```
#define ERR_CARD_ERROR 9
```

**File**

CardStructures.h ([see page 178](#))

---

## 1.3.33 ERR\_CERTIFICATE\_ATTR Macro

This is macro ERR\_CERTIFICATE\_ATTR.

**C++**

```
#define ERR_CERTIFICATE_ATTR 4
```

**File**

CardStructures.h ([see page 178](#))

---

## 1.3.34 ERR\_MESSAGE\_DECODE Macro

This is macro ERR\_MESSAGE\_DECODE.

**C++**

```
#define ERR_MESSAGE_DECODE 5
```

**File**

CardStructures.h (see page 178)

---

## 1.3.35 ERR\_MESSAGE\_ENCODE Macro

This is macro ERR\_MESSAGE\_ENCODE.

**C++**

```
#define ERR_MESSAGE_ENCODE 10
```

**File**

CardStructures.h (see page 178)

---

## 1.3.36 ERR\_MESSAGE\_PARAM Macro

This is macro ERR\_MESSAGE\_PARAM.

**C++**

```
#define ERR_MESSAGE_PARAM 6
```

**File**

CardStructures.h (see page 178)

---

## 1.3.37 ERR\_MESSAGE\_UPDATE Macro

This is macro ERR\_MESSAGE\_UPDATE.

**C++**

```
#define ERR_MESSAGE_UPDATE 7
```

**File**

CardStructures.h (see page 178)

---

## 1.3.38 ERR\_NO\_CERTIFICATE Macro

This is macro ERR\_NO\_CERTIFICATE.

**C++**

```
#define ERR_NO_CERTIFICATE 2
```

**File**

CardStructures.h ([see page 178](#))

---

## 1.3.39 ERR\_NO\_DATA Macro

This is macro ERR\_NO\_DATA.

**C++**

```
#define ERR_NO_DATA 1
```

**File**

CardStructures.h ([see page 178](#))

---

## 1.3.40 ERR\_NO\_PRIVATE\_KEY Macro

This is macro ERR\_NO\_PRIVATE\_KEY.

**C++**

```
#define ERR_NO_PRIVATE_KEY 3
```

**File**

CardStructures.h ([see page 178](#))

---

## 1.3.41 ERR\_OPEN\_CERTIFICATE Macro

This is macro ERR\_OPEN\_CERTIFICATE.

**C++**

```
#define ERR_OPEN_CERTIFICATE 8
```

**File**

CardStructures.h ([see page 178](#))

---

## 1.3.42 ERR\_SIGN\_ERROR Macro

This is macro ERR\_SIGN\_ERROR.

**C++**

```
#define ERR_SIGN_ERROR 12
```

**File**

CardStructures.h ([see page 178](#))

---

## 1.3.43 ERR\_TIMESTAMP Macro

This is macro ERR\_TIMESTAMP.

**C++**

```
#define ERR_TIMESTAMP 11
```

**File**

CardStructures.h ([↗](#) see page 178)

---

## 1.3.44 ERROR\_MAX\_DESCRIPTION Macro

Maximum error description length

**C++**

```
#define ERROR_MAX_DESCRIPTION 254
```

**File**

CardStructures.h ([↗](#) see page 178)

---

## 1.3.45 FONT\_BOLD Macro

This is macro FONT\_BOLD.

**C++**

```
#define FONT_BOLD 0x01
```

**File**

Graphics.h ([↗](#) see page 183)

---

## 1.3.46 FONT\_ITALIC Macro

This is macro FONT\_ITALIC.

**C++**

```
#define FONT_ITALIC 0x02
```

**File**

Graphics.h ([↗](#) see page 183)

---

## 1.3.47 FONT\_NORMAL Macro

This is macro FONT\_NORMAL.



**C++**

```
#define FONT_NORMAL 0x00
```

**File**

Graphics.h ([see page 183](#))

---

## 1.3.48 FONT\_STRIKEOUT Macro

This is macro FONT\_STRIKEOUT.

**C++**

```
#define FONT_STRIKEOUT 0x08
```

**File**

Graphics.h ([see page 183](#))

---

## 1.3.49 FONT\_UNDERLINE Macro

This is macro FONT\_UNDERLINE.

**C++**

```
#define FONT_UNDERLINE 0x04
```

**File**

Graphics.h ([see page 183](#))

---

## 1.3.50 GetFileSHA256 Macro

This is macro GetFileSHA256.

**C++**

```
#define GetFileSHA256 GetFileSHA256A
```

**File**

Encryption.h ([see page 180](#))

---

## 1.3.51 IID\_PPV\_ARG Macro

IID\_PPV\_ARG(IType, ppType) IType is the type of pType ppType is the variable of type IType that will be filled

RESULTS in: IID\_IType, ppvType will create a compiler error if wrong level of indirection is used.

macro for QueryInterface and related functions that require a IID and a (void \*\*) this will insure that the cast is safe and appropriate on C

**C++**

```
#define IID_PPV_ARG(IType, ppType) &IID_##IType, (void**)(ppType))
```

**File**

FileOperations.h ([↗](#) see page 181)

---

## 1.3.52 SaveCardToToXMLStreamEx Macro

This is macro SaveCardToToXMLStreamEx.

**C++**

```
#define SaveCardToToXMLStreamEx SaveCardToToXMLStreamExA
```

**File**

Swelio.h ([↗](#) see page 184)

---

## 1.3.53 SavePersonCsvToStream Macro

This is macro SavePersonCsvToStream.

**C++**

```
#define SavePersonCsvToStream SavePersonCsvToStreamA
```

**File**

Swelio.h ([↗](#) see page 184)

---

## 1.3.54 SIS\_FIELD\_MAX\_BIRTHDATE\_LEN Macro

Maximum length of the birth date field

**C++**

```
#define SIS_FIELD_MAX_BIRTHDATE_LEN 0x8
```

**File**

CardStructures.h ([↗](#) see page 178)

---

## 1.3.55 SIS\_FIELD\_MAX\_CAPTUREDATE\_LEN Macro

Maximum length of the capture date field

**C++**

```
#define SIS_FIELD_MAX_CAPTUREDATE_LEN 0x8
```

**File**

CardStructures.h ([↗](#) see page 178)

---

## 1.3.56 SIS\_FIELD\_MAX\_CARDNUMBER\_LEN Macro

Maximum length of the car number field

**C++**

```
#define SIS_FIELD_MAX_CARDNUMBER_LEN 0xa
```

**File**

CardStructures.h (see page 178)

---

## 1.3.57 SIS\_FIELD\_MAX\_SOCIAL\_SECURITY\_NUMBER\_LEN Macro

Maximum length of the social security number field

**C++**

```
#define SIS_FIELD_MAX_SOCIAL_SECURITY_NUMBER_LEN 0xb
```

**File**

CardStructures.h (see page 178)

---

## 1.3.58 SIS\_FIELD\_MAX\_VALIDBEGIN\_LEN Macro

Maximum length of the start validity date field

**C++**

```
#define SIS_FIELD_MAX_VALIDBEGIN_LEN 0x8
```

**File**

CardStructures.h (see page 178)

---

## 1.3.59 SIS\_FIELD\_MAX\_VALIDEND\_LEN Macro

Maximum length of the end validity date field

**C++**

```
#define SIS_FIELD_MAX_VALIDEND_LEN 0x8
```

**File**

CardStructures.h (see page 178)

---

## 1.3.60 SIS\_MAX\_CARDNAME\_LEN Macro

Maximum length of the card name field

**C++**

```
#define SIS_MAX_CARDNAME_LEN 0x6
```

**File**

CardStructures.h (see page 178)

---

## 1.3.61 SIS\_MAX\_FIRSTNAMES\_LEN Macro

Maximum length of the first name field

**C++**

```
#define SIS_MAX_FIRSTNAMES_LEN 0x18
```

**File**

CardStructures.h (see page 178)

---

## 1.3.62 SIS\_MAX\_INITIAL\_LEN Macro

Maximum length of the initial field

**C++**

```
#define SIS_MAX_INITIAL_LEN 0x1
```

**File**

CardStructures.h (see page 178)

---

## 1.3.63 SIS\_MAX\_NAME\_LEN Macro

Maximum length of the surname field

**C++**

```
#define SIS_MAX_NAME_LEN 0x30
```

**File**

CardStructures.h (see page 178)

---

## 1.3.64 SIS\_MAX\_SEX\_LEN Macro

Maximum length of the sex field

C++

```
#define SIS_MAX_SEX_LEN 0x1
```

File

CardStructures.h (see page 178)

### 1.3.65 WIDTHBYTES Macro

This is macro WIDTHBYTES.

C++

```
#define WIDTHBYTES(i) (((i) + 31) / 32 * 4)
```

File

Graphics.h (see page 183)

## 1.4 Files

The following table lists files in this documentation.

Files

Name	Description
CardEvents.h (see page 177)	The definition of the possible card events
CardStructures.h (see page 178)	The definition of the information storage structures
Encryption.h (see page 180)	Encryption and decryption operations
FileOperations.h (see page 181)	Files and folders manipulations
Graphics.h (see page 183)	This is file Graphics.h.
NationalityConverter.h (see page 184)	Nationality to ISO code cenvter
quicol.h (see page 184)	QR Code generator
Swelio.h (see page 184)	Belgian electronic Id card access engine
System.h (see page 189)	Windows related routines
SystemInfo.h (see page 190)	Routines for querying information about operating system

### 1.4.1 CardEvents.h

The definition of the possible card events

Enumerations

	Name	Description
📱	tagCardEventType (see page 132)	The type of the reader event
	CardEventType (see page 142)	The type of the reader event

## 1.4.2 CardStructures.h












The definition of the information storage structures

### Macros

Name	Description
EID_MAX_BIRTHDATE_LEN (see page 162)	Maximum length of the birthdate
EID_MAX_BIRTHPLACE_LEN (see page 162)	Maximum length of the birthplace
EID_MAX_BREXITMENTION1_LEN (see page 163)	Maximum length of the BREXIT mention field
EID_MAX_BREXITMENTION2_LEN (see page 163)	Maximum length of the BREXIT mention field
EID_MAX_CARD_NUMBER_LEN (see page 163)	Maximum length of the card number field
EID_MAX_CERT_LEN (see page 163)	Maximum length of the certificate data
EID_MAX_CHIP_NUMBER_LEN (see page 163)	Maximum length of the chip number field
EID_MAX_DATE_BEGIN_LEN (see page 164)	Maximum length of the begin date field
EID_MAX_DATE_END_LEN (see page 164)	Maximum length of the end date field
EID_MAX_DATEANDCOUNTRYOFPROTECTION_LEN (see page 164)	Maximum length of theDate and country of protection field
EID_MAX_DELIVERY_MUNICIPALITY_LEN (see page 164)	Maximum length of the name of the delivery municipality
EID_MAX_DOCUMENT_TYPE_LEN (see page 165)	Maximum length of the document type field
EID_MAX_DUPLICATE_LEN (see page 165)	Maximum length of the Duplicate field
EID_MAX_FIRST_NAME1_LEN (see page 165)	Maximum length of the first name
EID_MAX_FIRST_NAME2_LEN (see page 165)	Maximum length of the first name
EID_MAX_MEMBEROFFAMILY_LEN (see page 166)	Maximum length of the Member of family field
EID_MAX_MUNICIPALITY_LEN (see page 166)	Maximum length of the municipality name field
EID_MAX_NAME_LEN (see page 166)	Maximum length of the surname
EID_MAX_NATIONAL_NUMBER_LEN (see page 166)	Maximum length of the national number
EID_MAX_NATIONALITY_LEN (see page 166)	Maximum length of the nationality
EID_MAX_NOBLE_CONDITION_LEN (see page 167)	Maximum length of the noble condition field
EID_MAX_PICTURE_LEN (see page 167)	Maximum length of the picture data
EID_MAX_REGIONALFILENUMBER_LEN (see page 167)	Maximum length of regional file number field
EID_MAX_SEX_LEN (see page 167)	Maximum length of the sex field
EID_MAX_SPECIAL_STATUS_LEN (see page 168)	Maximum length of the special status field
EID_MAX_SPECIALORGANIZATION_LEN (see page 168)	Maximum length of the Special organization field
EID_MAX_STREET_LEN (see page 168)	Maximum length of the street name field
EID_MAX_VAT1_LEN (see page 168)	Maximum length of the VAT1 field
EID_MAX_VAT2_LEN (see page 168)	Maximum length of the VAT2 field
EID_MAX_WORKPERMITTYPE_LEN (see page 169)	Maximum length of the type of the workpermit
EID_MAX_ZIP_LEN (see page 169)	Maximum length of the ZIP code field
ERR_CARD_ERROR (see page 169)	This is macro ERR_CARD_ERROR.
ERR_CERTIFICATE_ATTR (see page 169)	This is macro ERR_CERTIFICATE_ATTR.
ERR_MESSAGE_DECODE (see page 170)	This is macro ERR_MESSAGE_DECODE.

ERR_MESSAGE_ENCODE (see page 170)	This is macro ERR_MESSAGE_ENCODE.
ERR_MESSAGE_PARAM (see page 170)	This is macro ERR_MESSAGE_PARAM.
ERR_MESSAGE_UPDATE (see page 170)	This is macro ERR_MESSAGE_UPDATE.
ERR_NO_CERTIFICATE (see page 170)	This is macro ERR_NO_CERTIFICATE.
ERR_NO_DATA (see page 171)	This is macro ERR_NO_DATA.
ERR_NO_PRIVATE_KEY (see page 171)	This is macro ERR_NO_PRIVATE_KEY.
ERR_OPEN_CERTIFICATE (see page 171)	This is macro ERR_OPEN_CERTIFICATE.
ERR_SIGN_ERROR (see page 171)	This is macro ERR_SIGN_ERROR.
ERR_TIMESTAMP (see page 172)	This is macro ERR_TIMESTAMP.
ERROR_MAX_DESCRIPTION (see page 172)	Maximum error description length
SIS_FIELD_MAX_BIRTHDATE_LEN (see page 174)	Maximum length of the birth date field
SIS_FIELD_MAX_CAPTUREDATE_LEN (see page 174)	Maximum length of the capture date field
SIS_FIELD_MAX_CARDNUMBER_LEN (see page 175)	Maximum length of the car number field
SIS_FIELD_MAX_SOCIAL_SECURITY_NUMBER_LEN (see page 175)	Maximum length of the social security number field
SIS_FIELD_MAX_VALIDBEGIN_LEN (see page 175)	Maximum length of the start validity date field
SIS_FIELD_MAX_VALIDEND_LEN (see page 175)	Maximum length of the end validity date field
SIS_MAX_CARDNAME_LEN (see page 176)	Maximum length of the card name field
SIS_MAX_FIRSTNAMES_LEN (see page 176)	Maximum length of the first name field
SIS_MAX_INITIAL_LEN (see page 176)	Maximum length of the initial field
SIS_MAX_NAME_LEN (see page 176)	Maximum length of the surname field
SIS_MAX_SEX_LEN (see page 176)	Maximum length of the sex field

## Structures

	Name	Description
	structErrorInformation (see page 132)	Information about error when operation with the card or certificate is performed
	tagEidAddressA (see page 133)	EID address information, stored on the card - ANSI version
	tagEidAddressW (see page 133)	EID address information, stored on the card - UNICODE version
	tagEidCertificate (see page 133)	Certificate, stored on EID card
	tagEidIdentityA (see page 134)	Identity information stored on EID card - ANSI version
	tagEidIdentityExA (see page 135)	Identity information stored on EID card - ANSI version
	tagEidIdentityExW (see page 137)	Identity information stored on EID card - UNICODE version
	tagEidIdentityW (see page 138)	Identity information stored on EID card - UNICODE version
	tagEidPicture (see page 140)	Raw picture data from EID card
	tagSISRecordA (see page 140)	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 (see page 141)	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 (see page 142)	EID address information, stored on the card - ANSI version
	EidAddressW (see page 142)	EID address information, stored on the card - UNICODE version
	EidCertificate (see page 143)	Certificate, stored on EID card
	EidIdentityA (see page 143)	Identity information stored on EID card - ANSI version
	EidIdentityExA (see page 145)	Identity information stored on EID card - ANSI version
	EidIdentityExW (see page 146)	Identity information stored on EID card - UNICODE version
	EidIdentityW (see page 148)	Identity information stored on EID card - UNICODE version
	EidPicture (see page 149)	Raw picture data from EID card

ErrorInformation (see page 149)	Information about error when operation with the card or certificate is performed
PEidAddressA (see page 150)	EID address information, stored on the card - ANSI version
PEidAddressW (see page 150)	EID address information, stored on the card - UNICODE version
PEidCertificate (see page 150)	Certificate, stored on EID card
PEidIdentityA (see page 151)	Identity information stored on EID card - ANSI version
PEidIdentityExA (see page 152)	Identity information stored on EID card - ANSI version
PEidIdentityExW (see page 154)	Identity information stored on EID card - UNICODE version
PEidIdentityW (see page 155)	Identity information stored on EID card - UNICODE version
PeidPicture (see page 157)	Raw picture data from EID card
PErrorInformation (see page 157)	Information about error when operation with the card or certificate is performed
PSISRecordA (see page 157)	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 158)	Public information stored on Belgian SIS card - UNICODE version. The SIS card is the social security card of each Belgian resident (Belgian or foreigner)
SISRecordA (see page 159)	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 160)	Public information stored on Belgian SIS card - UNICODE version. The SIS card is the social security card of each Belgian resident (Belgian or foreigner)

## 1.4.3 Encryption.h

Encryption and decryption operations

### Functions

	Name	Description
◆	CardDecryptFileA (see page 15)	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 16)	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 30)	Decrypts file using AES algorithm.
◆	EncryptFileAESA (see page 38)	Encrypts file using AES algorithm.
◆	EncryptFileAESW (see page 38)	Encrypts file using AES algorithm.
◆	GetFileMD5A (see page 62)	Gets the MD5 hash value for the file
◆	GetFileMD5W (see page 62)	Gets the MD5 hash value for the file
◆	GetFileSHA1A (see page 63)	Gets the SHA1 hash value for the file
◆	GetFileSHA1W (see page 64)	Gets the SHA1 hash value for the file
◆	GetFileSHA256A (see page 64)	Gets the SHA256 hash value for the file
◆	GetFileSHA256W (see page 64)	Gets the SHA256 hash value for the file
◆	GetMD5 (see page 67)	Gets the MD5 hash value for the content of the memory buffer
◆	GetSHA1 (see page 71)	Gets the SHA1 hash value for the content of the memory buffer



	GetSHA256 ( <a href="#">see page 71</a> )	Gets the SHA256 hash value for the content of the memory buffer
---	---	---






























**Macros**

Name	Description
GetFileSHA256 ( <a href="#">see page 173</a> )	This is macro GetFileSHA256.

## 1.4.4 FileOperations.h

Files and folders manipulations

**Functions**

	Name	Description
	AllocateBuffer ( <a href="#">see page 11</a> )	Allocates the buffer in memory
	ClearFileAttributesA ( <a href="#">see page 23</a> )	This function sets the file attributes to normal.
	ClearFileAttributesW ( <a href="#">see page 24</a> )	This function sets the file attributes to normal.
	CreateUnicodeFileA ( <a href="#">see page 27</a> )	Creates UNICODE file
	CreateUnicodeFileW ( <a href="#">see page 27</a> )	Creates UNICODE file
	DeallocateBuffer ( <a href="#">see page 29</a> )	Deallocates the memory buffer
	DeleteToRecycleBinA ( <a href="#">see page 31</a> )	Deletes file to the Windows Recycle Bin
	DeleteToRecycleBinW ( <a href="#">see page 31</a> )	Deletes file to WIndows Recycle Bin
	DirectoryExistsA ( <a href="#">see page 32</a> )	Determines whether a specified directory exists.
	DirectoryExistsW ( <a href="#">see page 33</a> )	Determines whether a specified directory exists.
	FileCloseA ( <a href="#">see page 39</a> )	Concludes input/output (I/O) to a file opened using the FileCreateRewrite function.
	FileCloseW ( <a href="#">see page 39</a> )	Concludes input/output (I/O) to a file opened using the FileCreateRewrite function.
	FileCopyA ( <a href="#">see page 39</a> )	The CopyFile function copies an existing file to a new file.
	FileCopyW ( <a href="#">see page 40</a> )	The CopyFile function copies an existing file to a new file.
	FileCreateRewriteA ( <a href="#">see page 40</a> )	Creates new or overwrites existing file
	FileCreateRewriteW ( <a href="#">see page 40</a> )	Creates new or overwrites existing file
	FileDeleteA ( <a href="#">see page 41</a> )	Deletes a file from disk.
	FileDeleteW ( <a href="#">see page 41</a> )	Deletes a file from disk.
	FileExistsA ( <a href="#">see page 42</a> )	Tests whether a specified file exists.
	FileExistsW ( <a href="#">see page 42</a> )	Tests whether a specified file exists.
	FileExtensionIsA ( <a href="#">see page 42</a> )	Checks the file extension
	FileExtensionIsW ( <a href="#">see page 43</a> )	Checks the file extension
	FileGetSizeA ( <a href="#">see page 43</a> )	Retrieves the size of a specified file.
	FileGetSizeW ( <a href="#">see page 43</a> )	Retrieves the size of a specified file.
	FileIsExeA ( <a href="#">see page 44</a> )	Checks if the file is a Windows executable
	FileIsExeW ( <a href="#">see page 44</a> )	Checks if the file is a Windows executable
	FileIsIconA ( <a href="#">see page 45</a> )	Checks if the file is a Windows icon (.ico) file
	FileIsIconW ( <a href="#">see page 45</a> )	Checks if the file is a Windows icon (.ico) file
	FileIsImageA ( <a href="#">see page 45</a> )	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 46)	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 47)	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 48)	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 49)	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 58)	Returns the names of files in a specified directory.
GetFilesCountA (see page 62)	Calculates the number of files in the given folder
GetFilesCountW (see page 63)	Calculates the number of files in the given folder
IsAnimatedGIFA (see page 75)	Checks if the file is an animated GIF image file
IsAnimatedGIFW (see page 75)	Checks if the file is an animated GIF image file
IsDirectoryA (see page 78)	Verifies that a path is a valid directory.
IsDirectoryW (see page 78)	Verifies that a path is a valid directory.
IsUnicodeFileA (see page 83)	Checks if the file is UNICODE file
IsUnicodeFileW (see page 84)	Checks if the file is UNICODE file
IsValidFileNameA (see page 84)	Checks if provided string is a valid file name
IsValidFileNameW (see page 84)	Checks if provided string is a valid file name
IsValidPathNameA (see page 85)	Checks if provided string is a valid file path
IsValidPathNameW (see page 85)	Checks if provided string is a valid file path
ReadBufferFromFileA (see page 95)	Reads the content of the file to the memory buffer
ReadBufferFromFileW (see page 95)	Reads the content of the file to the memory buffer
ShellCopyFileA (see page 124)	Copies file to the new location
ShellCopyFileW (see page 125)	Copies file to the new location
StripFileNameA (see page 126)	Replaces environment variable names with values
StripFileNameW (see page 127)	Replaces environment variable names with values
WriteBufferToFileA (see page 130)	Writes the memory buffer to file
WriteBufferToFileW (see page 130)	Writes the memory buffer to file

## Macros

Name	Description
IID_PPV_ARG (see page 173)	<p>IID_PPV_ARG(IType, ppType) IType is the type of pType ppType is the variable of type IType that will be filled</p> <p>RESULTS in: IID_IType, ppvType will create a compiler error if wrong level of indirection is used.</p> <p>macro for QueryInterface and related functions that require a IID and a (void **) this will insure that the cast is safe and appropriate on C</p>

## 1.4.5 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 24)	This is function CloneFont.
⇒	CopyNativeBitmap (↗ see page 26)	This is function CopyNativeBitmap.
⇒	CreateNativeBitmap (↗ see page 26)	This is function CreateNativeBitmap.
⇒	CreateWindowsFont (↗ see page 27)	This is function CreateWindowsFont.
⇒	DestroyFont (↗ see page 32)	This is function DestroyFont.
⇒	DpiY (↗ see page 33)	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 35)	This is function DrawTextDirectEx.
⇒	DrawTextGlow (↗ see page 35)	This is function DrawTextGlow.
⇒	DrawTextLine (↗ see page 36)	This is function DrawTextLine.
⇒	DrawTextOutline (↗ see page 36)	This is function DrawTextOutline.
⇒	DrawTextRect (↗ see page 36)	This is function DrawTextRect.
⇒	EmToPixels (↗ see page 37)	This is function EmToPixels.
⇒	GetTextLineSize (↗ see page 73)	This is function GetTextLineSize.
⇒	GetTextSize (↗ see page 73)	This is function GetTextSize.
⇒	GetTextSizeEx (↗ see page 73)	This is function GetTextSizeEx.
⇒	LoadBitmapJPG (↗ see page 88)	This is function LoadBitmapJPG.
⇒	LoadBitmapPNG (↗ see page 88)	This is function LoadBitmapPNG.
⇒	LoadPNGResource (↗ see page 90)	This is function LoadPNGResource.
⇒	MakeCompatibleBitmap (↗ see page 91)	This is function MakeCompatibleBitmap.
⇒	PointsToPixels (↗ see page 92)	This is function PointsToPixels.
⇒	StretchNativeBitmap (↗ see page 126)	This is function StretchNativeBitmap.

### Macros

Name	Description
FONT_BOLD (↗ see page 172)	This is macro FONT_BOLD.
FONT_ITALIC (↗ see page 172)	This is macro FONT_ITALIC.
FONT_NORMAL (↗ see page 172)	This is macro FONT_NORMAL.
FONT_STRIKEOUT (↗ see page 173)	This is macro FONT_STRIKEOUT.
FONT_UNDERLINE (↗ see page 173)	This is macro FONT_UNDERLINE.
WIDTHBYTES (↗ see page 177)	This is macro WIDTHBYTES.

## 1.4.6 NationalityConverter.h

Nationality to ISO code converter

### Functions

	Name	Description
◆	GetISOCodeA (see page 66)	Returns the country ISO code based on the nationality string
◆	GetISOCodeW (see page 66)	Returns the country ISO code based on the nationality string

## 1.4.7 quicol.h

QR Code generator

### Functions

	Name	Description
◆	DestroyImageBuffer (see page 32)	Destroys the memory buffer
◆	GenerateBMPA (see page 53)	Generates Windows Bitmap file with QR Code image
◆	GenerateBMPW (see page 53)	Generates Windows Bitmap file with QR Code image
◆	GeneratePNGA (see page 56)	Generates PNG file with QR Code image
◆	GeneratePNGW (see page 56)	Generates PNG file with QR Code image
◆	GetHBitmapA (see page 65)	Generates Windows Bitmap in memory with QR Code image
◆	GetHBitmapW (see page 65)	Generates Windows Bitmap in memory with QR Code image
◆	GetPNGA (see page 67)	Writes PNG image to the memory buffer.
◆	GetPNGW (see page 68)	Writes PNG image to the memory buffer.

## 1.4.8 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 10)	Established communication between the card and the reader
◆	AddFileToContainer (see page 10)	Add existing file to the container
◆	CardSignCadesT (see page 17)	Sign data with eID card according to CADES-T standard
◆	CardSignCadesTEx (see page 17)	Sign data with eID card according to CADES-T standard
◆	CardSignCMS (see page 18)	Sign data with eID card according to CMS standard
◆	CardSignCMSEx (see page 18)	Sign data with eID card according to CMS standard
◆	CertSignCadesT (see page 19)	Sign data with the certificate file according to CADES-T standard
◆	CertSignCadesTData (see page 19)	Sign data with the certificate according to CADES-T standard
◆	CertSignCadesTEx (see page 20)	Sign data with the certificate file according to CADES-T standard
◆	CertSignCMS (see page 21)	Sign data with the certificate file according to CMS standard

✦	CertSignCMSData (see page 21)	Sign data with the certificate according to CMS standard
✦	CertSignCMSEx (see page 22)	Sign data with the certificate file according to CMS 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 25)	Pick certificate to sign ASIC container
✦	CreateCardBuffer (see page 26)	Creates XML buffer
✦	DeactivateCard (see page 28)	Terminates a connection between a smart card and a reader
✦	DeactivateCardEx (see page 28)	Terminates a connection between a smart card and a reader
✦	DeleteCardBuffer (see page 31)	Deletes XML buffer
✦	DisplayCertificate (see page 33)	Displays the dialog window with certificate information
✦	EncodeCertificate (see page 37)	Performs Base64 encoding of the certificate
✦	EncodePhoto (see page 37)	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 52)	Generate authentication signature
✦	GenerateNonRepudiationSignatureA (see page 54)	Generate non repudiation signature
✦	GenerateNonRepudiationSignatureExA (see page 54)	Generate non repudiation signature
✦	GenerateNonRepudiationSignatureExW (see page 55)	Generate non repudiation signature
✦	GenerateNonRepudiationSignatureW (see page 55)	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 57)	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 59)	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 59)	Gets XML or CSV information from the memory buffer
✦	GetCardSerialNumber (see page 60)	Get the serial number of EID card
✦	GetCardVersion (see page 60)	Get the applet version number for card in the reader with specified number
✦	GetEncodedCertificateSize (see page 61)	Returns the size of the Base64 encoded certificate
✦	GetEncodedPhotoSize (see page 61)	Calculates buffer size for Base64 encoded photo
✦	GetReaderIndexA (see page 68)	Returns the zero-based reader index with specified name
✦	GetReaderIndexW (see page 68)	Returns the zero-based reader index with specified name
✦	GetReaderNameA (see page 69)	Returns the name of the card reader

◆	GetReaderNameLenA (see page 69)	Returns the length of the reader name
◆	GetReaderNameLenW (see page 70)	Returns the length of the reader name
◆	GetReaderNameW (see page 70)	Returns the name of the card reader
◆	GetReadersCount (see page 70)	Get number of card readers connected to PC
◆	GetSelectedReaderIndex (see page 71)	Returns the index of the active smart card reader
◆	GetSupportSIS (see page 73)	Checks if the SIS cards are supported by the engine
◆	IgnoreHardwareEvents (see page 74)	Ignore USB reader insert / remove events
◆	IgnoreServiceEvents (see page 74)	Ignore smartcard service stop events when reporting readers list change
◆	InitializeContainer (see page 75)	Initializes ASIC container
◆	IsCardActivated (see page 76)	Checks the connection between a smart card and a reader
◆	IsCardActivatedEx (see page 76)	Checks the connection between a smart card and a reader
◆	IsCardPresent (see page 76)	Checks if the card is present in the card reader
◆	IsCardPresentEx (see page 77)	Checks if the card is present in the card reader
◆	IsCardStillInserted (see page 77)	Checks if the card is still inserted in the card reader
◆	IsCardStillInsertedEx (see page 77)	Checks if the card is still inserted in the card reader
◆	IsEIDCard (see page 79)	Check if Belgian EID card is inserted into card reader
◆	IsEIDCardEx (see page 79)	Check if Belgian EID card is inserted into card reader
◆	IsEngineActive (see page 79)	Checks if the Swelio Engine is activated
◆	IsFemaleA (see page 80)	Checks if the card owner is female
◆	IsFemaleW (see page 80)	Checks if the card owner is female
◆	IsMaleA (see page 80)	Checks if the card owner is male
◆	IsMaleW (see page 81)	Checks if the card owner is male
◆	IsSISCard (see page 82)	Check if Belgian SIS card is inserted into card reader
◆	IsSISCardEx (see page 83)	Check if Belgian SIS card is inserted into card reader
◆	LoadCertificateA (see page 88)	Reads the certificate from a file
◆	LoadCertificateW (see page 89)	Reads the certificate from a file
◆	LoadIdentityA (see page 89)	Reads the raw identity information from a file
◆	LoadIdentityW (see page 89)	Reads the raw identity information from a file
◆	LoadPhotoA (see page 90)	Loads photo from a file
◆	LoadPhotoW (see page 90)	Loads photo from a file
◆	ReadAddressA (see page 93)	Read address information from Belgian eID card
◆	ReadAddressExA (see page 93)	Read address information from Belgian eID card
◆	ReadAddressExW (see page 94)	Read address information from Belgian eID card
◆	ReadAddressW (see page 94)	Read address information from Belgian eID card
◆	ReadAuthenticationCertificate (see page 94)	Read Authentication Certificate to memory
◆	ReadAuthenticationCertificateEx (see page 95)	Read Authentication Certificate to memory
◆	ReadCaCertificate (see page 96)	Read Ca Certificate to memory
◆	ReadCaCertificateEx (see page 96)	Read Ca Certificate to memory
◆	ReadIdentityA (see page 97)	Read identity information from Belgian eID card
◆	ReadIdentityExA (see page 97)	Read identity information from Belgian eID card
◆	ReadIdentityExW (see page 97)	Read identity information from Belgian eID card
◆	ReadIdentityW (see page 98)	Read identity information from Belgian eID card
◆	ReadNonRepudiationCertificate (see page 98)	Read Non Repudiation Certificate to memory
◆	ReadNonRepudiationCertificateEx (see page 99)	Read Non Repudiation Certificate to memory



ReadPhoto (see page 99)	Reads a photo from a card
ReadPhotoAsBitmap (see page 99)	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 100)	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 100)	Reads a photo from a card
ReadRootCaCertificate (see page 101)	Read Root Ca Certificate to memory
ReadRootCaCertificateEx (see page 101)	Read Root Ca Certificate to memory
ReadRrnCertificate (see page 101)	Read Rrn Certificate to memory
ReadRrnCertificateEx (see page 102)	Read Rrn Certificate to memory
ReadSISCardA (see page 102)	Read Belgian SIS card.
ReadSISCardExA (see page 103)	Read Belgian SIS card.
ReadSISCardExW (see page 103)	Read Belgian SIS card.
ReadSISCardW (see page 103)	Read Belgian SIS card.
ReloadReadersList (see page 104)	Reloads the list of the available card readers
RemoveCallback (see page 104)	Remove callback procedure for card events
SaveAuthenticationCertificateA (see page 106)	Save Authentication Certificate to a file
SaveAuthenticationCertificateExW (see page 106)	Save Authentication Certificate to a file
SaveAuthenticationCertificateW (see page 107)	Save Authentication Certificate to a file
SaveCaCertificateA (see page 107)	Save Ca Certificate to a file
SaveCaCertificateExW (see page 107)	Save Ca Certificate to a file
SaveCaCertificateW (see page 108)	Save Ca Certificate to a file
SaveCardToXMLStreamExA (see page 108)	Read eID card and save the information to XML buffer
SaveCardToXMLStreamExW (see page 108)	Read eID card and save the information to XML buffer
SaveCardToXmlA (see page 109)	Read eID card and save the information to XML file
SaveCardToXmlExA (see page 109)	Read eID card and save the information to XML file
SaveCardToXmlExW (see page 110)	Read eID card and save the information to XML file
SaveCardToXmlW (see page 110)	Read eID card and save the information to XML file
SaveContainer (see page 111)	Save container to the file
SaveIdentityA (see page 111)	Saves identity information to a file
SaveIdentityW (see page 111)	Saves identity information to a file
SaveNonRepudiationCertificateA (see page 112)	Save Non Repudiation Certificate to a file
SaveNonRepudiationCertificateExW (see page 112)	Save Non Repudiation Certificate to a file
SaveNonRepudiationCertificateW (see page 112)	Save Non Repudiation Certificate to a file
SavePersonCsvToStreamA (see page 113)	Read eID card and save the identity information to CSV memory buffer
SavePersonCsvToStreamW (see page 113)	Read eID card and save the identity information to CSV memory buffer

◆	SavePersonToCsvA (see page 114)	Read eID card and save the identity information and address to CSV file
◆	SavePersonToCsvExA (see page 114)	Read eID card and save the identity information and address to CSV file
◆	SavePersonToCsvExW (see page 114)	Read eID card and save the identity information and address to CSV file
◆	SavePersonToCsvW (see page 115)	Read eID card and save the identity information and address to CSV file
◆	SavePhotoA (see page 115)	Save photo to a file
◆	SavePhotoAsBitmapA (see page 116)	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 116)	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 116)	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 117)	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 117)	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 117)	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 118)	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 118)	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 119)	Saves photo to a file
◆	SaveRootCaCertificateA (see page 119)	Save Root Ca Certificate to a file
◆	SaveRootCaCertificateExW (see page 119)	Save Root Ca Certificate to a file
◆	SaveRootCaCertificateW (see page 120)	Save Root Ca Certificate to a file
◆	SaveRrnCertificateA (see page 120)	Save RRN Certificate to a file
◆	SaveRrnCertificateExW (see page 120)	Save RRN Certificate to a file
◆	SaveRrnCertificateW (see page 121)	Save RRN Certificate to a file
◆	SelectReader (see page 121)	When more than 1 reader connected, select the reader with specified number
◆	SelectReaderByNameA (see page 121)	Select active smart card reader by providing the reader name



✦	SelectReaderByNameW (see page 122)	Select active smart card reader by providing the reader name
✦	SendAPDU (see page 122)	This is function SendAPDU.
✦	SetCallback (see page 122)	Activates callback procedure for card status change event
✦	SetMWCompatibility (see page 123)	Set the compatibility mode with the old version of the official EID MiddleWare
✦	SetSupportSIS (see page 124)	Activates or deactivates SIS card support by engine
✦	StartEngine (see page 126)	Activates the Swelio Engine.
✦	StopEngine (see page 126)	Deactivates the Swelio Engine
✦	VerifyPinA (see page 128)	Verify PIN code
✦	VerifyPinExA (see page 128)	Verify PIN code
✦	VerifyPinExW (see page 129)	Verify PIN code
✦	VerifyPinW (see page 129)	Verify PIN code
✦	VerifySignature (see page 130)	Verifies the signature from the specified hash value.

**Macros**

Name	Description
SaveCardToToXMLStreamEx (see page 174)	This is macro SaveCardToToXMLStreamEx.
SavePersonCsvToStream (see page 174)	This is macro SavePersonCsvToStream.

## 1.4.9 System.h

Windows related routines

**Functions**

	Name	Description
✦	AddRemoveMessageFilter (see page 11)	Adds or removes a message from the User Interface Privilege Isolation (UIPI) message filter.
✦	AllocateDefaultHWNDAs (see page 12)	This function creates the invisible tool window
✦	AllocateDefaultHWNDA (see page 12)	This function creates the invisible tool window
✦	AllocateHWNDAs (see page 13)	This function creates the invisible tool window using the provided window procedure
✦	AllocateHWNDA (see page 13)	This function creates the invisible tool window using the provided window procedure
✦	AllocateLayeredWindowA (see page 13)	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 14)	This function creates the standard window using the provided window class name
✦	BringWindowToFront (see page 15)	This function brings the specified window to the top of the z-order.
✦	ClearUnusedMemory (see page 24)	Clears unused memory and minimized the application memory usage
✦	DeallocateHWNDAs (see page 29)	This function destroys the specified window.
✦	DeallocateHWNDA (see page 29)	This function destroys the specified window.

◆	DrawLayeredWindow (see page 34)	Repaints the surface of the layered window
◆	EmptyRecycleBin (see page 36)	Empties the recycle bin
◆	fpreset (see page 50)	This is function fpreset.
◆	GetStartupA (see page 72)	Checks if the application is registered to run when Windows starts
◆	GetStartupW (see page 72)	Checks if the application is registered to run when Windows starts
◆	HibernateWindows (see page 74)	Hibernates Windows
◆	LayeredWndProcA (see page 87)	The default window procedure for the layered window
◆	LayeredWndProcW (see page 87)	The default window procedure for the layered window
◆	MakeSoundFromFileA (see page 91)	Plays the wave sound from the file
◆	MakeSoundFromFileW (see page 91)	Plays the wave sound from the file
◆	MakeSoundFromResourceA (see page 92)	Plays the wave sound from the resource
◆	MakeSoundFromResourceW (see page 92)	Plays the wave sound from the resource
◆	RemoveStartupA (see page 105)	Removes the application from the list of the automatically started applications
◆	RemoveStartupW (see page 105)	Removes the application from the list of the automatically started applications
◆	RestoreWindowSubclassA (see page 105)	Restores window standard procedure
◆	RestoreWindowSubclassW (see page 106)	Restores window standard procedure
◆	SetStartupA (see page 123)	Register application to run when Windows starts
◆	SetStartupW (see page 123)	Register application to run when Windows starts
◆	ShutdownWindows (see page 125)	Logs off the interactive user, shuts down the system.
◆	SuspendWindows (see page 127)	Suspends Windows
◆	TurnMonitorOff (see page 127)	Turns the monitor off
◆	TurnMonitorOn (see page 128)	Turns the monitor on
◆	UpdateWindowPosition (see page 128)	Updated the window position

## 1.4.10 SystemInfo.h

Routines for querying information about operating system

### Functions

	Name	Description
◆	CurrentIPAddressA (see page 27)	Returns the IP address
◆	CurrentIPAddressW (see page 28)	Returns the IP address
◆	IsCitrixSession (see page 78)	Checks if application is running in Citrix session
◆	IsConnectedToInternet (see page 78)	Checks if PC is connected to Internet
◆	IsMediaCenter (see page 81)	Checks if the Media Center version of Windows is installed
◆	IsMetroActive (see page 81)	Checks if metro interface is active
◆	IsMultiTouchReady (see page 82)	Checks if the system is multi touch ready
◆	IsNativeWin64 (see page 82)	Checks if the application is native 64 bit executable

◆	IsRemoteSession ( see page 82)	Checks if application is running in RDP session
◆	IsTabletPC ( see page 83)	Checks if the application is running on the Tablet PC
◆	IsWindows10 ( see page 86)	Checks if PC is running Windows 10 or better
◆	IsWindows7 ( see page 86)	Checks if PC is running Windows 7 or better
◆	IsWindows8 ( see page 86)	Checks if PC is Running Windows 8 or better
◆	IsWindowsVista ( see page 86)	Checks if PC is running Windows Vista or better
◆	IsWindowsXP ( see page 87)	Checks if PC is running Windows XP
◆	IsWindowsXPSP2 ( see page 87)	Checks if PC is running Windows XP with Service Pack 2 installed
◆	IsWow64 ( see page 87)	Checks if the 32 bit application runs on 64 bit Windows
◆	PortAvailable ( see page 93)	Checks if the port with specified number is available
◆	RecycleBinEmpty ( see page 104)	Returns TRUE if Windows Recycle Bin is empty

## Index

### A

ActivateCard 10  
ActivateCard function 10  
ActivateCardEx 10  
ActivateCardEx function 10  
AddFileToContainer 10  
AddFileToContainer function 10  
AddRemoveMessageFilter 11  
AddRemoveMessageFilter function 11  
AllocateBuffer 11  
AllocateBuffer function 11  
AllocateDefaultHWND 12  
AllocateDefaultHWND function 12  
AllocateDefaultHWNDW 12  
AllocateDefaultHWNDW function 12  
AllocateHWND 13  
AllocateHWND function 13  
AllocateHWNDW 13  
AllocateHWNDW function 13  
AllocateLayeredWindowA 13  
AllocateLayeredWindowA function 13  
AllocateLayeredWindowW 14  
AllocateLayeredWindowW function 14  
AllocateWindowClassA 14  
AllocateWindowClassA function 14  
AllocateWindowClassW 14  
AllocateWindowClassW function 14  
AlphaBlendBitmap 15  
AlphaBlendBitmap function 15  
AlphaBlendNative 15  
AlphaBlendNative function 15

### B

BringWindowToFront 15  
BringWindowToFront function 15

### C

CardDecryptFileA 15  
CardDecryptFileA function 15

CardDecryptFileW 16  
CardDecryptFileW function 16  
CardEncryptFileA 16  
CardEncryptFileA function 16  
CardEncryptFileW 16  
CardEncryptFileW function 16  
CardEvents.h 177  
CardEventType 142  
CardEventType enumeration 142  
CardSignCadesT 17  
CardSignCadesT function 17  
CardSignCadesTEx 17  
CardSignCadesTEx function 17  
CardSignCMS 18  
CardSignCMS function 18  
CardSignCMSEx 18  
CardSignCMSEx function 18  
CardStructures.h 178  
CertSignCadesT 19  
CertSignCadesT function 19  
CertSignCadesTData 19  
CertSignCadesTData function 19  
CertSignCadesTEx 20  
CertSignCadesTEx function 20  
CertSignCMS 21  
CertSignCMS function 21  
CertSignCMSData 21  
CertSignCMSData function 21  
CertSignCMSEx 22  
CertSignCMSEx function 22  
CheckMD5 22  
CheckMD5 function 22  
CheckSHA1 23  
CheckSHA1 function 23  
CheckSHA256 23  
CheckSHA256 function 23  
ClearFileAttributesA 23  
ClearFileAttributesA function 23  
ClearFileAttributesW 24  
ClearFileAttributesW function 24  
ClearUnusedMemory 24  
ClearUnusedMemory function 24

CloneFont 24  
 CloneFont function 24  
 ContainerCertificate 25  
 ContainerCertificate function 25  
 ContainerEidCertificate 25  
 ContainerEidCertificate function 25  
 ContainerPickCertificate 25  
 ContainerPickCertificate function 25  
 CopyNativeBitmap 26  
 CopyNativeBitmap function 26  
 CreateCardBuffer 26  
 CreateCardBuffer function 26  
 CreateNativeBitmap 26  
 CreateNativeBitmap function 26  
 CreateUnicodeFileA 27  
 CreateUnicodeFileA function 27  
 CreateUnicodeFileW 27  
 CreateUnicodeFileW function 27  
 CreateWindowsFont 27  
 CreateWindowsFont function 27  
 CurrentIPAddressA 27  
 CurrentIPAddressA function 27  
 CurrentIPAddressW 28  
 CurrentIPAddressW function 28

## D

DeactivateCard 28  
 DeactivateCard function 28  
 DeactivateCardEx 28  
 DeactivateCardEx function 28  
 DeallocateBuffer 29  
 DeallocateBuffer function 29  
 DeallocateHWNDAs 29  
 DeallocateHWNDAs function 29  
 DeallocateHWNDA 29  
 DeallocateHWNDA function 29  
 DeallocateHWNDAW 29  
 DeallocateHWNDAW function 29  
 DecryptFileAESA 30  
 DecryptFileAESA function 30  
 DecryptFileAESW 30  
 DecryptFileAESW function 30  
 DeleteCardBuffer 31  
 DeleteCardBuffer function 31

DeleteToRecycleBinA 31  
 DeleteToRecycleBinA function 31  
 DeleteToRecycleBinW 31  
 DeleteToRecycleBinW function 31  
 DestroyFont 32  
 DestroyFont function 32  
 DestroyImageBuffer 32  
 DestroyImageBuffer function 32  
 DirectoryExistsA 32  
 DirectoryExistsA function 32  
 DirectoryExistsW 33  
 DirectoryExistsW function 33  
 DisplayCertificate 33  
 DisplayCertificate function 33  
 DpiY 33  
 DpiY function 33  
 DrawAlphaText 34  
 DrawAlphaText function 34  
 DrawAlphaTextRect 34  
 DrawAlphaTextRect function 34  
 DrawLayeredWindow 34  
 DrawLayeredWindow function 34  
 DrawNativeBitmap 35  
 DrawNativeBitmap function 35  
 DrawTextDirect 35  
 DrawTextDirect function 35  
 DrawTextDirectEx 35  
 DrawTextDirectEx function 35  
 DrawTextGlow 35  
 DrawTextGlow function 35  
 DrawTextLine 36  
 DrawTextLine function 36  
 DrawTextOutline 36  
 DrawTextOutline function 36  
 DrawTextRect 36  
 DrawTextRect function 36

## E

EID\_MAX\_BIRTHDATE\_LEN 162  
 EID\_MAX\_BIRTHDATE\_LEN macro 162  
 EID\_MAX\_BIRTHPLACE\_LEN 162  
 EID\_MAX\_BIRTHPLACE\_LEN macro 162

EID_MAX_BREXITMENTION1_LEN 163	EID_MAX_SEX_LEN 167
EID_MAX_BREXITMENTION1_LEN macro 163	EID_MAX_SEX_LEN macro 167
EID_MAX_BREXITMENTION2_LEN 163	EID_MAX_SPECIAL_STATUS_LEN 168
EID_MAX_BREXITMENTION2_LEN macro 163	EID_MAX_SPECIAL_STATUS_LEN macro 168
EID_MAX_CARD_NUMBER_LEN 163	EID_MAX_SPECIALORGANIZATION_LEN 168
EID_MAX_CARD_NUMBER_LEN macro 163	EID_MAX_SPECIALORGANIZATION_LEN macro 168
EID_MAX_CERT_LEN 163	EID_MAX_STREET_LEN 168
EID_MAX_CERT_LEN macro 163	EID_MAX_STREET_LEN macro 168
EID_MAX_CHIP_NUMBER_LEN 163	EID_MAX_VAT1_LEN 168
EID_MAX_CHIP_NUMBER_LEN macro 163	EID_MAX_VAT1_LEN macro 168
EID_MAX_DATE_BEGIN_LEN 164	EID_MAX_VAT2_LEN 168
EID_MAX_DATE_BEGIN_LEN macro 164	EID_MAX_VAT2_LEN macro 168
EID_MAX_DATE_END_LEN 164	EID_MAX_WORKPERMITTYPE_LEN 169
EID_MAX_DATE_END_LEN macro 164	EID_MAX_WORKPERMITTYPE_LEN macro 169
EID_MAX_DATEANDCOUNTRYOFPROTECTION_LEN 164	EID_MAX_ZIP_LEN 169
EID_MAX_DATEANDCOUNTRYOFPROTECTION_LEN macro 164	EID_MAX_ZIP_LEN macro 169
EID_MAX_DELIVERY_MUNICIPALITY_LEN 164	EidAddressA 142
EID_MAX_DELIVERY_MUNICIPALITY_LEN macro 164	EidAddressA structure 142
EID_MAX_DOCUMENT_TYPE_LEN 165	EidAddressW 142
EID_MAX_DOCUMENT_TYPE_LEN macro 165	EidAddressW structure 142
EID_MAX_DUPLICATE_LEN 165	EidCertificate 143
EID_MAX_DUPLICATE_LEN macro 165	EidCertificate structure 143
EID_MAX_FIRST_NAME1_LEN 165	EidIdentityA 143
EID_MAX_FIRST_NAME1_LEN macro 165	EidIdentityA structure 143
EID_MAX_FIRST_NAME2_LEN 165	EidIdentityExA 145
EID_MAX_FIRST_NAME2_LEN macro 165	EidIdentityExA structure 145
EID_MAX_MEMBEROFFAMILY_LEN 166	EidIdentityExW 146
EID_MAX_MEMBEROFFAMILY_LEN macro 166	EidIdentityExW structure 146
EID_MAX_MUNICIPALITY_LEN 166	EidIdentityW 148
EID_MAX_MUNICIPALITY_LEN macro 166	EidIdentityW structure 148
EID_MAX_NAME_LEN 166	EidPicture 149
EID_MAX_NAME_LEN macro 166	EidPicture structure 149
EID_MAX_NATIONAL_NUMBER_LEN 166	EmptyRecycleBin 36
EID_MAX_NATIONAL_NUMBER_LEN macro 166	EmptyRecycleBin function 36
EID_MAX_NATIONALITY_LEN 166	EmToPixels 37
EID_MAX_NATIONALITY_LEN macro 166	EmToPixels function 37
EID_MAX_NOBLE_CONDITION_LEN 167	EncodeCertificate 37
EID_MAX_NOBLE_CONDITION_LEN macro 167	EncodeCertificate function 37
EID_MAX_PICTURE_LEN 167	EncodePhoto 37
EID_MAX_PICTURE_LEN macro 167	EncodePhoto function 37
EID_MAX_REGIONALFILENUMBER_LEN 167	EncryptFileAESA 38
EID_MAX_REGIONALFILENUMBER_LEN macro 167	EncryptFileAESA function 38

EncryptFileAESW 38  
 EncryptFileAESW function 38  
 Encryption.h 180  
 ERR\_CARD\_ERROR 169  
 ERR\_CARD\_ERROR macro 169  
 ERR\_CERTIFICATE\_ATTR 169  
 ERR\_CERTIFICATE\_ATTR macro 169  
 ERR\_MESSAGE\_DECODE 170  
 ERR\_MESSAGE\_DECODE macro 170  
 ERR\_MESSAGE\_ENCODE 170  
 ERR\_MESSAGE\_ENCODE macro 170  
 ERR\_MESSAGE\_PARAM 170  
 ERR\_MESSAGE\_PARAM macro 170  
 ERR\_MESSAGE\_UPDATE 170  
 ERR\_MESSAGE\_UPDATE macro 170  
 ERR\_NO\_CERTIFICATE 170  
 ERR\_NO\_CERTIFICATE macro 170  
 ERR\_NO\_DATA 171  
 ERR\_NO\_DATA macro 171  
 ERR\_NO\_PRIVATE\_KEY 171  
 ERR\_NO\_PRIVATE\_KEY macro 171  
 ERR\_OPEN\_CERTIFICATE 171  
 ERR\_OPEN\_CERTIFICATE macro 171  
 ERR\_SIGN\_ERROR 171  
 ERR\_SIGN\_ERROR macro 171  
 ERR\_TIMESTAMP 172  
 ERR\_TIMESTAMP macro 172  
 ERROR\_MAX\_DESCRIPTION 172  
 ERROR\_MAX\_DESCRIPTION macro 172  
 ErrorInformation 149  
 ErrorInformation structure 149  
 ewtCardInsert enumeration member 132  
 ewtCardRemove enumeration member 132  
 ewtReadersChange enumeration member 132  
 ewtUnknownEvent enumeration member 132

## F

FileCloseA 39  
 FileCloseA function 39  
 FileCloseW 39  
 FileCloseW function 39  
 FileCopyA 39  
 FileCopyA function 39  
 FileCopyW 40  
 FileCopyW function 40  
 FileCreateRewriteA 40  
 FileCreateRewriteA function 40  
 FileCreateRewriteW 40  
 FileCreateRewriteW function 40  
 FileDeleteA 41  
 FileDeleteA function 41  
 FileDeleteW 41  
 FileDeleteW function 41  
 FileExistsA 42  
 FileExistsA function 42  
 FileExistsW 42  
 FileExistsW function 42  
 FileExtensionIsA 42  
 FileExtensionIsA function 42  
 FileExtensionIsW 43  
 FileExtensionIsW function 43  
 FileGetSizeA 43  
 FileGetSizeA function 43  
 FileGetSizeW 43  
 FileGetSizeW function 43  
 FileIsExeA 44  
 FileIsExeA function 44  
 FileIsExeW 44  
 FileIsExeW function 44  
 FileIsIconA 45  
 FileIsIconA function 45  
 FileIsIconW 45  
 FileIsIconW function 45  
 FileIsImageA 45  
 FileIsImageA function 45  
 FileIsImageW 46  
 FileIsImageW function 46  
 FileIsLink 46  
 FileIsLink function 46  
 FileOperations.h 181  
 FileOrFolderExistsA 46  
 FileOrFolderExistsA function 46  
 FileOrFolderExistsW 47  
 FileOrFolderExistsW function 47

FileRenameA 47	GenerateAuthenticationSignatureExW 52
FileRenameA function 47	GenerateAuthenticationSignatureExW function 52
FileRenameW 47	GenerateAuthenticationSignatureW 52
FileRenameW function 47	GenerateAuthenticationSignatureW function 52
Files 177	GenerateBMPA 53
FileWriteA 48	GenerateBMPA function 53
FileWriteA function 48	GenerateBMPW 53
FileWriteCharA 48	GenerateBMPW function 53
FileWriteCharA function 48	GenerateNonRepudiationSignatureA 54
FileWriteCharW 48	GenerateNonRepudiationSignatureA function 54
FileWriteCharW function 48	GenerateNonRepudiationSignatureExA 54
FileWriteNewLineA 49	GenerateNonRepudiationSignatureExA function 54
FileWriteNewLineA function 49	GenerateNonRepudiationSignatureExW 55
FileWriteNewLineW 49	GenerateNonRepudiationSignatureExW function 55
FileWriteNewLineW function 49	GenerateNonRepudiationSignatureW 55
FileWriteW 49	GenerateNonRepudiationSignatureW function 55
FileWriteW function 49	GeneratePNGA 56
FONT_BOLD 172	GeneratePNGA function 56
FONT_BOLD macro 172	GeneratePNGW 56
FONT_ITALIC 172	GeneratePNGW function 56
FONT_ITALIC macro 172	GenerateQRCodeA 57
FONT_NORMAL 172	GenerateQRCodeA function 57
FONT_NORMAL macro 172	GenerateQRCodeExA 57
FONT_STRIKEOUT 173	GenerateQRCodeExA function 57
FONT_STRIKEOUT macro 173	GenerateQRCodeExW 57
FONT_UNDERLINE 173	GenerateQRCodeExW function 57
FONT_UNDERLINE macro 173	GenerateQRCodeW 58
fpreset 50	GenerateQRCodeW function 58
fpreset function 50	GetAllFiles 58
FreeContainer 50	GetAllFiles function 58
FreeContainer function 50	GetCardBufferA 59
FullPathA 50	GetCardBufferA function 59
FullPathA function 50	GetCardBufferSize 59
FullPathW 51	GetCardBufferSize function 59
FullPathW function 51	GetCardBufferW 59
Functions 1	GetCardBufferW function 59
<b>G</b>	GetCardSerialNumber 60
GenerateAuthenticationSignatureA 51	GetCardSerialNumber function 60
GenerateAuthenticationSignatureA function 51	GetCardVersion 60
GenerateAuthenticationSignatureExA 51	GetCardVersion function 60
GenerateAuthenticationSignatureExA function 51	GetEncodedCertificateSize 61
	GetEncodedCertificateSize function 61



GetEncodedPhotoSize 61  
 GetEncodedPhotoSize function 61  
 GetFileMD5A 62  
 GetFileMD5A function 62  
 GetFileMD5W 62  
 GetFileMD5W function 62  
 GetFilesCountA 62  
 GetFilesCountA function 62  
 GetFilesCountW 63  
 GetFilesCountW function 63  
 GetFileSHA1A 63  
 GetFileSHA1A function 63  
 GetFileSHA1W 64  
 GetFileSHA1W function 64  
 GetFileSHA256 173  
 GetFileSHA256 macro 173  
 GetFileSHA256A 64  
 GetFileSHA256A function 64  
 GetFileSHA256W 64  
 GetFileSHA256W function 64  
 GetHBitmapA 65  
 GetHBitmapA function 65  
 GetHBitmapW 65  
 GetHBitmapW function 65  
 GetISOCODEA 66  
 GetISOCODEA function 66  
 GetISOCODEW 66  
 GetISOCODEW function 66  
 GetMD5 67  
 GetMD5 function 67  
 GetPNGA 67  
 GetPNGA function 67  
 GetPNGW 68  
 GetPNGW function 68  
 GetReaderIndexA 68  
 GetReaderIndexA function 68  
 GetReaderIndexW 68  
 GetReaderIndexW function 68  
 GetReaderNameA 69  
 GetReaderNameA function 69  
 GetReaderNameLenA 69  
 GetReaderNameLenA function 69

GetReaderNameLenW 70  
 GetReaderNameLenW function 70  
 GetReaderNameW 70  
 GetReaderNameW function 70  
 GetReadersCount 70  
 GetReadersCount function 70  
 GetSelectedReaderIndex 71  
 GetSelectedReaderIndex function 71  
 GetSHA1 71  
 GetSHA1 function 71  
 GetSHA256 71  
 GetSHA256 function 71  
 GetStartupA 72  
 GetStartupA function 72  
 GetStartupW 72  
 GetStartupW function 72  
 GetSupportSIS 73  
 GetSupportSIS function 73  
 GetTextLineSize 73  
 GetTextLineSize function 73  
 GetTextSize 73  
 GetTextSize function 73  
 GetTextSizeEx 73  
 GetTextSizeEx function 73  
 Graphics.h 183

## H

HibernateWindows 74  
 HibernateWindows function 74

## I

IgnoreHardwareEvents 74  
 IgnoreHardwareEvents function 74  
 IgnoreServiceEvents 74  
 IgnoreServiceEvents function 74  
 IID\_PPV\_ARG 173  
 IID\_PPV\_ARG macro 173  
 InitializeContainer 75  
 InitializeContainer function 75  
 IsAnimatedGIFA 75  
 IsAnimatedGIFA function 75  
 IsAnimatedGIFW 75

IsAnimatedGIFW function 75	IsNativeWin64 function 82
IsCardActivated 76	IsRemoteSession 82
IsCardActivated function 76	IsRemoteSession function 82
IsCardActivatedEx 76	IsSISCard 82
IsCardActivatedEx function 76	IsSISCard function 82
IsCardPresent 76	IsSISCardEx 83
IsCardPresent function 76	IsSISCardEx function 83
IsCardPresentEx 77	IsTabletPC 83
IsCardPresentEx function 77	IsTabletPC function 83
IsCardStillInserted 77	IsUnicodeFileA 83
IsCardStillInserted function 77	IsUnicodeFileA function 83
IsCardStillInsertedEx 77	IsUnicodeFileW 84
IsCardStillInsertedEx function 77	IsUnicodeFileW function 84
IsCitrixSession 78	IsValidFileNameA 84
IsCitrixSession function 78	IsValidFileNameA function 84
IsConnectedToInternet 78	IsValidFileNameW 84
IsConnectedToInternet function 78	IsValidFileNameW function 84
IsDirectoryA 78	IsValidPathNameA 85
IsDirectoryA function 78	IsValidPathNameA function 85
IsDirectoryW 78	IsValidPathNameW 85
IsDirectoryW function 78	IsValidPathNameW function 85
IsEIDCard 79	IsWindows10 86
IsEIDCard function 79	IsWindows10 function 86
IsEIDCardEx 79	IsWindows7 86
IsEIDCardEx function 79	IsWindows7 function 86
IsEngineActive 79	IsWindows8 86
IsEngineActive function 79	IsWindows8 function 86
IsFemaleA 80	IsWindowsVista 86
IsFemaleA function 80	IsWindowsVista function 86
IsFemaleW 80	IsWindowsXP 87
IsFemaleW function 80	IsWindowsXP function 87
IsMaleA 80	IsWindowsXPSP2 87
IsMaleA function 80	IsWindowsXPSP2 function 87
IsMaleW 81	IsWow64 87
IsMaleW function 81	IsWow64 function 87
IsMediaCenter 81	
IsMediaCenter function 81	
IsMetroActive 81	
IsMetroActive function 81	
IsMultiTouchReady 82	
IsMultiTouchReady function 82	
IsNativeWin64 82	

## L

LayeredWndProcA 87
LayeredWndProcA function 87
LayeredWndProcW 87
LayeredWndProcW function 87
LoadBitmapJPG 88

LoadBitmapJPG function 88  
LoadBitmapPNG 88  
LoadBitmapPNG function 88  
LoadCertificateA 88  
LoadCertificateA function 88  
LoadCertificateW 89  
LoadCertificateW function 89  
LoadIdentityA 89  
LoadIdentityA function 89  
LoadIdentityW 89  
LoadIdentityW function 89  
LoadPhotoA 90  
LoadPhotoA function 90  
LoadPhotoW 90  
LoadPhotoW function 90  
LoadPNGResource 90  
LoadPNGResource function 90

## M

Macros 160  
MakeCompatibleBitmap 91  
MakeCompatibleBitmap function 91  
MakeSoundFromFileA 91  
MakeSoundFromFileA function 91  
MakeSoundFromFileW 91  
MakeSoundFromFileW function 91  
MakeSoundFromResourceA 92  
MakeSoundFromResourceA function 92  
MakeSoundFromResourceW 92  
MakeSoundFromResourceW function 92

## N

NationalityConverter.h 184

## P

PEidAddressA 150  
PEidAddressA structure 150  
PEidAddressW 150  
PEidAddressW structure 150  
PEidCertificate 150  
PEidCertificate structure 150  
PEidIdentityA 151

PEidIdentityA structure 151  
PEidIdentityExA 152  
PEidIdentityExA structure 152  
PEidIdentityExW 154  
PEidIdentityExW structure 154  
PEidIdentityW 155  
PEidIdentityW structure 155  
PeidPicture 157  
PeidPicture structure 157  
PErrorInformation 157  
PErrorInformation structure 157  
PointsToPixels 92  
PointsToPixels function 92  
PortAvailable 93  
PortAvailable function 93  
PSISRecordA 157  
PSISRecordA structure 157  
PSISRecordW 158  
PSISRecordW structure 158

## Q

quicol.h 184

## R

ReadAddressA 93  
ReadAddressA function 93  
ReadAddressExA 93  
ReadAddressExA function 93  
ReadAddressExW 94  
ReadAddressExW function 94  
ReadAddressW 94  
ReadAddressW function 94  
ReadAuthenticationCertificate 94  
ReadAuthenticationCertificate function 94  
ReadAuthenticationCertificateEx 95  
ReadAuthenticationCertificateEx function 95  
ReadBufferFromFileA 95  
ReadBufferFromFileA function 95  
ReadBufferFromFileW 95  
ReadBufferFromFileW function 95  
ReadCaCertificate 96  
ReadCaCertificate function 96

ReadCaCertificateEx 96  
ReadCaCertificateEx function 96  
ReadIdentityA 97  
ReadIdentityA function 97  
ReadIdentityExA 97  
ReadIdentityExA function 97  
ReadIdentityExW 97  
ReadIdentityExW function 97  
ReadIdentityW 98  
ReadIdentityW function 98  
ReadNonRepudiationCertificate 98  
ReadNonRepudiationCertificate function 98  
ReadNonRepudiationCertificateEx 99  
ReadNonRepudiationCertificateEx function 99  
ReadPhoto 99  
ReadPhoto function 99  
ReadPhotoAsBitmap 99  
ReadPhotoAsBitmap function 99  
ReadPhotoAsBitmapEx 100  
ReadPhotoAsBitmapEx function 100  
ReadPhotoEx 100  
ReadPhotoEx function 100  
ReadRootCaCertificate 101  
ReadRootCaCertificate function 101  
ReadRootCaCertificateEx 101  
ReadRootCaCertificateEx function 101  
ReadRrnCertificate 101  
ReadRrnCertificate function 101  
ReadRrnCertificateEx 102  
ReadRrnCertificateEx function 102  
ReadSISCardA 102  
ReadSISCardA function 102  
ReadSISCardExA 103  
ReadSISCardExA function 103  
ReadSISCardExW 103  
ReadSISCardExW function 103  
ReadSISCardW 103  
ReadSISCardW function 103  
RecycleBinEmpty 104  
RecycleBinEmpty function 104  
ReloadReadersList 104  
ReloadReadersList function 104

RemoveCallback 104  
RemoveCallback function 104  
RemoveStartupA 105  
RemoveStartupA function 105  
RemoveStartupW 105  
RemoveStartupW function 105  
RestoreWindowSubclassA 105  
RestoreWindowSubclassA function 105  
RestoreWindowSubclassW 106  
RestoreWindowSubclassW function 106

## S

SaveAuthenticationCertificateA 106  
SaveAuthenticationCertificateA function 106  
SaveAuthenticationCertificateExW 106  
SaveAuthenticationCertificateExW function 106  
SaveAuthenticationCertificateW 107  
SaveAuthenticationCertificateW function 107  
SaveCaCertificateA 107  
SaveCaCertificateA function 107  
SaveCaCertificateExW 107  
SaveCaCertificateExW function 107  
SaveCaCertificateW 108  
SaveCaCertificateW function 108  
SaveCardToToXMLStreamEx 174  
SaveCardToToXMLStreamEx macro 174  
SaveCardToToXMLStreamExA 108  
SaveCardToToXMLStreamExA function 108  
SaveCardToToXMLStreamExW 108  
SaveCardToToXMLStreamExW function 108  
SaveCardToXmlA 109  
SaveCardToXmlA function 109  
SaveCardToXmlExA 109  
SaveCardToXmlExA function 109  
SaveCardToXmlExW 110  
SaveCardToXmlExW function 110  
SaveCardToXmlW 110  
SaveCardToXmlW function 110  
SaveContainer 111  
SaveContainer function 111  
SaveIdentityA 111  
SaveIdentityA function 111

SaveIdentityW 111	SaveRootCaCertificateA 119
SaveIdentityW function 111	SaveRootCaCertificateA function 119
SaveNonRepudiationCertificateA 112	SaveRootCaCertificateExW 119
SaveNonRepudiationCertificateA function 112	SaveRootCaCertificateExW function 119
SaveNonRepudiationCertificateExW 112	SaveRootCaCertificateW 120
SaveNonRepudiationCertificateExW function 112	SaveRootCaCertificateW function 120
SaveNonRepudiationCertificateW 112	SaveRrnCertificateA 120
SaveNonRepudiationCertificateW function 112	SaveRrnCertificateA function 120
SavePersonCsvToStream 174	SaveRrnCertificateExW 120
SavePersonCsvToStream macro 174	SaveRrnCertificateExW function 120
SavePersonCsvToStreamA 113	SaveRrnCertificateW 121
SavePersonCsvToStreamA function 113	SaveRrnCertificateW function 121
SavePersonCsvToStreamW 113	SelectReader 121
SavePersonCsvToStreamW function 113	SelectReader function 121
SavePersonToCsvA 114	SelectReaderByNameA 121
SavePersonToCsvA function 114	SelectReaderByNameA function 121
SavePersonToCsvExA 114	SelectReaderByNameW 122
SavePersonToCsvExA function 114	SelectReaderByNameW function 122
SavePersonToCsvExW 114	SendAPDU 122
SavePersonToCsvExW function 114	SendAPDU function 122
SavePersonToCsvW 115	SetCallback 122
SavePersonToCsvW function 115	SetCallback function 122
SavePhotoA 115	SetMWCompatibility 123
SavePhotoA function 115	SetMWCompatibility function 123
SavePhotoAsBitmapA 116	SetStartupA 123
SavePhotoAsBitmapA function 116	SetStartupA function 123
SavePhotoAsBitmapExA 116	SetStartupW 123
SavePhotoAsBitmapExA function 116	SetStartupW function 123
SavePhotoAsBitmapExW 116	SetSupportSIS 124
SavePhotoAsBitmapExW function 116	SetSupportSIS function 124
SavePhotoAsBitmapW 117	ShellCopyFileA 124
SavePhotoAsBitmapW function 117	ShellCopyFileA function 124
SavePhotoAsJpegA 117	ShellCopyFileW 125
SavePhotoAsJpegA function 117	ShellCopyFileW function 125
SavePhotoAsJpegExA 117	ShutdownWindows 125
SavePhotoAsJpegExA function 117	ShutdownWindows function 125
SavePhotoAsJpegExW 118	SIS_FIELD_MAX_BIRTHDATE_LEN 174
SavePhotoAsJpegExW function 118	SIS_FIELD_MAX_BIRTHDATE_LEN macro 174
SavePhotoAsJpegW 118	SIS_FIELD_MAX_CAPTUREDATE_LEN 174
SavePhotoAsJpegW function 118	SIS_FIELD_MAX_CAPTUREDATE_LEN macro 174
SavePhotoW 119	SIS_FIELD_MAX_CARDNUMBER_LEN 175
SavePhotoW function 119	SIS_FIELD_MAX_CARDNUMBER_LEN macro 175

SIS_FIELD_MAX_SOCIAL_SECURITY_NUMBER_LEN 175	tagEidAddressA 133
SIS_FIELD_MAX_SOCIAL_SECURITY_NUMBER_LEN macro 175	tagEidAddressA structure 133
SIS_FIELD_MAX_VALIDBEGIN_LEN 175	tagEidAddressW 133
SIS_FIELD_MAX_VALIDBEGIN_LEN macro 175	tagEidAddressW structure 133
SIS_FIELD_MAX_VALIDEND_LEN 175	tagEidCertificate 133
SIS_FIELD_MAX_VALIDEND_LEN macro 175	tagEidCertificate structure 133
SIS_MAX_CARDNAME_LEN 176	tagEidIdentityA 134
SIS_MAX_CARDNAME_LEN macro 176	tagEidIdentityA structure 134
SIS_MAX_FIRSTNAMES_LEN 176	tagEidIdentityExA 135
SIS_MAX_FIRSTNAMES_LEN macro 176	tagEidIdentityExA structure 135
SIS_MAX_INITIAL_LEN 176	tagEidIdentityExW 137
SIS_MAX_INITIAL_LEN macro 176	tagEidIdentityExW structure 137
SIS_MAX_NAME_LEN 176	tagEidIdentityW 138
SIS_MAX_NAME_LEN macro 176	tagEidIdentityW structure 138
SIS_MAX_SEX_LEN 176	tagEidPicture 140
SIS_MAX_SEX_LEN macro 176	tagEidPicture structure 140
SISRecordA 159	tagSISRecordA 140
SISRecordA structure 159	tagSISRecordA structure 140
SISRecordW 160	tagSISRecordW 141
SISRecordW structure 160	tagSISRecordW structure 141
StartEngine 126	TurnMonitorOff 127
StartEngine function 126	TurnMonitorOff function 127
StopEngine 126	TurnMonitorOn 128
StopEngine function 126	TurnMonitorOn function 128
StretchNativeBitmap 126	
StretchNativeBitmap function 126	
StripFileNameA 126	
StripFileNameA function 126	
StripFileNameW 127	
StripFileNameW function 127	
structErrorInformation 132	
structErrorInformation structure 132	
Structs, Records, Enums 131	
SuspendWindows 127	
SuspendWindows function 127	
Swelio.h 184	
System.h 189	
SystemInfo.h 190	

**T**

tagCardEventType 132

tagCardEventType enumeration 132

**U**

UpdateWindowPosition 128

UpdateWindowPosition function 128

**V**

VerifyPinA 128

VerifyPinA function 128

VerifyPinExA 128

VerifyPinExA function 128

VerifyPinExW 129

VerifyPinExW function 129

VerifyPinW 129

VerifyPinW function 129

VerifySignature 130

VerifySignature function 130

## W

WIDTHBYTES 177

WIDTHBYTES macro 177

WriteBufferToFileA 130

WriteBufferToFileA function 130

WriteBufferToFileW 130

WriteBufferToFileW function 130