# Swelio Library

Belgian Electronic ID card access library

# Table of Contents

# Index a

# 1 Symbol Reference

## 1.1 Functions

The following table lists functions in this documentation.

**Functions**

| | Name | Description |
|---|---|---|
| | ActivateCard (see page 10) | Established communication between the card and the reader |
| | ActivateCardEx (see page 11) | Established communication between the card and the reader |
| | AddRemoveMessageFilter (see page 11) | Adds or removes a message from the User Interface Privilege Isolation (UIPI) message filter. |
| | AllocateBuffer (see page 12) | Allocates the buffer in memory |
| | AllocateDefaultHWND (see page 12) | This function creates the invisible tool window |
| | AllocateDefaultHWNDA (see page 12) | This function creates the invisible tool window |
| | AllocateDefaultHWNDW (see page 13) | This function creates the invisible tool window |
| | AllocateHWND (see page 13) | This function creates the invisible tool window using the provided window procedure |
| | AllocateHWNDA (see page 13) | This function creates the invisible tool window using the provided window procedure |
| | AllocateHWNDW (see page 14) | This function creates the invisible tool window using the provided window procedure |
| | AllocateLayeredWindow (see page 14) | This function creates the layered window using the provided window class name |
| | AllocateLayeredWindowA (see page 14) | This function creates the layered window using the provided window class name |
| | AllocateLayeredWindowW (see page 15) | This function creates the layered window using the provided window class name |
| | AllocateWindowClass (see page 15) | This function creates the standard window using the provided window class name |
| | AllocateWindowClassA (see page 15) | This function creates the standard window using the provided window class name |
| | AllocateWindowClassW (see page 16) | This function creates the standard window using the provided window class name |
| | BringWindowToFront (see page 16) | This function brings the specified window to the top of the z-order. |
| | CardDecryptFile (see page 16) | Decrypt file using Belgian Id card |
| | CardDecryptFileA (see page 17) | Decrypt file using Belgian Id card |
| | CardDecryptFileW (see page 17) | Decrypt file using Belgian Id card |
| | CardEncryptFile (see page 17) | Encrypt file using Belgian Id card |
| | CardEncryptFileA (see page 18) | Encrypt file using Belgian Id card |
| | CardEncryptFileW (see page 18) | Encrypt file using Belgian Id card |
| | CardSignCadesT (see page 18) | Sign data with eID card according to CADES-T standard |
| | CardSignCMS (see page 19) | Sign data with eID card according to CMS standard |

| | | |
|---|---|---|
| | CertSignCadesT (see page 19) | Sign data with certificate according to CADES-T standard |
| | CertSignCMS (see page 20) | Sign data with certificate according to CMS standard |
| | CheckMD5 (see page 20) | Checks the MD5 hash value of the memory buffer |
| | CheckSHA1 (see page 21) | Checks the SHA1 hash value of the memory buffer |
| | ClearFileAttributes (see page 21) | This function sets the file attributes to normal. |
| | ClearFileAttributesA (see page 22) | This function sets the file attributes to normal. |
| | ClearFileAttributesW (see page 22) | This function sets the file attributes to normal. |
| | ClearUnusedMemory (see page 22) | Clears unused memory and minimized the application memory usage |
| | CreateUnicodeFile (see page 23) | Creates UNICODE file |
| | CreateUnicodeFileA (see page 23) | Creates UNICODE file |
| | CreateUnicodeFileW (see page 23) | Creates UNICODE file |
| | CurrentIPAddress (see page 24) | Returns the IP address |
| | CurrentIPAddressA (see page 24) | Returns the IP address |
| | CurrentIPAddressW (see page 24) | Returns the IP address |
| | DeactivateCard (see page 24) | Terminates a cennection between a smart card and a reader |
| | DeactivateCardEx (see page 25) | Terminates a connection between a smart card and a reader |
| | DeallocateBuffer (see page 25) | Deallocates the memory buffer |
| | DeallocateHWND (see page 25) | This function destroys the specified window. |
| | DeallocateHWNDA (see page 26) | This function destroys the specified window. |
| | DeallocateHWNDW (see page 26) | This function destroys the specified window. |
| | DecryptFileAES (see page 26) | Decrypts file using AES algorithm. |
| | DecryptFileAESA (see page 27) | Decrypts file using AES algorithm. |
| | DecryptFileAESW (see page 27) | Decrypts file using AES algorithm. |
| | DeleteToRecycleBin (see page 28) | Deletes file to WIndows Recycle Bin |
| | DeleteToRecycleBinA (see page 28) | Deletes file to the Windows Recycle Bin |
| | DeleteToRecycleBinW (see page 28) | Deletes file to the Windows Recycle Bin |
| | DestroyImageBuffer (see page 29) | Destroys the memory buffer |
| | DirectoryExists (see page 29) | Determines whether a specified directory exists. |
| | DirectoryExistsA (see page 30) | Determines whether a specified directory exists. |
| | DirectoryExistsW (see page 30) | Determines whether a specified directory exists. |
| | DisplayCertificate (see page 30) | Displays the dialog window with certificate information |
| | DocumentTypeToString (see page 31) | Returns the textual representation of the card type (in English) |
| | DrawLayeredWindow (see page 31) | Repaints the surface of the layered window |
| | EmptyRecycleBin (see page 32) | Empties the recycle bin |
| | EncodeCertificate (see page 32) | Performs Base64 encoding of the certificate |
| | EncodePhoto (see page 32) | Performs Base64 encoding of the photo |
| | EncryptFileAES (see page 33) | Encrypts file using AES algorithm. |
| | EncryptFileAESA (see page 33) | Encrypts file using AES algorithm. |
| | EncryptFileAESW (see page 34) | Encrypts file using AES algorithm. |

| | | |
|---|---|---|
| | FileClose (see page 34) | Concludes input/output (I/O) to a file opened using the FileCreateRewrite (see page 36) function. |
| | FileCloseA (see page 34) | Concludes input/output (I/O) to a file opened using the FileCreateRewrite (see page 36) function. |
| | FileCloseW (see page 35) | Concludes input/output (I/O) to a file opened using the FileCreateRewrite (see page 36) function. |
| | FileCopy (see page 35) | The CopyFile function copies an existing file to a new file. |
| | FileCopyA (see page 36) | The CopyFile function copies an existing file to a new file. |
| | FileCopyW (see page 36) | The CopyFile function copies an existing file to a new file. |
| | FileCreateRewrite (see page 36) | Creates new or overwrites existing file |
| | FileCreateRewriteA (see page 37) | Creates new or overwrites existing file |
| | FileCreateRewriteW (see page 37) | Creates new or overwrites existing file |
| | FileDelete (see page 37) | Deletes a file from disk. |
| | FileDeleteA (see page 38) | Deletes a file from disk. |
| | FileDeleteW (see page 38) | Deletes a file from disk. |
| | FileExists (see page 38) | Tests whether a specified file exists. |
| | FileExistsA (see page 39) | Tests whether a specified file exists. |
| | FileExistsW (see page 39) | Tests whether a specified file exists. |
| | FileExtensionIs (see page 39) | Checks the file extension |
| | FileExtensionIsA (see page 40) | Checks the file extension |
| | FileExtensionIsW (see page 40) | Checks the file extension |
| | FileGetSize (see page 41) | Retrieves the size of a specified file. |
| | FileGetSizeA (see page 41) | Retrieves the size of a specified file. |
| | FileGetSizeW (see page 42) | Retrieves the size of a specified file. |
| | FileIsExe (see page 42) | Checks if the file is a Windows executable |
| | FileIsExeA (see page 42) | Checks if the file is a Windows executable |
| | FileIsExeW (see page 43) | Checks if the file is a Windows executable |
| | FileIsIcon (see page 43) | Checks if the file is a Windows icon (.ico) file |
| | FileIsIconA (see page 43) | Checks if the file is a Windows icon (.ico) file |
| | FileIsIconW (see page 44) | Checks if the file is a Windows icon (.ico) file |
| | FileIsImage (see page 44) | Checks if the file is an image file |
| | FileIsImageA (see page 44) | Checks if the file is an image file |
| | FileIsImageW (see page 45) | Checks if the file is an image file |
| | FileOrFolderExists (see page 45) | Checks if the file or folder with the given name exists |
| | FileOrFolderExistsA (see page 46) | Checks if the file or folder with the given name exists |
| | FileOrFolderExistsW (see page 46) | Checks if the file or folder with the given name exists |
| | FileRename (see page 46) | Renames the file |
| | FileRenameA (see page 47) | Renames the file |
| | FileRenameW (see page 47) | Renames the file |
| | FileWrite (see page 47) | Writes string to the file |
| | FileWriteA (see page 48) | Writes string to the file |
| | FileWriteChar (see page 48) | Writes one character to the file |
| | FileWriteCharA (see page 48) | Writes one character to the file |
| | FileWriteCharW (see page 49) | Writes one character to the file |
| | FileWriteNewLine (see page 49) | Writes new line sequence to the file |
| | FileWriteNewLineA (see page 49) | Writes new line sequence to the file |
| | FileWriteNewLineW (see page 49) | Writes new line sequence to the file |

| | | |
|---|---|---|
| | FileWriteW (see page 50) | Writes string to the file |
| | FormatCardNumber (see page 50) | Format card number string for better visualization |
| | FormatEIDDate (see page 50) | Converts the national number value to its formatted String representation |
| | FormatNationalNumber (see page 51) | Format the national number string for better visualization |
| | FullPath (see page 51) | Gets the full path to the file based on file name |
| | FullPathA (see page 51) | Gets the full path to the file based on file name |
| | FullPathW (see page 52) | Gets the full path to the file based on file name |
| | GenerateAuthenticationSignature (see page 52) | Generate authentication signature |
| | GenerateAuthenticationSignatureA (see page 53) | Generate authentication signature |
| | GenerateAuthenticationSignatureEx (see page 53) | Generate authentication signature |
| | GenerateAuthenticationSignatureExA (see page 54) | Generate authentication signature |
| | GenerateAuthenticationSignatureExW (see page 54) | Generate authentication signature |
| | GenerateAuthenticationSignatureW (see page 55) | Generate authentication signature |
| | GenerateBMP (see page 55) | Generates Windows Bitmap file with QR Code image |
| | GenerateBMPA (see page 56) | Generates Windows Bitmap file with QR Code image |
| | GenerateBMPW (see page 56) | Generates Windows Bitmap file with QR Code image |
| | GenerateNonRepudiationSignature (see page 56) | Generate non repudiation signature |
| | GenerateNonRepudiationSignatureA (see page 57) | Generate non repudiation signature |
| | GenerateNonRepudiationSignatureEx (see page 57) | Generate non repudiation signature |
| | GenerateNonRepudiationSignatureExA (see page 58) | Generate non repudiation signature |
| | GenerateNonRepudiationSignatureExW (see page 59) | Generate non repudiation signature |
| | GenerateNonRepudiationSignatureW (see page 59) | Generate non repudiation signature |
| | GeneratePNG (see page 60) | Generates PNG file with QR Code image |
| | GeneratePNGA (see page 60) | Generates PNG file with QR Code image |
| | GeneratePNGW (see page 60) | Generates PNG file with QR Code image |
| | GenerateQRCode (see page 61) | Read eID card and save the identity information and address to PNG QR Code file |
| | GenerateQRCodeA (see page 61) | Read eID card and save the identity information and address to PNG QR Code file |
| | GenerateQRCodeEx (see page 62) | Read eID card and save the identity information and address to PNG QR Code file |
| | GenerateQRCodeExA (see page 62) | Read eID card and save the identity information and address to PNG QR Code file |
| | GenerateQRCodeExW (see page 63) | Read eID card and save the identity information and address to PNG QR Code file |
| | GenerateQRCodeW (see page 63) | Read eID card and save the identity information and address to PNG QR Code file |
| | GetCardSerialNumber (see page 63) | Gets the card serial number |

| | | |
|---|---|---|
| | GetEncodedCertificateSize (see page 64) | Returns the size of the Base64 encoded certificate |
| | GetEncodedPhotoSize (see page 64) | Calculates buffer size for Base64 encoded photo |
| | GetFileMD5 (see page 65) | Gets the MD5 hash value for the file |
| | GetFileMD5A (see page 65) | Gets the MD5 hash value for the file |
| | GetFileMD5W (see page 65) | Gets the MD5 hash value for the file |
| | GetFilesCount (see page 66) | Calculates the number of files in the given folder |
| | GetFilesCountA (see page 66) | Calculates the number of files in the given folder |
| | GetFilesCountW (see page 67) | Calculates the number of files in the given folder |
| | GetFileSHA1 (see page 67) | Gets the SHA1 hash value for the file |
| | GetFileSHA1A (see page 67) | Gets the SHA1 hash value for the file |
| | GetFileSHA1W (see page 68) | Gets the SHA1 hash value for the file |
| | GetHBitmap (see page 68) | Generates Windows Bitmap in memory with QR Code image |
| | GetHBitmapA (see page 69) | Generates Windows Bitmap in memory with QR Code image |
| | GetHBitmapW (see page 69) | Generates Windows Bitmap in memory with QR Code image |
| | GetISOCode (see page 70) | Returns the country ISO code based on the nationality string |
| | GetISOCodeA (see page 70) | Returns the country ISO code based on the nationality string |
| | GetISOCodeW (see page 71) | Returns the country ISO code based on the nationality string |
| | GetMD5 (see page 71) | Gets the MD5 hash value for the content of the memory buffer |
| | GetPNG (see page 72) | Writes PNG image to the memory buffer. |
| | GetPNGA (see page 72) | Writes PNG image to the memory buffer. |
| | GetPNGW (see page 73) | Writes PNG image to the memory buffer. |
| | GetReaderIndex (see page 73) | Returns the zero-based reader index with specified name |
| | GetReaderIndexA (see page 73) | Returns the zero-based reader index with specified name |
| | GetReaderIndexW (see page 74) | Returns the zero-based reader index with specified name |
| | GetReaderName (see page 74) | Returns the name of the reader |
| | GetReaderNameA (see page 75) | Returns the name of the reader |
| | GetReaderNameLen (see page 75) | Returns the length of the reader name |
| | GetReaderNameLenA (see page 75) | Returns the length of the reader name |
| | GetReaderNameLenW (see page 76) | Returns the length of the reader name |
| | GetReaderNameW (see page 76) | Returns the name of the reader |
| | GetReadersCount (see page 76) | Get number of card readers connected to PC |
| | GetSelectedReaderIndex (see page 77) | Returns the index of the active smart card reader |
| | GetSHA1 (see page 77) | Gets the SHA1 hash value for the content of the memory buffer |
| | GetStartup (see page 78) | Checks if the application is registered to run when Windows starts |
| | GetStartupA (see page 78) | Checks if the application is registered to run when Windows starts |
| | GetStartupW (see page 78) | Checks if the application is registered to run when Windows starts |
| | GetSupportSIS (see page 78) | Checks if the SIS cards are supported by the engine |
| | HibernateWindows (see page 79) | Hibernates Windows |
| | IsAnimatedGIF (see page 79) | Checks if the file is an animated GIF image file |
| | IsAnimatedGIFA (see page 79) | Checks if the file is an animated GIF image file |
| | IsAnimatedGIFW (see page 80) | Checks if the file is an animated GIF image file |
| | IsCardPresent (see page 80) | Checks if the card is present in the card reader |
| | IsCardPresentEx (see page 80) | Checks if the card is present in the card reader |
| | IsConnectedToInternet (see page 81) | Checks if PC is connected to Internet |
| | IsDirectory (see page 81) | Verifies that a path is a valid directory. |
| | IsDirectoryA (see page 81) | Verifies that a path is a valid directory. |
| | IsDirectoryW (see page 82) | Verifies that a path is a valid directory. |

| | | |
|---|---|---|
| | IsEIDCard (see page 82) | Check if Belgian EID card is inserted into card reader |
| | IsEIDCardEx (see page 82) | Check if Belgian EID card is inserted into card reader |
| | IsEngineActive (see page 83) | Checks if the Swelio Engine is activated |
| | IsFemale (see page 83) | Checks if the card owner is female |
| | IsFemaleA (see page 83) | Checks if the card owner is female |
| | IsFemaleW (see page 84) | Checks if the card owner is female |
| | IsMale (see page 84) | Checks if the card owner is male |
| | IsMaleA (see page 85) | Checks if the card owner is male |
| | IsMaleW (see page 85) | Checks if the card owner is male |
| | IsMediaCenter (see page 85) | Checks if the Media Center version of Windows is installed |
| | IsMetroActive (see page 86) | Checks if metro interface is active |
| | IsMultiTouchReady (see page 86) | Checks if the system is multi touch ready |
| | IsNativeWin64 (see page 86) | Checks if the application is native 64 bit executable |
| | IsSISCard (see page 86) | Check if Belgian SIS card is inserted into card reader |
| | IsSISCardEx (see page 87) | Check if Belgian SIS card is inserted into card reader |
| | IsTabletPC (see page 87) | Checks if the application is running on the Tablet PC |
| | IsUnicodeFile (see page 87) | Checks if the file is UNICODE file |
| | IsUnicodeFileA (see page 88) | Checks if the file is UNICODE file |
| | IsUnicodeFileW (see page 88) | Checks if the file is UNICODE file |
| | IsValidFileName (see page 88) | Checks if provided string is a valid file name |
| | IsValidFileNameA (see page 89) | Checks if provided string is a valid file name |
| | IsValidFileNameW (see page 89) | Checks if provided string is a valid file name |
| | IsValidPathName (see page 90) | Checks if provided string is a valid file path |
| | IsValidPathNameA (see page 90) | Checks if provided string is a valid file path |
| | IsValidPathNameW (see page 90) | Checks if provided string is a valid file path |
| | IsWindows7 (see page 91) | Checks if PC is running Windows 7 or better |
| | IsWindows8 (see page 91) | Checks if PC is Running Windows 8 or better |
| | IsWindowsVista (see page 91) | Checks if PC is running Windows Vista or better |
| | IsWindowsXP (see page 91) | Checks if PC is running Windows XP |
| | IsWindowsXPSP2 (see page 92) | Checks if PC is running Windows XP with Service Pack 2 installed |
| | IsWow64 (see page 92) | Checks if the 32 bit application runs on 64 bit Windows |
| | LayeredWndProc (see page 92) | The default window procedure for the layered window |
| | LayeredWndProcA (see page 92) | The default window procedure for the layered window |
| | LayeredWndProcW (see page 93) | The default window procedure for the layered window |
| | LoadCertificate (see page 93) | Reads the certificate from a file |
| | LoadCertificateA (see page 93) | Reads the certificate from a file |
| | LoadCertificateW (see page 94) | Reads the certificate from a file |
| | LoadIdentity (see page 94) | Reads the raw identity information from a file |
| | LoadIdentityA (see page 94) | Reads the raw identity information from a file |
| | LoadIdentityW (see page 95) | Reads the raw identity information from a file |
| | LoadPhoto (see page 95) | Loads photo from a file |
| | LoadPhotoA (see page 95) | Loads photo from a file |
| | LoadPhotoW (see page 96) | Loads photo from a file |
| | MakeSoundFromFile (see page 96) | Plays the wave sound from the file |
| | MakeSoundFromFileA (see page 96) | Plays the wave sound from the file |
| | MakeSoundFromFileW (see page 97) | Plays the wave sound from the file |
| | MakeSoundFromResource (see page 97) | Plays the wave sound from the resource |

| | | |
|---|---|---|
| | MakeSoundFromResourceA (see page 97) | Plays the wave sound from the resource |
| | MakeSoundFromResourceW (see page 98) | Plays the wave sound from the resource |
| | PortAvailable (see page 98) | Checks if the port with specified number is available |
| | ReadAddress (see page 98) | Read address information from Belgian eID card |
| | ReadAddressA (see page 99) | Read address information from Belgian eID card |
| | ReadAddressEx (see page 99) | Read address information from Belgian eID card |
| | ReadAddressExA (see page 100) | Read address information from Belgian eID card |
| | ReadAddressExW (see page 100) | Read address information from Belgian eID card |
| | ReadAddressW (see page 100) | Read address information from Belgian eID card |
| | ReadAuthenticationCertificate (see page 101) | Read Authentication Certificate to memory |
| | ReadBufferFromFile (see page 101) | Reads the content of the file to the memory buffer |
| | ReadBufferFromFileA (see page 101) | Reads the content of the file to the memory buffer |
| | ReadBufferFromFileW (see page 102) | Reads the content of the file to the memory buffer |
| | ReadCaCertificate (see page 102) | Read Ca Certificate to memory |
| | ReadIdentity (see page 103) | Read identity information from Belgian eID card |
| | ReadIdentityA (see page 103) | Read identity information from Belgian eID card |
| | ReadIdentityEx (see page 103) | Read identity information from Belgian eID card |
| | ReadIdentityExA (see page 104) | Read identity information from Belgian eID card |
| | ReadIdentityExW (see page 104) | Read identity information from Belgian eID card |
| | ReadIdentityW (see page 104) | Read identity information from Belgian eID card |
| | ReadNonRepudiationCertificate (see page 105) | Read Non Repudiation Certificate to memory |
| | ReadPhoto (see page 105) | Reads a photo from a card |
| | ReadPhotoAsBitmap (see page 106) | Reads the picture from the card, converts it to bitmap and returns the bitmap handle Decription: 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 106) | Reads the picture from the card, converts it to bitmap and returns the bitmap handle Decription: 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 106) | Reads a photo from a card |
| | ReadRootCaCertificate (see page 107) | Read Root Ca Certificate to memory |
| | ReadRrnCertificate (see page 107) | Read Rrn Certificate to memory |
| | ReadSISCard (see page 107) | Read Belgian SIS card. |
| | ReadSISCardA (see page 108) | Read Belgian SIS card. |
| | ReadSISCardEx (see page 108) | Read Belgian SIS card. |
| | ReadSISCardExA (see page 109) | Read Belgian SIS card. |
| | ReadSISCardExW (see page 109) | Read Belgian SIS card. |
| | ReadSISCardW (see page 110) | Read Belgian SIS card. |
| | ReloadReadersList (see page 110) | Reloads the list of the available card readers |
| | RemoveCallback (see page 110) | Remove callback procedure for card events |
| | RemoveStartup (see page 111) | Removes the application from the list of the automatically started applications |
| | RemoveStartupA (see page 111) | Removes the application from the list of the automatically started applications |
| | RemoveStartupW (see page 111) | Removes the application from the list of the automatically started applications |

| | | |
|---|---|---|
| | RestoreWindowSubclass (see page 112) | Restores window standard procedure |
| | RestoreWindowSubclassA (see page 112) | Restores window standard procedure |
| | RestoreWindowSubclassW (see page 112) | Restores window standard procedure |
| | SaveAuthenticationCertificate (see page 112) | Save Authentication Certificate to a file |
| | SaveAuthenticationCertificateA (see page 113) | Save Authentication Certificate to a file |
| | SaveAuthenticationCertificateW (see page 113) | Save Authentication Certificate to a file |
| | SaveCaCertificate (see page 113) | Save Ca Certificate to a file |
| | SaveCaCertificateA (see page 114) | Save Ca Certificate to a file |
| | SaveCaCertificateW (see page 114) | Save Ca Certificate to a file |
| | SaveCardToXml (see page 115) | Read eID card and save the information to XML file |
| | SaveCardToXmlA (see page 115) | Read eID card and save the information to XML file |
| | SaveCardToXmlEx (see page 115) | Read eID card and save the information to XML file |
| | SaveCardToXmlExA (see page 116) | Read eID card and save the information to XML file |
| | SaveCardToXmlExW (see page 116) | Read eID card and save the information to XML file |
| | SaveCardToXmlW (see page 117) | Read eID card and save the information to XML file |
| | SaveIdentity (see page 117) | Saves indentity infornation to a file |
| | SaveIdentityA (see page 117) | Saves indentity infornation to a file |
| | SaveIdentityW (see page 118) | Saves indentity infornation to a file |
| | SaveNonRepudiationCertificate (see page 118) | Save Non Repudiation Certificate to a file |
| | SaveNonRepudiationCertificateA (see page 118) | Save Non Repudiation Certificate to a file |
| | SaveNonRepudiationCertificateW (see page 119) | Save Non Repudiation Certificate to a file |
| | SavePersonToCsv (see page 119) | Read eID card and save the identity information and address to CSV file |
| | SavePersonToCsvA (see page 120) | Read eID card and save the identity information and address to CSV file |
| | SavePersonToCsvEx (see page 120) | Read eID card and save the identity information and address to CSV file |
| | SavePersonToCsvExA (see page 120) | Read eID card and save the identity information and address to CSV file |
| | SavePersonToCsvExW (see page 121) | Read eID card and save the identity information and address to CSV file |
| | SavePersonToCsvW (see page 121) | Read eID card and save the identity information and address to CSV file |
| | SavePhoto (see page 122) | Saves photo to a file |
| | SavePhotoA (see page 122) | Saves photo to a file |
| | SavePhotoAsBitmap (see page 122) | 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. |
| | SavePhotoAsBitmapA (see page 123) | 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. |

| | | |
|---|---|---|
| | SavePhotoAsBitmapEx (see page 123) | 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. |
| | SavePhotoAsBitmapExA (see page 123) | 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 124) | 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 124) | 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. |
| | SavePhotoAsJpeg (see page 125) | 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. |
| | SavePhotoAsJpegA (see page 125) | 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. |
| | SavePhotoAsJpegEx (see page 125) | 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 126) | 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 126) | 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. |
| | SavePhotoAsJpegW (see page 127) | 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. |
| | SavePhotoW (see page 127) | Saves photo to a file |
| | SaveRootCaCertificate (see page 127) | Save Root Ca Certificate to a file |
| | SaveRootCaCertificateA (see page 128) | Save Root Ca Certificate to a file |
| | SaveRootCaCertificateW (see page 128) | Save Root Ca Certificate to a file |
| | SaveRrnCertificate (see page 128) | Save RRN Certificate to a file |
| | SaveRrnCertificateA (see page 129) | Save RRN Certificate to a file |
| | SaveRrnCertificateW (see page 129) | Save RRN Certificate to a file |
| | SelectReader (see page 129) | When more than 1 reader connected, select the reader with specified number The first reader has number 0 |
| | SelectReaderByName (see page 130) | Select active smart card reader by providing the reader name |
| | SelectReaderByNameA (see page 130) | Select active smart card reader by providing the reader name |
| | SelectReaderByNameW (see page 131) | Select active smart card reader by providing the reader name |
| | SetCallback (see page 131) | Activates callback procedure for card status change event |

| | | |
|---|---|---|
| | SetMWCompatibility (see page 131) | Set the compatibility mode with the old version of the oficial EID MiddleWare |
| | SetStartup (see page 132) | Register application to run when Windows starts |
| | SetStartupA (see page 132) | Register application to run when Windows starts |
| | SetStartupW (see page 132) | Register application to run when Windows starts |
| | SetSupportSIS (see page 133) | Activates or deactivates SIS card support by engine |
| | ShellCopyFile (see page 133) | Copies file to the new location |
| | ShellCopyFileA (see page 133) | Copies file to the new location |
| | ShellCopyFileW (see page 134) | Copies file to the new location |
| | ShowError (see page 134) | Shows Dialog with the text message corresponding to the Windows error code |
| | ShutdownWindows (see page 134) | Logs off the interactive user, shuts down the system. |
| | StartEngine (see page 135) | Activates the Swelio Engine. |
| | StopEngine (see page 135) | Deactivates the Swelio Engine |
| | StripFileName (see page 135) | Replaces environment variable names with values |
| | StripFileNameA (see page 136) | Replaces environment variable names with values |
| | StripFileNameW (see page 136) | Replaces environment variable names with values |
| | SuspendWindows (see page 137) | Suspends Windows |
| | TurnMonitorOff (see page 137) | Turns the monitor off |
| | TurnMonitorOn (see page 137) | Turns the monitor on |
| | UpdateWindowPosition (see page 137) | Updated the window position |
| | VerifyPin (see page 138) | Verify PIN code |
| | VerifyPinA (see page 138) | Verify PIN code |
| | VerifyPinEx (see page 138) | Verify PIN code |
| | VerifyPinExA (see page 139) | Verify PIN code |
| | VerifyPinExW (see page 139) | Verify PIN code |
| | VerifyPinW (see page 139) | Verify PIN code |
| | VerifySignature (see page 140) | Verifies the signature from the specified hash value. |
| | WriteBufferToFile (see page 140) | Writes the memory buffer to file |
| | WriteBufferToFileA (see page 141) | Writes the memory buffer to file |
| | WriteBufferToFileW (see page 141) | Writes the memory buffer to file |

## 1.1.1 ActivateCard Function

Established communication between the card and the reader

**Pascal**

```
function ActivateCard: BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**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.

## 1.1.2 ActivateCardEx Function

Established communication between the card and the reader

**Pascal**

```
function ActivateCardEx(ReaderNumber: integer): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | 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 AddRemoveMessageFilter Function

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

**Pascal**

```
procedure AddRemoveMessageFilter(Message: UINT; dwFlags: DWORD); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Message: UINT | Specifies the message to add to or remove from the filter. |
| dwFlags: DWORD | Specifies the action to be performed. One of the following values.<br>• MSGFLT_ADD - Adds the message to the filter. This has the effect of allowing the message to be received.<br>• MSGFLT_REMOVE - Removes the message from the filter. This has the effect of blocking the message. |

**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.

## 1.1.4 AllocateBuffer Function

Allocates the buffer in memory

**Pascal**

```
function AllocateBuffer(BufferSize: integer): Pointer; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| BufferSize: integer | 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 25) function.

## 1.1.5 AllocateDefaultHWND Function

This function creates the invisible tool window

**Pascal**

```
function AllocateDefaultHWND: THandle; stdcall;
```

**File**

SwelioEngine (see page 162)

**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...

## 1.1.6 AllocateDefaultHWNDA Function

This function creates the invisible tool window

**Pascal**

```
function AllocateDefaultHWNDA: THandle; stdcall;
```

**File**

SwelioEngine (see page 162)

**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

**Pascal**

```
function AllocateDefaultHWNDW: THandle; stdcall;
```

**File**

SwelioEngine (see page 162)

**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 Function

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

**Pascal**

```
function AllocateHWND(WndFunc: TFNWndProc): THandle; stdcall;
```

**File**

SwelioEngine (see page 162)

**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 AllocateHWNDA Function

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

**Pascal**

```
function AllocateHWNDA(WndFunc: TFNWndProc): THandle; stdcall;
```

**File**

SwelioEngine (see page 162)

**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 AllocateHWNDW Function

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

**Pascal**

```
function AllocateHWNDW(WndFunc: TFNWndProc): THandle; stdcall;
```

**File**

SwelioEngine (see page 162)

**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...

## 1.1.11 AllocateLayeredWindow Function

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

**Pascal**

```
function AllocateLayeredWindow(const ClassName: PChar): THandle; stdcall;
```

**File**

SwelioEngine (see page 162)

**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 AllocateLayeredWindowA Function

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

**Pascal**

```
function AllocateLayeredWindowA(const ClassName: PAnsiChar): THandle; stdcall;
```

**File**

SwelioEngine (see page 162)

**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

## 1.1.13 AllocateLayeredWindowW Function

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

**Pascal**

```
function AllocateLayeredWindowW(const ClassName: PWideChar): THandle; stdcall;
```

**File**

SwelioEngine (see page 162)

**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

## 1.1.14 AllocateWindowClass Function

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

**Pascal**

```
function AllocateWindowClass(const ClassName: PChar): THandle; stdcall;
```

**File**

SwelioEngine (see page 162)

**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

## 1.1.15 AllocateWindowClassA Function

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

**Pascal**

```
function AllocateWindowClassA(const ClassName: PAnsiChar): THandle; stdcall;
```

**File**

SwelioEngine (see page 162)

**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

## 1.1.16 AllocateWindowClassW Function

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

**Pascal**

```
function AllocateWindowClassW(const ClassName: PWideChar): THandle; stdcall;
```

**File**

SwelioEngine (see page 162)

**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

## 1.1.17 BringWindowToFront Function

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

**Pascal**

```
procedure BringWindowToFront(Window: THandle); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Window: THandle | Handle to the window to bring to the top of the z-order. |

## 1.1.18 CardDecryptFile Function

Decrypt file using Belgian Id card

**Pascal**

```
function CardDecryptFile(Source: PChar; Destination: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| szSource | The name of the encrypted file |
| szDestination | The name of the decrypted file |

**Description**

Decrypt file which was encrypted using CardEncryptFile (see page 17) function

## 1.1.19 CardDecryptFileA Function

Decrypt file using Belgian Id card

**Pascal**

```
function CardDecryptFileA(Source: PAnsiChar; Destination: PAnsiChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| szSource | The name of the encrypted file |
| szDestination | The name of the decrypted file |

**Description**

Decrypt file which was encrypted using CardEncryptFile (see page 17) function

## 1.1.20 CardDecryptFileW Function

Decrypt file using Belgian Id card

**Pascal**

```
function CardDecryptFileW(Source: PWideChar; Destination: PWideChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| szSource | The name of the encrypted file |
| szDestination | The name of the decrypted file |

**Description**

Decrypt file which was encrypted using CardEncryptFile (see page 17) function

## 1.1.21 CardEncryptFile Function

Encrypt file using Belgian Id card

**Pascal**

```
function CardEncryptFile(Source: PChar; Destination: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| szSource | The name of the source file |

| | |
|---|---|
| szDestination | The name of the encrypted file |

**Description**

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

## 1.1.22 CardEncryptFileA Function

Encrypt file using Belgian Id card

**Pascal**

```
function CardEncryptFileA(Source: PAnsiChar; Destination: PAnsiChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| szSource | The name of the source file |
| szDestination | The name of the encrypted file |

**Description**

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

## 1.1.23 CardEncryptFileW Function

Encrypt file using Belgian Id card

**Pascal**

```
function CardEncryptFileW(Source: PWideChar; Destination: PWideChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| szSource | The name of the source file |
| szDestination | The name of the encrypted file |

**Description**

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

## 1.1.24 CardSignCadesT Function

Sign data with eID card according to CADES-T standard

**Pascal**

```
function CardSignCadesT(readerNumber: integer; data: PBYTE; dataLen: integer; signature:
PBYTE; var signatureLen: longword): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| readerNumber: integer | The zero-based index of the card reader. |
| data: PBYTE | the data to sign |
| dataLen: integer | the size of the data buffer |
| signature: PBYTE | the signature buffer |
| var signatureLen: longword | 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

## 1.1.25 CardSignCMS Function

Sign data with eID card according to CMS standard

**Pascal**

```
function CardSignCMS(readerNumber: integer; data: PBYTE; dataLen: integer; signature:
PBYTE; var signatureLen: longword): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| readerNumber: integer | The zero-based index of the card reader. |
| data: PBYTE | the data to sign |
| dataLen: integer | the size of the data buffer |
| signature: PBYTE | the signature buffer |
| var signatureLen: longword | the size of the signature buffer |

**Returns**

Returns true if the operation is successful, otherwise returns false

**Description**

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

## 1.1.26 CertSignCadesT Function

Sign data with certificate according to CADES-T standard

**Pascal**

```
function CertSignCadesT(certificate: PWideChar; password: PWideChar; data: PBYTE; dataLen:
integer; signature: PBYTE; var signatureLen: longword): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| certificate: PWideChar | the name of the certificate file |
| password: PWideChar | password of the certificate file |
| data: PBYTE | the data to sign |
| dataLen: integer | the size of the data buffer |
| signature: PBYTE | the signature buffer |
| var signatureLen: longword | 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

## 1.1.27 CertSignCMS Function

Sign data with certificate according to CMS standard

**Pascal**

```
function CertSignCMS(certificate: PWideChar; password: PWideChar; data: PBYTE; dataLen:
integer; signature: PBYTE; var signatureLen: longword): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| certificate: PWideChar | the name of the certificate file |
| password: PWideChar | password of the certificate file |
| data: PBYTE | the data to sign |
| dataLen: integer | the size of the data buffer |
| signature: PBYTE | the signature buffer |
| var signatureLen: longword | 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

## 1.1.28 CheckMD5 Function

Checks the MD5 hash value of the memory buffer

**Pascal**

```
function CheckMD5(Source: PBYTE; SourceSize: integer; Buffer: PBYTE; BufferSize: integer):
BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Source: PBYTE | The source bytes |
| SourceSize: integer | The size of the source buffer |
| Buffer: PBYTE | The hash value buffer |
| BufferSize: integer | 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

## 1.1.29 CheckSHA1 Function

Checks the SHA1 hash value of the memory buffer

**Pascal**

```
function CheckSHA1(Source: PBYTE; SourceSize: integer; Buffer: PBYTE; BufferSize: integer):
BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Source: PBYTE | The source bytes |
| SourceSize: integer | The size of the source buffer |
| Buffer: PBYTE | The hash value buffer |
| BufferSize: integer | 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

## 1.1.30 ClearFileAttributes Function

This function sets the file attributes to normal.

**Pascal**

```
procedure ClearFileAttributes(FileName: PChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PChar | The name of the file |

**Description**

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

## 1.1.31 ClearFileAttributesA Function

This function sets the file attributes to normal.

**Pascal**

```
procedure ClearFileAttributesA(FileName: PAnsiChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PAnsiChar | The name of the file |

**Description**

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

## 1.1.32 ClearFileAttributesW Function

This function sets the file attributes to normal.

**Pascal**

```
procedure ClearFileAttributesW(FileName: PWideChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PWideChar | The name of the file |

**Description**

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

## 1.1.33 ClearUnusedMemory Function

Clears unused memory and minimized the application memory usage

**Pascal**

```
procedure ClearUnusedMemory; stdcall;
```

**File**

SwelioEngine (see page 162)

## 1.1.34 CreateUnicodeFile Function

Creates UNICODE file

**Pascal**

```
procedure CreateUnicodeFile(const FileName: PChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| const FileName: PChar | The name of the file |

## 1.1.35 CreateUnicodeFileA Function

Creates UNICODE file

**Pascal**

```
procedure CreateUnicodeFileA(const FileName: PAnsiChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| const FileName: PAnsiChar | The name of the file |

## 1.1.36 CreateUnicodeFileW Function

Creates UNICODE file

**Pascal**

```
procedure CreateUnicodeFileW(const FileName: PWideChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| const FileName: PWideChar | The name of the file |

## 1.1.37 CurrentIPAddress Function

Returns the IP address

**Pascal**

```
procedure CurrentIPAddress(Address: PChar; Len: UINT); stdcall;
```

**File**

SwelioEngine (see page 162)

## 1.1.38 CurrentIPAddressA Function

Returns the IP address

**Pascal**

```
procedure CurrentIPAddressA(Address: PAnsiChar; Len: UINT); stdcall;
```

**File**

SwelioEngine (see page 162)

## 1.1.39 CurrentIPAddressW Function

Returns the IP address

**Pascal**

```
procedure CurrentIPAddressW(Address: PWideChar; Len: UINT); stdcall;
```

**File**

SwelioEngine (see page 162)

## 1.1.40 DeactivateCard Function

Terminates a cennection between a smart card and a reader

**Pascal**

```
procedure DeactivateCard; stdcall;
```

**File**

SwelioEngine (see page 162)

**Description**

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

## 1.1.41 DeactivateCardEx Function

Terminates a connection between a smart card and a reader

**Pascal**

```
procedure DeactivateCardEx(ReaderNumber: integer); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader |

**Description**

The DeactivateCard (see page 24) function terminates a connection previously opened between the calling application and a smart card in the target reader.

## 1.1.42 DeallocateBuffer Function

Deallocates the memory buffer

**Pascal**

```
procedure DeallocateBuffer(Buffer: Pointer); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Buffer: Pointer | The pointer to the memory buffer |

**Description**

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

## 1.1.43 DeallocateHWND Function

This function destroys the specified window.

**Pascal**

```
function DeallocateHWND(Hwnd: THandle): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Hwnd: THandle | Handle to the window to be destroyed |

**Description**

This function restores the window default procedure and destroys the window

## 1.1.44 DeallocateHWNDA Function

This function destroys the specified window.

**Pascal**

```
function DeallocateHWNDA(Hwnd: THandle): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Hwnd: THandle | Handle to the window to be destroyed |

**Description**

This function restores the window default procedure and destroys the window

## 1.1.45 DeallocateHWNDW Function

This function destroys the specified window.

**Pascal**

```
function DeallocateHWNDW(Hwnd: THandle): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Hwnd: THandle | Handle to the window to be destroyed |

**Description**

This function restores the window default procedure and destroys the window

## 1.1.46 DecryptFileAES Function

Decrypts file using AES algorithm.

**Pascal**

```
function DecryptFileAES(Source: PChar; Destination: PChar; Password: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| szSource | The source file name |
| szDestination | The decrypted file name |
| 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

## 1.1.47 DecryptFileAESA Function

Decrypts file using AES algorithm.

**Pascal**

```
function DecryptFileAESA(Source: PAnsiChar; Destination: PAnsiChar; Password: PAnsiChar):
BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| szSource | The source file name |
| szDestination | The decrypted file name |
| 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

## 1.1.48 DecryptFileAESW Function

Decrypts file using AES algorithm.

**Pascal**

```
function DecryptFileAESW(Source: PWideChar; Destination: PWideChar; Password: PWideChar):
BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| szSource | The source file name |
| szDestination | The decrypted file name |
| 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

## 1.1.49 DeleteToRecycleBin Function

Deletes file to WIndows Recycle Bin

**Pascal**

```
procedure DeleteToRecycleBin(FileName: PChar; Silent: BOOL); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PChar | The name of the file |
| Silent: BOOL | Do not display a progress dialog box |

**Description**

Use this function to delete the file to Windows Recycle Bin

## 1.1.50 DeleteToRecycleBinA Function

Deletes file to the Windows Recycle Bin

**Pascal**

```
procedure DeleteToRecycleBinA(FileName: PAnsiChar; Silent: BOOL); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PAnsiChar | The name of the file |
| Silent: BOOL | Do not display a progress dialog box |

**Description**

Use this function to delete the file to Windows Recycle Bin

## 1.1.51 DeleteToRecycleBinW Function

Deletes file to the Windows Recycle Bin

**Pascal**

```
procedure DeleteToRecycleBinW(FileName: PWideChar; Silent: BOOL); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PWideChar | The name of the file |
| Silent: BOOL | Do not display a progress dialog box |

**Description**

Use this function to delete the file to Windows Recycle Bin

## 1.1.52 DestroyImageBuffer Function

Destroys the memory buffer

**Pascal**

```
procedure DestroyImageBuffer(Buffer: PByte); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Buffer: PByte | The memory buffer |

**Description**

Destroys the memory buffer created to hold the image returned by GetPNGA (see page 72) (Ansi) or GetPNGW (see page 73) (Unicode) functions

## 1.1.53 DirectoryExists Function

Determines whether a specified directory exists.

**Pascal**

```
function DirectoryExists(FileName: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PChar | 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.

## 1.1.54 DirectoryExistsA Function

Determines whether a specified directory exists.

**Pascal**

```
function DirectoryExistsA(FileName: PAnsiChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PAnsiChar | The name of the directory |

**Returns**

Returns TRUE if the directory exists, otherwise returns FALSE

**Description**

Call DirectoryExists (see page 29) 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 (see page 29) searches for the directory along the designated path. Otherwise, the Directory parameter is interpreted as a relative path name from the current directory.

## 1.1.55 DirectoryExistsW Function

Determines whether a specified directory exists.

**Pascal**

```
function DirectoryExistsW(FileName: PWideChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PWideChar | The name of the directory |

**Returns**

Returns TRUE if the directory exists, otherwise returns FALSE

**Description**

Call DirectoryExists (see page 29) 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 (see page 29) searches for the directory along the designated path. Otherwise, the Directory parameter is interpreted as a relative path name from the current directory.

## 1.1.56 DisplayCertificate Function

Displays the dialog window with certificate information

**Pascal**

```
procedure DisplayCertificate(Certificate: PEidCertificate); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Certificate: PEidCertificate | The certificate data |

**Description**

Use this function to show the certificate for the user

## 1.1.57 DocumentTypeToString Function

Returns the textual representation of the card type (in English)

**Pascal**

```
function DocumentTypeToString(AType: integer): String;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| AType: integer | the document type (eID card, Kids card, etc...) |

**Returns**

English description of the card type.

## 1.1.58 DrawLayeredWindow Function

Repaints the surface of the layered window

**Pascal**

```
procedure DrawLayeredWindow(WindowHndle: THandle; Left: integer; Top: integer; Width:
integer; Height: integer; buffer: HDC; ColorKey: COLORREF; Alpha: byte; redrawOnly: BOOL);
stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Left: integer | The horizontal coordinate of the window |
| Top: integer | The vertical coordinate of the window |
| Width: integer | The width of the window |
| Height: integer | The height of the window |
| buffer: HDC | Handle to a DC for the surface that defines the layered window |

| | |
|---|---|
| ColorKey: COLORREF | COLORREF structure that specifies the color key to be used when composing the layered window. |
| Alpha: byte | Specifies an alpha transparency value to be used on the entire source bitmap |
| redrawOnly: BOOL | Only redraw and do not update the window position |
| handle | The handle of the window |

## 1.1.59 EmptyRecycleBin Function

Empties the recycle bin

**Pascal**

```
procedure EmptyRecycleBin; stdcall;
```

**File**

SwelioEngine (see page 162)

**Description**

Removes all files from the Windows recycle bin

## 1.1.60 EncodeCertificate Function

Performs Base64 encoding of the certificate

**Pascal**

```
function EncodeCertificate(Certificate: PEidCertificate; Buffer: PBYTE; BufferSize:
integer): integer; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Certificate: PEidCertificate | The certificate data |
| Buffer: PBYTE | The Base64 encoded certificate buffer |
| BufferSize: integer | The size of the buffer |

**Returns**

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

## 1.1.61 EncodePhoto Function

Performs Base64 encoding of the photo

**Pascal**

```
function EncodePhoto(Photo: PeidPicture; Buffer: PBYTE; BufferSize: integer): integer;
stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Photo: PeidPicture | The pointer to EidPicture structure |
| Buffer: PBYTE | The Base64 encoded photo buffer |
| BufferSize: integer | 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

## 1.1.62 EncryptFileAES Function

Encrypts file using AES algorithm.

**Pascal**

```
function EncryptFileAES(Source: PChar; Destination: PChar; Password: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| szSource | The source file name |
| szDestination | The encrypted file name |
| 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

## 1.1.63 EncryptFileAESA Function

Encrypts file using AES algorithm.

**Pascal**

```
function EncryptFileAESA(Source: PAnsiChar; Destination: PAnsiChar; Password: PAnsiChar):
BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| szSource | The source file name |
| szDestination | The encrypted file name |
| 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

## 1.1.64 EncryptFileAESW Function

Encrypts file using AES algorithm.

**Pascal**

```
function EncryptFileAESW(Source: PWideChar; Destination: PWideChar; Password: PWideChar):
BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| szSource | The source file name |
| szDestination | The encrypted file name |
| 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

## 1.1.65 FileClose Function

Concludes input/output (I/O) to a file opened using the FileCreateRewrite (see page 36) function.

**Pascal**

```
procedure FileClose(Handle: THandle); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Handle: THandle | The handle of the file |

**Description**

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

## 1.1.66 FileCloseA Function

Concludes input/output (I/O) to a file opened using the FileCreateRewrite (see page 36) function.

**Pascal**

```
procedure FileCloseA(Handle: THandle); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Handle: THandle | The handle of the file |

**Description**

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

## 1.1.67 FileCloseW Function

Concludes input/output (I/O) to a file opened using the FileCreateRewrite (see page 36) function.

**Pascal**

```
procedure FileCloseW(Handle: THandle); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Handle: THandle | The handle of the file |

**Description**

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

## 1.1.68 FileCopy Function

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

**Pascal**

```
function FileCopy(OldName: PChar; NewName: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| OldName: PChar | The name of the source file |
| NewName: PChar | 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.

## 1.1.69 FileCopyA Function

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

**Pascal**

```
function FileCopyA(OldName: PAnsiChar; NewName: PAnsiChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| OldName: PAnsiChar | The name of the source file |
| NewName: PAnsiChar | 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.

## 1.1.70 FileCopyW Function

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

**Pascal**

```
function FileCopyW(OldName: PWideChar; NewName: PWideChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| OldName: PWideChar | The name of the source file |
| NewName: PWideChar | 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.

## 1.1.71 FileCreateRewrite Function

Creates new or overwrites existing file

**Pascal**

```
function FileCreateRewrite(const FileName: PChar): THandle; stdcall;
```

**File**

SwelioEngine (see page 162)

**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

## 1.1.72 FileCreateRewriteA Function

Creates new or overwrites existing file

**Pascal**

```
function FileCreateRewriteA(const FileName: PAnsiChar): THandle; stdcall;
```

**File**

SwelioEngine (see page 162)

**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

## 1.1.73 FileCreateRewriteW Function

Creates new or overwrites existing file

**Pascal**

```
function FileCreateRewriteW(const FileName: PWideChar): THandle; stdcall;
```

**File**

SwelioEngine (see page 162)

**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

## 1.1.74 FileDelete Function

Deletes a file from disk.

**Pascal**

```
procedure FileDelete(FileName: PChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PChar | The name of the file |

**Description**

DeleteFile deletes the file named by fileName from the disk.

## 1.1.75 FileDeleteA Function

Deletes a file from disk.

**Pascal**

```
procedure FileDeleteA(FileName: PAnsiChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PAnsiChar | The name of the file |

**Description**

DeleteFile deletes the file named by fileName from the disk.

## 1.1.76 FileDeleteW Function

Deletes a file from disk.

**Pascal**

```
procedure FileDeleteW(FileName: PWideChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PWideChar | The name of the file |

**Description**

DeleteFile deletes the file named by fileName from the disk.

## 1.1.77 FileExists Function

Tests whether a specified file exists.

**Pascal**

```
function FileExists(FileName: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**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.

## 1.1.78 FileExistsA Function

Tests whether a specified file exists.

**Pascal**

```
function FileExistsA(FileName: PAnsiChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Returns**

FileExists (see page 38) returns TRUE if the file specified by FileName exists. If the file does not exist, FileExists (see page 38) returns FALSE.

**Description**

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

## 1.1.79 FileExistsW Function

Tests whether a specified file exists.

**Pascal**

```
function FileExistsW(FileName: PWideChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Returns**

FileExists (see page 38) returns TRUE if the file specified by FileName exists. If the file does not exist, FileExists (see page 38) returns FALSE.

**Description**

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

## 1.1.80 FileExtensionIs Function

Checks the file extension

**Pascal**

```
function FileExtensionIs(const FileName: PChar; Ext: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| const FileName: PChar | The name of the file |
| Ext: PChar | 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

## 1.1.81 FileExtensionIsA Function

Checks the file extension

**Pascal**

```
function FileExtensionIsA(const FileName: PAnsiChar; Ext: PAnsiChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| const FileName: PAnsiChar | The name of the file |
| Ext: PAnsiChar | 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

## 1.1.82 FileExtensionIsW Function

Checks the file extension

**Pascal**

```
function FileExtensionIsW(const FileName: PWideChar; Ext: PWideChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| const FileName: PWideChar | The name of the file |

| | |
|---|---|
| Ext: PWideChar | 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

## 1.1.83 FileGetSize Function

Retrieves the size of a specified file.

**Pascal**

```
function FileGetSize(const FileName: PChar): DWORD; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| const FileName: PChar | 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

## 1.1.84 FileGetSizeA Function

Retrieves the size of a specified file.

**Pascal**

```
function FileGetSizeA(const FileName: PAnsiChar): DWORD; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| const FileName: PAnsiChar | 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.

## 1.1.85 FileGetSizeW Function

Retrieves the size of a specified file.

**Pascal**

```
function FileGetSizeW(const FileName: PWideChar): DWORD; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| const FileName: PWideChar | 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.

## 1.1.86 FileIsExe Function

Checks if the file is a Windows executable

**Pascal**

```
function FileIsExe(FileName: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PChar | The name of the file |

**Returns**

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

## 1.1.87 FileIsExeA Function

Checks if the file is a Windows executable

**Pascal**

```
function FileIsExeA(FileName: PAnsiChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PAnsiChar | The name of the file |

**Returns**

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

## 1.1.88 FileIsExeW Function

Checks if the file is a Windows executable

**Pascal**

```
function FileIsExeW(FileName: PWideChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PWideChar | The name of the file |

**Returns**

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

## 1.1.89 FileIsIcon Function

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

**Pascal**

```
function FileIsIcon(FileName: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PChar | The name of the file |

**Returns**

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

## 1.1.90 FileIsIconA Function

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

**Pascal**

```
function FileIsIconA(FileName: PAnsiChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PAnsiChar | The name of the file |

**Returns**

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

## 1.1.91 FileIsIconW Function

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

**Pascal**

```
function FileIsIconW(FileName: PWideChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PWideChar | The name of the file |

**Returns**

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

## 1.1.92 FileIsImage Function

Checks if the file is an image file

**Pascal**

```
function FileIsImage(const FileName: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| const FileName: PChar | The name of the file |

**Returns**

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

## 1.1.93 FileIsImageA Function

Checks if the file is an image file

**Pascal**

```
function FileIsImageA(const FileName: PAnsiChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| const FileName: PAnsiChar | The name of the file |

**Returns**

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

## 1.1.94 FileIsImageW Function

Checks if the file is an image file

**Pascal**

```
function FileIsImageW(const FileName: PWideChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| const FileName: PWideChar | The name of the file |

**Returns**

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

## 1.1.95 FileOrFolderExists Function

Checks if the file or folder with the given name exists

**Pascal**

```
function FileOrFolderExists(FileName: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PChar | The file or folder name |

**Returns**

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

# 1.1.96 FileOrFolderExistsA Function

Checks if the file or folder with the given name exists

**Pascal**

```
function FileOrFolderExistsA(FileName: PAnsiChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PAnsiChar | The file or folder name |

**Returns**

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

# 1.1.97 FileOrFolderExistsW Function

Checks if the file or folder with the given name exists

**Pascal**

```
function FileOrFolderExistsW(FileName: PWideChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PWideChar | The file or folder name |

**Returns**

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

# 1.1.98 FileRename Function

Renames the file

**Pascal**

```
function FileRename(OldName: PChar; NewName: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| OldName: PChar | File to be renamed |
| NewName: PChar | New name of the file |

**Returns**

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

## 1.1.99 FileRenameA Function

Renames the file

**Pascal**

```
function FileRenameA(OldName: PAnsiChar; NewName: PAnsiChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| OldName: PAnsiChar | File to be renamed |
| NewName: PAnsiChar | New name of the file |

**Returns**

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

## 1.1.100 FileRenameW Function

Renames the file

**Pascal**

```
function FileRenameW(OldName: PWideChar; NewName: PWideChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| OldName: PWideChar | File to be renamed |
| NewName: PWideChar | New name of the file |

**Returns**

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

## 1.1.101 FileWrite Function

Writes string to the file

**Pascal**

```
procedure FileWrite(handle: THandle; Text: PChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| handle: THandle | The handle of the file |
| Text: PChar | The text to write |

## 1.1.102 FileWriteA Function

Writes string to the file

**Pascal**

```
procedure FileWriteA(Handle: THandle; Text: PAnsiChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Handle: THandle | The handle of the file |
| Text: PAnsiChar | The text to write |

## 1.1.103 FileWriteChar Function

Writes one character to the file

**Pascal**

```
procedure FileWriteChar(Handle: THandle; Text: WideChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Handle: THandle | The handle of the file |
| Text: WideChar | The character to write |

## 1.1.104 FileWriteCharA Function

Writes one character to the file

**Pascal**

```
procedure FileWriteCharA(Handle: THandle; Text: AnsiChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Handle: THandle | The handle of the file |
| Text: AnsiChar | The character to write |

# 1.1.105 FileWriteCharW Function

Writes one character to the file

**Pascal**

```
procedure FileWriteCharW(Handle: THandle; Text: WideChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Handle: THandle | The handle of the file |
| Text: WideChar | The character to write |

# 1.1.106 FileWriteNewLine Function

Writes new line sequence to the file

**Pascal**

```
procedure FileWriteNewLine(Handle: THandle); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Handle: THandle | The handle of the file |

# 1.1.107 FileWriteNewLineA Function

Writes new line sequence to the file

**Pascal**

```
procedure FileWriteNewLineA(Handle: THandle); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Handle: THandle | The handle of the file |

# 1.1.108 FileWriteNewLineW Function

Writes new line sequence to the file

**Pascal**

```
procedure FileWriteNewLineW(Handle: THandle); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Handle: THandle | The handle of the file |

## 1.1.109 FileWriteW Function

Writes string to the file

**Pascal**

```
procedure FileWriteW(handle: THandle; Text: PWideChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| handle: THandle | The handle of the file |
| Text: PWideChar | The text to write |

## 1.1.110 FormatCardNumber Function

Format card number string for better visualization

**Pascal**

```
function FormatCardNumber(Number: String): String;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Number: String | The unformatted card number |

**Returns**

Formatted string

## 1.1.111 FormatEIDDate Function

Converts the national number value to its formatted String representation

**Pascal**

```
function FormatEIDDate(D: String): String;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| D: String | The unformatted national number |

**Returns**

Formatted string

## 1.1.112 FormatNationalNumber Function

Format the national number string for better visualization

**Pascal**

```
function FormatNationalNumber(Number: String): String;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Number: String | The unformatted national number |

**Returns**

Formatted string

## 1.1.113 FullPath Function

Gets the full path to the file based on file name

**Pascal**

```
function FullPath(FileName: PChar; FullName: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PChar | The name of the file |
| FullName: PChar | The full path to the file |

## 1.1.114 FullPathA Function

Gets the full path to the file based on file name

**Pascal**

```
function FullPathA(FileName: PAnsiChar; FullName: PAnsiChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PAnsiChar | The name of the file |
| FullName: PAnsiChar | The full path to the file |

## 1.1.115 FullPathW Function

Gets the full path to the file based on file name

**Pascal**

```
function FullPathW(FileName: PWideChar; FullName: PWideChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PWideChar | The name of the file |
| FullName: PWideChar | The full path to the file |

## 1.1.116 GenerateAuthenticationSignature Function

Generate authentication signature

**Pascal**

```
function GenerateAuthenticationSignature(PinCode: PChar; DataHash: PBYTE; HashSize:
integer; Signature: PBYTE; SignatureSize: LPDWORD): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| PinCode: PChar | The card PIN code value |
| DataHash: PBYTE | The hash value buffer |
| HashSize: integer | The size of the hash data buffer |
| Signature: PBYTE | The output buffer that contains the generated signature |
| SignatureSize: LPDWORD | 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 GenerateAuthenticationSignatureA Function

Generate authentication signature

**Pascal**

```
function GenerateAuthenticationSignatureA(PinCode: PAnsiChar; DataHash: PBYTE; HashSize:
integer; Signature: PBYTE; SignatureSize: LPDWORD): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| PinCode: PAnsiChar | The card PIN code value |
| DataHash: PBYTE | The hash value buffer |
| HashSize: integer | The size of the hash data buffer |
| Signature: PBYTE | The output buffer that contains the generated signature |
| SignatureSize: LPDWORD | 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 GenerateAuthenticationSignatureEx Function

Generate authentication signature

**Pascal**

```
function GenerateAuthenticationSignatureEx(ReaderNumber: integer; PinCode: PChar; DataHash:
PBYTE; HashSize: integer; Signature: PBYTE; SignatureSize: LPDWORD): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader. |
| PinCode: PChar | The card PIN code value |
| DataHash: PBYTE | The hash value buffer |
| HashSize: integer | The size of the hash data buffer |
| Signature: PBYTE | The output buffer that contains the generated signature |
| SignatureSize: LPDWORD | 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 GenerateAuthenticationSignatureExA Function

Generate authentication signature

**Pascal**

```
function GenerateAuthenticationSignatureExA(ReaderNumber: integer; PinCode: PAnsiChar;
DataHash: PBYTE; HashSize: integer; Signature: PBYTE; SignatureSize: LPDWORD): BOOL;
stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader. |
| PinCode: PAnsiChar | The card PIN code value |
| DataHash: PBYTE | The hash value buffer |
| HashSize: integer | The size of the hash data buffer |
| Signature: PBYTE | The output buffer that contains the generated signature |
| SignatureSize: LPDWORD | 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 GenerateAuthenticationSignatureExW Function

Generate authentication signature

**Pascal**

```
function GenerateAuthenticationSignatureExW(ReaderNumber: integer; PinCode: PWideChar;
DataHash: PBYTE; HashSize: integer; Signature: PBYTE; SignatureSize: LPDWORD): BOOL;
stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader. |
| PinCode: PWideChar | The card PIN code value |
| DataHash: PBYTE | The hash value buffer |
| HashSize: integer | The size of the hash data buffer |
| Signature: PBYTE | The output buffer that contains the generated signature |
| SignatureSize: LPDWORD | 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

## 1.1.121 GenerateAuthenticationSignatureW Function

Generate authentication signature

**Pascal**

```
function GenerateAuthenticationSignatureW(PinCode: PWideChar; DataHash: PBYTE; HashSize:
integer; Signature: PBYTE; SignatureSize: LPDWORD): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| PinCode: PWideChar | The card PIN code value |
| DataHash: PBYTE | The hash value buffer |
| HashSize: integer | The size of the hash data buffer |
| Signature: PBYTE | The output buffer that contains the generated signature |
| SignatureSize: LPDWORD | 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

## 1.1.122 GenerateBMP Function

Generates Windows Bitmap file with QR Code image

**Pascal**

```
procedure GenerateBMP(FileName: PChar; Text: PChar; Margin: integer; Size: integer; Level:
integer); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PChar | The name of the file |
| Text: PChar | The text to encode |
| Margin: integer | The margin from the border in points |
| Size: integer | The size of the one point in pixels |
| Level: integer | The error correction level |

**Description**

Generate Windows Bitmap file with encoded text as QR Code image

# 1.1.123 GenerateBMPA Function

Generates Windows Bitmap file with QR Code image

**Pascal**

```
procedure GenerateBMPA(FileName: PAnsiChar; Text: PAnsiChar; Margin: integer; Size:
integer; Level: integer); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PAnsiChar | The name of the file |
| Text: PAnsiChar | The text to encode |
| Margin: integer | The margin from the border in points |
| Size: integer | The size of the one point in pixels |
| Level: integer | The error correction level |

**Description**

Generate Windows Bitmap file with encoded text as QR Code image

# 1.1.124 GenerateBMPW Function

Generates Windows Bitmap file with QR Code image

**Pascal**

```
procedure GenerateBMPW(FileName: PWideChar; Text: PWideChar; Margin: integer; Size:
integer; Level: integer); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PWideChar | The name of the file |
| Text: PWideChar | The text to encode |
| Margin: integer | The margin from the border in points |
| Size: integer | The size of the one point in pixels |
| Level: integer | The error correction level |

**Description**

Generate Windows Bitmap file with encoded text as QR Code image

# 1.1.125 GenerateNonRepudiationSignature Function

Generate non repudiation signature

**Pascal**

```
function GenerateNonRepudiationSignature(PinCode: PChar; DataHash: PBYTE; HashSize:
integer; Signature: PBYTE; SignatureSize: LPDWORD): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| PinCode: PChar | The card PIN code value |
| DataHash: PBYTE | The hash value buffer |
| HashSize: integer | The size of the hash data buffer |
| Signature: PBYTE | The output buffer that contains the generated signature |
| SignatureSize: LPDWORD | 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 GenerateNonRepudiationSignatureA Function

Generate non repudiation signature

**Pascal**

```
function GenerateNonRepudiationSignatureA(PinCode: PAnsiChar; DataHash: PBYTE; HashSize:
integer; Signature: PBYTE; SignatureSize: LPDWORD): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| PinCode: PAnsiChar | The card PIN code value |
| DataHash: PBYTE | The hash value buffer |
| HashSize: integer | The size of the hash data buffer |
| Signature: PBYTE | The output buffer that contains the generated signature |
| SignatureSize: LPDWORD | 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

## 1.1.127 GenerateNonRepudiationSignatureEx Function

Generate non repudiation signature

**Pascal**

```
function GenerateNonRepudiationSignatureEx(ReaderNumber: integer; PinCode: PChar; DataHash:
PBYTE; HashSize: integer; Signature: PBYTE; SignatureSize: LPDWORD): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader. |
| PinCode: PChar | The card PIN code value |
| DataHash: PBYTE | The hash value buffer |
| HashSize: integer | The size of the hash data buffer |
| Signature: PBYTE | The output buffer that contains the generated signature |
| SignatureSize: LPDWORD | 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

## 1.1.128 GenerateNonRepudiationSignatureExA Function

Generate non repudiation signature

**Pascal**

```
function GenerateNonRepudiationSignatureExA(ReaderNumber: integer; PinCode: PAnsiChar;
DataHash: PBYTE; HashSize: integer; Signature: PBYTE; SignatureSize: LPDWORD): BOOL;
stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader. |
| PinCode: PAnsiChar | The card PIN code value |
| DataHash: PBYTE | The hash value buffer |
| HashSize: integer | The size of the hash data buffer |
| Signature: PBYTE | The output buffer that contains the generated signature |
| SignatureSize: LPDWORD | 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

# 1.1.129 GenerateNonRepudiationSignatureExW Function

Generate non repudiation signature

**Pascal**

```
function GenerateNonRepudiationSignatureExW(ReaderNumber: integer; PinCode: PWideChar;
DataHash: PBYTE; HashSize: integer; Signature: PBYTE; SignatureSize: LPDWORD): BOOL;
stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader. |
| PinCode: PWideChar | The card PIN code value |
| DataHash: PBYTE | The hash value buffer |
| HashSize: integer | The size of the hash data buffer |
| Signature: PBYTE | The output buffer that contains the generated signature |
| SignatureSize: LPDWORD | 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

# 1.1.130 GenerateNonRepudiationSignatureW Function

Generate non repudiation signature

**Pascal**

```
function GenerateNonRepudiationSignatureW(PinCode: PWideChar; DataHash: PBYTE; HashSize:
integer; Signature: PBYTE; SignatureSize: LPDWORD): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| PinCode: PWideChar | The card PIN code value |
| DataHash: PBYTE | The hash value buffer |
| HashSize: integer | The size of the hash data buffer |
| Signature: PBYTE | The output buffer that contains the generated signature |
| SignatureSize: LPDWORD | 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

# 1.1.131 GeneratePNG Function

Generates PNG file with QR Code image

**Pascal**

```
procedure GeneratePNG(FileName: PChar; Text: PChar; Margin: integer; Size: integer; Level:
integer); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PChar | The name of the file |
| Text: PChar | The text to encode |
| Margin: integer | The margin from the border in points |
| Size: integer | The size of the one point in pixels |
| Level: integer | The error correction level |

**Description**

Generates PNG file with encoded text as QR Code image

# 1.1.132 GeneratePNGA Function

Generates PNG file with QR Code image

**Pascal**

```
procedure GeneratePNGA(FileName: PAnsiChar; Text: PAnsiChar; Margin: integer; Size:
integer; Level: integer); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PAnsiChar | The name of the file |
| Text: PAnsiChar | The text to encode |
| Margin: integer | The margin from the border in points |
| Size: integer | The size of the one point in pixels |
| Level: integer | The error correction level |

**Description**

Generates PNG file with encoded text as QR Code image

# 1.1.133 GeneratePNGW Function

Generates PNG file with QR Code image

**Pascal**

```
procedure GeneratePNGW(FileName: PWideChar; Text: PWideChar; Margin: integer; Size:
integer; Level: integer); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PWideChar | The name of the file |
| Text: PWideChar | The text to encode |
| Margin: integer | The margin from the border in points |
| Size: integer | The size of the one point in pixels |
| Level: integer | The error correction level |

**Description**

Generates PNG file with encoded text as QR Code image

## 1.1.134 GenerateQRCode Function

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

**Pascal**

```
function GenerateQRCode(FileName: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PChar | 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

## 1.1.135 GenerateQRCodeA Function

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

**Pascal**

```
function GenerateQRCodeA(FileName: PAnsiChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PAnsiChar | 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

## 1.1.136 GenerateQRCodeEx Function

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

**Pascal**

```
function GenerateQRCodeEx(ReaderNumber: integer; FileName: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader. |
| FileName: PChar | 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

## 1.1.137 GenerateQRCodeExA Function

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

**Pascal**

```
function GenerateQRCodeExA(ReaderNumber: integer; FileName: PAnsiChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader. |
| FileName: PAnsiChar | 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

# 1.1.138 GenerateQRCodeExW Function

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

**Pascal**

```
function GenerateQRCodeExW(ReaderNumber: integer; FileName: PWideChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader. |
| FileName: PWideChar | 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

# 1.1.139 GenerateQRCodeW Function

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

**Pascal**

```
function GenerateQRCodeW(FileName: PWideChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PWideChar | 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

# 1.1.140 GetCardSerialNumber Function

Gets the card serial number

**Pascal**

```
function GetCardSerialNumber(readerNumber: integer; serialNumber: PBYTE; var
serialNumberSize: DWORD): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| readerNumber: integer | the index of the card reader |
| serialNumber: PBYTE | the buffer to get the serial number value |
| var serialNumberSize: DWORD | the size of the buffer |

**Description**

Use this function to read the serial number of the card

## 1.1.141 GetEncodedCertificateSize Function

Returns the size of the Base64 encoded certificate

**Pascal**

```
function GetEncodedCertificateSize(Certificate: PEidCertificate): integer; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Certificate: PEidCertificate | 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

## 1.1.142 GetEncodedPhotoSize Function

Calculates buffer size for Base64 encoded photo

**Pascal**

```
function GetEncodedPhotoSize(photo: PeidPicture): integer; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| photo: PeidPicture | The pointer to EidPicture 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

## 1.1.143 GetFileMD5 Function

Gets the MD5 hash value for the file

**Pascal**

```
function GetFileMD5(FileName: PChar; Buffer: PBYTE; BufferSize: integer): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PChar | The name of the file |
| Buffer: PBYTE | The buffer to store the hash value |
| BufferSize: integer | 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

## 1.1.144 GetFileMD5A Function

Gets the MD5 hash value for the file

**Pascal**

```
function GetFileMD5A(FileName: PAnsiChar; Buffer: PBYTE; BufferSize: integer): BOOL;
stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PAnsiChar | The name of the file |
| Buffer: PBYTE | The buffer to store the hash value |
| BufferSize: integer | 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

## 1.1.145 GetFileMD5W Function

Gets the MD5 hash value for the file

**Pascal**

```
function GetFileMD5W(FileName: PWideChar; Buffer: PBYTE; BufferSize: integer): BOOL;
stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PWideChar | The name of the file |
| Buffer: PBYTE | The buffer to store the hash value |
| BufferSize: integer | 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

## 1.1.146 GetFilesCount Function

Calculates the number of files in the given folder

**Pascal**

```
function GetFilesCount(FolderName: PChar): integer; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FolderName: PChar | The name of the folder |

**Returns**

The number of files in the given folder

## 1.1.147 GetFilesCountA Function

Calculates the number of files in the given folder

**Pascal**

```
function GetFilesCountA(FolderName: PAnsiChar): integer; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FolderName: PAnsiChar | The name of the folder |

**Returns**

The number of files in the given folder

## 1.1.148 GetFilesCountW Function

Calculates the number of files in the given folder

**Pascal**

```
function GetFilesCountW(FolderName: PWideChar): integer; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FolderName: PWideChar | The name of the folder |

**Returns**

The number of files in the given folder

## 1.1.149 GetFileSHA1 Function

Gets the SHA1 hash value for the file

**Pascal**

```
function GetFileSHA1(FileName: PChar; Buffer: PBYTE; BufferSize: integer): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PChar | The name of the file |
| Buffer: PBYTE | The buffer to store the hash value |
| BufferSize: integer | 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

## 1.1.150 GetFileSHA1A Function

Gets the SHA1 hash value for the file

**Pascal**

```
function GetFileSHA1A(FileName: PAnsiChar; Buffer: PBYTE; BufferSize: integer): BOOL;
stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PAnsiChar | The name of the file |
| Buffer: PBYTE | The buffer to store the hash value |
| BufferSize: integer | 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

## 1.1.151 GetFileSHA1W Function

Gets the SHA1 hash value for the file

**Pascal**

```
function GetFileSHA1W(FileName: PWideChar; Buffer: PBYTE; BufferSize: integer): BOOL;
stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PWideChar | The name of the file |
| Buffer: PBYTE | The buffer to store the hash value |
| BufferSize: integer | 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

## 1.1.152 GetHBitmap Function

Generates Windows Bitmap in memory with QR Code image

**Pascal**

```
function GetHBitmap(Text: PChar; Margin: integer; Size: integer; Level: integer): HBITMAP;
stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Text: PChar | The text to encode |
| Margin: integer | The margin from the border in points |
| Size: integer | The size of the one point in pixels |

| | |
|---|---|
| Level: integer | 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

## 1.1.153 GetHBitmapA Function

Generates Windows Bitmap in memory with QR Code image

**Pascal**

```
function GetHBitmapA(Text: PAnsiChar; Margin: integer; Size: integer; Level: integer):
HBITMAP; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Text: PAnsiChar | The text to encode |
| Margin: integer | The margin from the border in points |
| Size: integer | The size of the one point in pixels |
| Level: integer | The error correction level |

**Returns**

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

**Description**

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

## 1.1.154 GetHBitmapW Function

Generates Windows Bitmap in memory with QR Code image

**Pascal**

```
function GetHBitmapW(Text: PWideChar; Margin: integer; Size: integer; Level: integer):
HBITMAP; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Text: PWideChar | The text to encode |
| Margin: integer | The margin from the border in points |
| Size: integer | The size of the one point in pixels |
| Level: integer | The error correction level |

**Returns**

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

**Description**

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

## 1.1.155 GetISOCode Function

Returns the country ISO code based on the nationality string

**Pascal**

```
function GetISOCode(Nationality: PChar; IsoCode: PChar; BufferSize: integer): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Nationality: PChar | The nationality string |
| BufferSize: integer | The size if the memory buffer |
| iso | The ISO code memory buffer |

**Returns**

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

**Description**

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

## 1.1.156 GetISOCodeA Function

Returns the country ISO code based on the nationality string

**Pascal**

```
function GetISOCodeA(Nationality: PAnsiChar; IsoCode: PAnsiChar; BufferSize: integer):
BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Nationality: PAnsiChar | The nationality string |
| BufferSize: integer | The size if the memory buffer |
| iso | The ISO code memory buffer |

**Returns**

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

**Description**

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

# 1.1.157 GetISOCodeW Function

Returns the country ISO code based on the nationality string

**Pascal**

```
function GetISOCodeW(Nationality: PWideChar; IsoCode: PWideChar; BufferSize: integer):
BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Nationality: PWideChar | The nationality string |
| BufferSize: integer | The size if the memory buffer |
| iso | The ISO code 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

# 1.1.158 GetMD5 Function

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

**Pascal**

```
function GetMD5(Source: PBYTE; SourceSize: integer; Buffer: PBYTE; BufferSize: integer):
BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Source: PBYTE | The source memory block |
| SourceSize: integer | The size of the source memory block |
| Buffer: PBYTE | The buffer for the hash value |
| BufferSize: integer | 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

# 1.1.159 GetPNG Function

Writes PNG image to the memory buffer.

**Pascal**

```
procedure GetPNG(Text: PChar; Margin: integer; Size: integer; Level: integer; var BufSize:
integer; out ppvBits: PByte); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Text: PChar | The text to encode |
| Margin: integer | The margin from the image border in points |
| Size: integer | The size of the point in pixels |
| Level: integer | The error correction level |
| var BufSize: integer | The size of the output buffer |
| out ppvBits: PByte | The buffer when the resulting image is stored |

**Description**

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

# 1.1.160 GetPNGA Function

Writes PNG image to the memory buffer.

**Pascal**

```
procedure GetPNGA(Text: PAnsiChar; Margin: integer; Size: integer; Level: integer; var
BufSize: integer; out ppvBits: PByte); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Text: PAnsiChar | The text to encode |
| Margin: integer | The margin from the image border in points |
| Size: integer | The size of the point in pixels |
| Level: integer | The error correction level |
| var BufSize: integer | The size of the output buffer |
| out ppvBits: PByte | The buffer when the resulting image is stored |

**Description**

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

## 1.1.161 GetPNGW Function

Writes PNG image to the memory buffer.

**Pascal**

```
procedure GetPNGW(Text: PWideChar; Margin: integer; Size: integer; Level: integer; var
BufSize: integer; out ppvBits: PByte); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Text: PWideChar | The text to encode |
| Margin: integer | The margin from the image border in points |
| Size: integer | The size of the point in pixels |
| Level: integer | The error correction level |
| var BufSize: integer | The size of the output buffer |
| out ppvBits: PByte | The buffer when the resulting image is stored |

**Description**

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

## 1.1.162 GetReaderIndex Function

Returns the zero-based reader index with specified name

**Pascal**

```
function GetReaderIndex(ReaderName: PChar): integer; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderName: PChar | 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

## 1.1.163 GetReaderIndexA Function

Returns the zero-based reader index with specified name

**Pascal**

```
function GetReaderIndexA(ReaderName: PAnsiChar): integer; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderName: PAnsiChar | 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

## 1.1.164 GetReaderIndexW Function

Returns the zero-based reader index with specified name

**Pascal**

```
function GetReaderIndexW(ReaderName: PWideChar): integer; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderName: PWideChar | 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

## 1.1.165 GetReaderName Function

Returns the name of the reader

**Pascal**

```
function GetReaderName(ReaderNumber: integer; StrDest: PChar; Count: integer): integer;
stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | index of the reader |
| StrDest: PChar | Destination string |
| Count: integer | Number of characters to be copied |

# 1.1.166 GetReaderNameA Function

Returns the name of the reader

**Pascal**

```
function GetReaderNameA(ReaderNumber: integer; StrDest: PAnsiChar; Count: integer):
integer; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | index of the reader |
| StrDest: PAnsiChar | Destination string |
| Count: integer | Number of characters to be copied |

# 1.1.167 GetReaderNameLen Function

Returns the length of the reader name

**Pascal**

```
function GetReaderNameLen(ReaderNumber: integer): integer; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader |

**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

# 1.1.168 GetReaderNameLenA Function

Returns the length of the reader name

**Pascal**

```
function GetReaderNameLenA(ReaderNumber: integer): integer; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader |

**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

## 1.1.169 GetReaderNameLenW Function

Returns the length of the reader name

**Pascal**

```
function GetReaderNameLenW(ReaderNumber: integer): integer; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader |

**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

## 1.1.170 GetReaderNameW Function

Returns the name of the reader

**Pascal**

```
function GetReaderNameW(ReaderNumber: integer; StrDest: PWideChar; Count: integer):
integer; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | index of the reader |
| StrDest: PWideChar | Destination string |
| Count: integer | Number of characters to be copied |

## 1.1.171 GetReadersCount Function

Get number of card readers connected to PC

**Pascal**

```
function GetReadersCount: integer; stdcall;
```

**File**

SwelioEngine (see page 162)

**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

## 1.1.172 GetSelectedReaderIndex Function

Returns the index of the active smart card reader

**Pascal**

```
function GetSelectedReaderIndex: integer; stdcall;
```

**File**

SwelioEngine (see page 162)

**Returns**

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

## 1.1.173 GetSHA1 Function

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

**Pascal**

```
function GetSHA1(Source: PBYTE; SourceSize: integer; Buffer: PBYTE; BufferSize: integer):
BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Source: PBYTE | The source memory block |
| SourceSize: integer | The size of the source memory block |
| Buffer: PBYTE | The buffer for the hash value |
| BufferSize: integer | 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

## 1.1.174 GetStartup Function

Checks if the application is registered to run when Windows starts

**Pascal**

```
function GetStartup(const AppName: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| const AppName: PChar | The name of the application |

## 1.1.175 GetStartupA Function

Checks if the application is registered to run when Windows starts

**Pascal**

```
function GetStartupA(const AppName: PAnsiChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| const AppName: PAnsiChar | The name of the application |

## 1.1.176 GetStartupW Function

Checks if the application is registered to run when Windows starts

**Pascal**

```
function GetStartupW(const AppName: PWideChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| const AppName: PWideChar | The name of the application |

## 1.1.177 GetSupportSIS Function

Checks if the SIS cards are supported by the engine

**Pascal**

```
function GetSupportSIS: BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**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 133) function. Use GetSupportSIS function to check if the SIS card support is activated.

## 1.1.178 HibernateWindows Function

Hibernates Windows

**Pascal**

```
function HibernateWindows: BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

## 1.1.179 IsAnimatedGIF Function

Checks if the file is an animated GIF image file

**Pascal**

```
function IsAnimatedGIF(FileName: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PChar | The name of the file |

**Returns**

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

## 1.1.180 IsAnimatedGIFA Function

Checks if the file is an animated GIF image file

**Pascal**

```
function IsAnimatedGIFA(FileName: PAnsiChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PAnsiChar | The name of the file |

**Returns**

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

## 1.1.181 IsAnimatedGIFW Function

Checks if the file is an animated GIF image file

**Pascal**

```
function IsAnimatedGIFW(FileName: PWideChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PWideChar | The name of the file |

**Returns**

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

## 1.1.182 IsCardPresent Function

Checks if the card is present in the card reader

**Pascal**

```
function IsCardPresent: BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**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

## 1.1.183 IsCardPresentEx Function

Checks if the card is present in the card reader

**Pascal**

```
function IsCardPresentEx(ReaderNumber: integer): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | 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

## 1.1.184 IsConnectedToInternet Function

Checks if PC is connected to Internet

**Pascal**

```
function IsConnectedToInternet: BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

## 1.1.185 IsDirectory Function

Verifies that a path is a valid directory.

**Pascal**

```
function IsDirectory(FolderName: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**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 IsDirectoryA Function

Verifies that a path is a valid directory.

**Pascal**

```
function IsDirectoryA(FolderName: PAnsiChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**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 IsDirectoryW Function

Verifies that a path is a valid directory.

**Pascal**

```
function IsDirectoryW(FolderName: PWideChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**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

## 1.1.188 IsEIDCard Function

Check if Belgian EID card is inserted into card reader

**Pascal**

```
function IsEIDCard: BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**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 IsEIDCardEx Function

Check if Belgian EID card is inserted into card reader

**Pascal**

```
function IsEIDCardEx(ReaderNumber: integer): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | 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.

## 1.1.190 IsEngineActive Function

Checks if the Swelio Engine is activated

**Pascal**

```
function IsEngineActive: BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**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 135) function.

## 1.1.191 IsFemale Function

Checks if the card owner is female

**Pascal**

```
function IsFemale(Identity: PEidIdentity): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Identity: PEidIdentity | 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 IsFemaleA Function

Checks if the card owner is female

**Pascal**

```
function IsFemaleA(Identity: PEidIdentityA): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Identity: PEidIdentityA | 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

## 1.1.193 IsFemaleW Function

Checks if the card owner is female

**Pascal**

```
function IsFemaleW(Identity: PEidIdentityW): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Identity: PEidIdentityW | 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

## 1.1.194 IsMale Function

Checks if the card owner is male

**Pascal**

```
function IsMale(Identity: PEidIdentity): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Identity: PEidIdentity | 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

# 1.1.195 IsMaleA Function

Checks if the card owner is male

**Pascal**

```
function IsMaleA(Identity: PEidIdentityA): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Identity: PEidIdentityA | 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

# 1.1.196 IsMaleW Function

Checks if the card owner is male

**Pascal**

```
function IsMaleW(Identity: PEidIdentityW): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Identity: PEidIdentityW | 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

# 1.1.197 IsMediaCenter Function

Checks if the Media Center version of Windows is installed

**Pascal**

```
function IsMediaCenter: BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

## 1.1.198 IsMetroActive Function

Checks if metro interface is active

**Pascal**

```
function IsMetroActive: BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

## 1.1.199 IsMultiTouchReady Function

Checks if the system is multi touch ready

**Pascal**

```
function IsMultiTouchReady: BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

## 1.1.200 IsNativeWin64 Function

Checks if the application is native 64 bit executable

**Pascal**

```
function IsNativeWin64: BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

## 1.1.201 IsSISCard Function

Check if Belgian SIS card is inserted into card reader

**Pascal**

```
function IsSISCard: BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**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.

# 1.1.202 IsSISCardEx Function

Check if Belgian SIS card is inserted into card reader

**Pascal**

```
function IsSISCardEx(ReaderNumber: integer): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | 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.

# 1.1.203 IsTabletPC Function

Checks if the application is running on the Tablet PC

**Pascal**

```
function IsTabletPC: BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

# 1.1.204 IsUnicodeFile Function

Checks if the file is UNICODE file

**Pascal**

```
function IsUnicodeFile(const FileName: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| const FileName: PChar | 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).

# 1.1.205 IsUnicodeFileA Function

Checks if the file is UNICODE file

**Pascal**

```
function IsUnicodeFileA(const FileName: PAnsiChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| const FileName: PAnsiChar | 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).

# 1.1.206 IsUnicodeFileW Function

Checks if the file is UNICODE file

**Pascal**

```
function IsUnicodeFileW(const FileName: PWideChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| const FileName: PWideChar | 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).

# 1.1.207 IsValidFileName Function

Checks if provided string is a valid file name

**Pascal**

```
function IsValidFileName(FileName: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PChar | 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

## 1.1.208 IsValidFileNameA Function

Checks if provided string is a valid file name

**Pascal**

```
function IsValidFileNameA(FileName: PAnsiChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PAnsiChar | 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

## 1.1.209 IsValidFileNameW Function

Checks if provided string is a valid file name

**Pascal**

```
function IsValidFileNameW(FileName: PWideChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PWideChar | 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

# 1.1.210 IsValidPathName Function

Checks if provided string is a valid file path

**Pascal**

```
function IsValidPathName(FileName: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PChar | 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

# 1.1.211 IsValidPathNameA Function

Checks if provided string is a valid file path

**Pascal**

```
function IsValidPathNameA(FileName: PAnsiChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PAnsiChar | 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

# 1.1.212 IsValidPathNameW Function

Checks if provided string is a valid file path

**Pascal**

```
function IsValidPathNameW(FileName: PWideChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PWideChar | 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

## 1.1.213 IsWindows7 Function

Checks if PC is running Windows 7 or better

**Pascal**

```
function IsWindows7: BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

## 1.1.214 IsWindows8 Function

Checks if PC is Running Windows 8 or better

**Pascal**

```
function IsWindows8: BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

## 1.1.215 IsWindowsVista Function

Checks if PC is running Windows Vista or better

**Pascal**

```
function IsWindowsVista: BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

## 1.1.216 IsWindowsXP Function

Checks if PC is running Windows XP

**Pascal**

```
function IsWindowsXP: BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

## 1.1.217 IsWindowsXPSP2 Function

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

**Pascal**

```
function IsWindowsXPSP2: BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

## 1.1.218 IsWow64 Function

Checks if the 32 bit application runs on 64 bit Windows

**Pascal**

```
function IsWow64: BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

## 1.1.219 LayeredWndProc Function

The default window procedure for the layered window

**Pascal**

```
function LayeredWndProc(hWnd: THandle; Message: UINT; wParam: WPARAM; lParam: LPARAM):
LRESULT; stdcall;
```

**File**

SwelioEngine (see page 162)

## 1.1.220 LayeredWndProcA Function

The default window procedure for the layered window

**Pascal**

```
function LayeredWndProcA(hWnd: THandle; Message: UINT; wParam: WPARAM; lParam: LPARAM):
LRESULT; stdcall;
```

**File**

SwelioEngine (see page 162)

## 1.1.221 LayeredWndProcW Function

The default window procedure for the layered window

**Pascal**

```
function LayeredWndProcW(hWnd: THandle; Message: UINT; wParam: WPARAM; lParam: LPARAM):
LRESULT; stdcall;
```

**File**

SwelioEngine (see page 162)

## 1.1.222 LoadCertificate Function

Reads the certificate from a file

**Pascal**

```
procedure LoadCertificate(FileName: PChar; Certificate: PEidCertificate); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PChar | The source file name |
| Certificate: PEidCertificate | The pointer to EidCertificate structure |

**Description**

Use this function to read the certificate from the file

## 1.1.223 LoadCertificateA Function

Reads the certificate from a file

**Pascal**

```
procedure LoadCertificateA(FileName: PAnsiChar; Certificate: PEidCertificate); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PAnsiChar | The source file name |
| Certificate: PEidCertificate | The pointer to EidCertificate structure |

**Description**

Use this function to read the certificate from the file

# 1.1.224 LoadCertificateW Function

Reads the certificate from a file

**Pascal**

```
procedure LoadCertificateW(FileName: PWideChar; Certificate: PEidCertificate); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PWideChar | The source file name |
| Certificate: PEidCertificate | The pointer to EidCertificate structure |

**Description**

Use this function to read the certificate from the file

# 1.1.225 LoadIdentity Function

Reads the raw identity information from a file

**Pascal**

```
procedure LoadIdentity(FileName: PChar; identity: PEidIdentity); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PChar | The name of the source file |
| identity: PEidIdentity | The pointer to EidIdentityW structure |

**Description**

Use this function to read back the identity information stored to the file using SaveIdentityW (see page 118) function

# 1.1.226 LoadIdentityA Function

Reads the raw identity information from a file

**Pascal**

```
procedure LoadIdentityA(FileName: PAnsiChar; identity: PEidIdentityA); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PAnsiChar | The name of the source file |

| | |
|---|---|
| identity: PEidIdentityA | The pointer to EidIdentityW structure |

**Description**

Use this function to read back the identity information stored to the file using SaveIdentityW (see page 118) function

## 1.1.227 LoadIdentityW Function

Reads the raw identity information from a file

**Pascal**

```
procedure LoadIdentityW(FileName: PWideChar; identity: PEidIdentityW); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PWideChar | The name of the source file |
| identity: PEidIdentityW | The pointer to EidIdentityW structure |

**Description**

Use this function to read back the identity information stored to the file using SaveIdentityW (see page 118) function

## 1.1.228 LoadPhoto Function

Loads photo from a file

**Pascal**

```
procedure LoadPhoto(Photo: PeidPicture; FileName: PChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Photo: PeidPicture | The pointer to EidPicture structure |
| FileName: PChar | Destination file name |

**Description**

Loads raw picture data from a file

## 1.1.229 LoadPhotoA Function

Loads photo from a file

**Pascal**

```
procedure LoadPhotoA(Photo: PeidPicture; FileName: PAnsiChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Photo: PeidPicture | The pointer to EidPicture structure |
| FileName: PAnsiChar | Destination file name |

**Description**

Loads raw picture data from a file

## 1.1.230 LoadPhotoW Function

Loads photo from a file

**Pascal**

```
procedure LoadPhotoW(Photo: PeidPicture; FileName: PWideChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Photo: PeidPicture | The pointer to EidPicture structure |
| FileName: PWideChar | Destination file name |

**Description**

Loads raw picture data from a file

## 1.1.231 MakeSoundFromFile Function

Plays the wave sound from the file

**Pascal**

```
procedure MakeSoundFromFile(const SoundName: PChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| const SoundName: PChar | The name of the file |

**Description**

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

## 1.1.232 MakeSoundFromFileA Function

Plays the wave sound from the file

**Pascal**

```
procedure MakeSoundFromFileA(const SoundName: PAnsiChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| const SoundName: PAnsiChar | The name of the file |

**Description**

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

## 1.1.233 MakeSoundFromFileW Function

Plays the wave sound from the file

**Pascal**

```
procedure MakeSoundFromFileW(const SoundName: PWideChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| const SoundName: PWideChar | The name of the file |

**Description**

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

## 1.1.234 MakeSoundFromResource Function

Plays the wave sound from the resource

**Pascal**

```
procedure MakeSoundFromResource(ModuleHandle: THandle; const SoundName: PChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| const SoundName: PChar | A string that specifies the sound to play. |
| hModule | Handle to the executable file that contains the resource to be loaded. |

**Description**

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

## 1.1.235 MakeSoundFromResourceA Function

Plays the wave sound from the resource

**Pascal**

```
procedure MakeSoundFromResourceA(ModuleHandle: THandle; const SoundName: PAnsiChar);
stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| const SoundName: PAnsiChar | A string that specifies the sound to play. |
| hModule | Handle to the executable file that contains the resource to be loaded. |

**Description**

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

## 1.1.236 MakeSoundFromResourceW Function

Plays the wave sound from the resource

**Pascal**

```
procedure MakeSoundFromResourceW(ModuleHandle: THandle; const SoundName: PWideChar);
stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| const SoundName: PWideChar | A string that specifies the sound to play. |
| hModule | Handle to the executable file that contains the resource to be loaded. |

**Description**

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

## 1.1.237 PortAvailable Function

Checks if the port with specified number is available

**Pascal**

```
function PortAvailable(PortNumber: integer): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

## 1.1.238 ReadAddress Function

Read address information from Belgian eID card

**Pascal**

```
function ReadAddress(address: PEidAddress): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| address: PEidAddress | the pointer to the address information structure |

**Returns**

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

## 1.1.239 ReadAddressA Function

Read address information from Belgian eID card

**Pascal**

```
function ReadAddressA(address: PEidAddressA): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| address: PEidAddressA | the pointer to the address information structure |

**Returns**

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

## 1.1.240 ReadAddressEx Function

Read address information from Belgian eID card

**Pascal**

```
function ReadAddressEx(ReaderNumber: integer; address: PEidAddress): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader. |
| address: PEidAddress | The pointer to the address information structure |

**Returns**

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

# 1.1.241 ReadAddressExA Function

Read address information from Belgian eID card

**Pascal**

```
function ReadAddressExA(ReaderNumber: integer; address: PEidAddressA): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader. |
| address: PEidAddressA | The pointer to the address information structure |

**Returns**

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

# 1.1.242 ReadAddressExW Function

Read address information from Belgian eID card

**Pascal**

```
function ReadAddressExW(ReaderNumber: integer; address: PEidAddressW): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader. |
| address: PEidAddressW | The pointer to the address information structure |

**Returns**

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

# 1.1.243 ReadAddressW Function

Read address information from Belgian eID card

**Pascal**

```
function ReadAddressW(address: PEidAddressW): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| address: PEidAddressW | the pointer to the address information structure |

**Returns**

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

## 1.1.244 ReadAuthenticationCertificate Function

Read Authentication Certificate to memory

**Pascal**

```
function ReadAuthenticationCertificate(Certificate: PEidCertificate): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| cerificate | The pointer to EidCertificate structure |

**Returns**

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

**Description**

Read Authentication Certificate from the card to EidCertificate structure

## 1.1.245 ReadBufferFromFile Function

Reads the content of the file to the memory buffer

**Pascal**

```
procedure ReadBufferFromFile(FileName: PChar; Buffer: PBYTE; BufferSize: integer); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PChar | The name of the file |
| Buffer: PBYTE | The address of the memory block |
| BufferSize: integer | The size of the memory block |

**Description**

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

## 1.1.246 ReadBufferFromFileA Function

Reads the content of the file to the memory buffer

**Pascal**

```
procedure ReadBufferFromFileA(FileName: PAnsiChar; Buffer: PBYTE; BufferSize: integer);
stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PAnsiChar | The name of the file |
| Buffer: PBYTE | The address of the memory block |
| BufferSize: integer | The size of the memory block |

**Description**

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

## 1.1.247 ReadBufferFromFileW Function

Reads the content of the file to the memory buffer

**Pascal**

```
procedure ReadBufferFromFileW(FileName: PWideChar; Buffer: PBYTE; BufferSize: integer);
stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PWideChar | The name of the file |
| Buffer: PBYTE | The address of the memory block |
| BufferSize: integer | The size of the memory block |

**Description**

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

## 1.1.248 ReadCaCertificate Function

Read Ca Certificate to memory

**Pascal**

```
function ReadCaCertificate(Certificate: PEidCertificate): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| cerificate | The pointer to EidCertificate structure |

**Returns**

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

**Description**

Read Ca Certificate to EidCertificate structure

# 1.1.249 ReadIdentity Function

Read identity information from Belgian eID card

**Pascal**

```
function ReadIdentity(identity: PEidIdentity): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| identity: PEidIdentity | The pointer to the identity information structure |

**Returns**

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

# 1.1.250 ReadIdentityA Function

Read identity information from Belgian eID card

**Pascal**

```
function ReadIdentityA(identity: PEidIdentityA): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| identity: PEidIdentityA | The pointer to the identity information structure |

**Returns**

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

# 1.1.251 ReadIdentityEx Function

Read identity information from Belgian eID card

**Pascal**

```
function ReadIdentityEx(ReaderNumber: integer; identity: PEidIdentity): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader. |
| identity: PEidIdentity | The pointer to the identity information structure |

**Returns**

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

## 1.1.252 ReadIdentityExA Function

Read identity information from Belgian eID card

**Pascal**

```
function ReadIdentityExA(ReaderNumber: integer; identity: PEidIdentityA): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader. |
| identity: PEidIdentityA | The pointer to the identity information structure |

**Returns**

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

## 1.1.253 ReadIdentityExW Function

Read identity information from Belgian eID card

**Pascal**

```
function ReadIdentityExW(ReaderNumber: integer; identity: PEidIdentityW): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader. |
| identity: PEidIdentityW | The pointer to the identity information structure |

**Returns**

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

## 1.1.254 ReadIdentityW Function

Read identity information from Belgian eID card

**Pascal**

```
function ReadIdentityW(identity: PEidIdentityW): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| identity: PEidIdentityW | The pointer to the identity information structure |

**Returns**

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

## 1.1.255 ReadNonRepudiationCertificate Function

Read Non Repudiation Certificate to memory

**Pascal**

```
function ReadNonRepudiationCertificate(Certificate: PEidCertificate): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| cerificate | The pointer to EidCertificate structure |

**Returns**

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

**Description**

Read Non Repudiation Certificate to EidCertificate structure

## 1.1.256 ReadPhoto Function

Reads a photo from a card

**Pascal**

```
function ReadPhoto(Photo: PeidPicture): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Photo: PeidPicture | The pointer to EidPicture 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 structure. This structure holds the raw image bytes and the length of the image bytes array

# 1.1.257 ReadPhotoAsBitmap Function

Reads the picture from the card, converts it to bitmap and returns the bitmap handle Decription: 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.

**Pascal**

```
function ReadPhotoAsBitmap: HBITMAP; stdcall;
```

**File**

SwelioEngine (see page 162)

**Returns**

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

# 1.1.258 ReadPhotoAsBitmapEx Function

Reads the picture from the card, converts it to bitmap and returns the bitmap handle Decription: 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.

**Pascal**

```
function ReadPhotoAsBitmapEx(ReaderNumber: integer): HBITMAP; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader. |

**Returns**

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

# 1.1.259 ReadPhotoEx Function

Reads a photo from a card

**Pascal**

```
function ReadPhotoEx(ReaderNumber: integer; Photo: PeidPicture): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader. |
| Photo: PeidPicture | The pointer to EidPicture 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 structure

## 1.1.260 ReadRootCaCertificate Function

Read Root Ca Certificate to memory

**Pascal**

```
function ReadRootCaCertificate(Certificate: PEidCertificate): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| cerificate | The pointer to EidCertificate structure |

**Returns**

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

**Description**

Read Root Ca Certificate to EidCertificate structure

## 1.1.261 ReadRrnCertificate Function

Read Rrn Certificate to memory

**Pascal**

```
function ReadRrnCertificate(Certificate: PEidCertificate): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| cerificate | The pointer to EidCertificate structure |

**Returns**

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

**Description**

Read Rrn Certificate to EidCertificate structure

## 1.1.262 ReadSISCard Function

Read Belgian SIS card.

**Pascal**

```
function ReadSISCard(Identity: PSISRecord): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| PSISRecordW | The pointer to SISRecordW 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)

## 1.1.263 ReadSISCardA Function

Read Belgian SIS card.

**Pascal**

```
function ReadSISCardA(Identity: PSISRecordA): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| PSISRecordW | The pointer to SISRecordW 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)

## 1.1.264 ReadSISCardEx Function

Read Belgian SIS card.

**Pascal**

```
function ReadSISCardEx(ReaderNumber: integer; Identity: PSISRecord): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader. |
| PSISRecordA | The pointer to SISRecordA 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)

## 1.1.265 ReadSISCardExA Function

Read Belgian SIS card.

**Pascal**

```
function ReadSISCardExA(ReaderNumber: integer; Identity: PSISRecordA): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader. |
| PSISRecordA | The pointer to SISRecordA 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)

## 1.1.266 ReadSISCardExW Function

Read Belgian SIS card.

**Pascal**

```
function ReadSISCardExW(ReaderNumber: integer; Identity: PSISRecordW): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader. |
| PSISRecordA | The pointer to SISRecordA 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)

# 1.1.267 ReadSISCardW Function

Read Belgian SIS card.

**Pascal**

```
function ReadSISCardW(Identity: PSISRecordW): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| PSISRecordW | The pointer to SISRecordW 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)

# 1.1.268 ReloadReadersList Function

Reloads the list of the available card readers

**Pascal**

```
procedure ReloadReadersList; stdcall;
```

**File**

SwelioEngine (see page 162)

**Description**

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

# 1.1.269 RemoveCallback Function

Remove callback procedure for card events

**Pascal**

```
procedure RemoveCallback; stdcall;
```

**File**

SwelioEngine (see page 162)

**Description**

Use this function to deactivate card events callback procedure

## 1.1.270 RemoveStartup Function

Removes the application from the list of the automatically started applications

**Pascal**

```
procedure RemoveStartup(const AppName: PChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| const AppName: PChar | The name of the application |

**Description**

For application that starts automatically when Windows starts removes it from the automatically launching applications list

## 1.1.271 RemoveStartupA Function

Removes the application from the list of the automatically started applications

**Pascal**

```
procedure RemoveStartupA(const AppName: PAnsiChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| const AppName: PAnsiChar | The name of the application |

**Description**

For application that starts automatically when Windows starts removes it from the automatically launching applications list

## 1.1.272 RemoveStartupW Function

Removes the application from the list of the automatically started applications

**Pascal**

```
procedure RemoveStartupW(const AppName: PWideChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| const AppName: PWideChar | The name of the application |

**Description**

For application that starts automatically when Windows starts removes it from the automatically launching applications list

## 1.1.273 RestoreWindowSubclass Function

Restores window standard procedure

**Pascal**

```
procedure RestoreWindowSubclass(Hwnd: THandle); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Hwnd: THandle | The window handle |

## 1.1.274 RestoreWindowSubclassA Function

Restores window standard procedure

**Pascal**

```
procedure RestoreWindowSubclassA(Hwnd: THandle); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Hwnd: THandle | The window handle |

## 1.1.275 RestoreWindowSubclassW Function

Restores window standard procedure

**Pascal**

```
procedure RestoreWindowSubclassW(Hwnd: THandle); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Hwnd: THandle | The window handle |

## 1.1.276 SaveAuthenticationCertificate Function

Save Authentication Certificate to a file

**Pascal**

```
procedure SaveAuthenticationCertificate(FileName: PChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PChar | File name to store the certificate |

**Description**

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

## 1.1.277 SaveAuthenticationCertificateA Function

Save Authentication Certificate to a file

**Pascal**

```
procedure SaveAuthenticationCertificateA(FileName: PAnsiChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PAnsiChar | File name to store the certificate |

**Description**

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

## 1.1.278 SaveAuthenticationCertificateW Function

Save Authentication Certificate to a file

**Pascal**

```
procedure SaveAuthenticationCertificateW(FileName: PWideChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PWideChar | File name to store the certificate |

**Description**

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

## 1.1.279 SaveCaCertificate Function

Save Ca Certificate to a file

**Pascal**

```
procedure SaveCaCertificate(FileName: PChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PChar | File name to store the certificate |

**Description**

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

## 1.1.280 SaveCaCertificateA Function

Save Ca Certificate to a file

**Pascal**

```
procedure SaveCaCertificateA(FileName: PAnsiChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PAnsiChar | File name to store the certificate |

**Description**

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

## 1.1.281 SaveCaCertificateW Function

Save Ca Certificate to a file

**Pascal**

```
procedure SaveCaCertificateW(FileName: PWideChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PWideChar | File name to store the certificate |

**Description**

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

# 1.1.282 SaveCardToXml Function

Read eID card and save the information to XML file

**Pascal**

```
function SaveCardToXml(FileName: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PChar | 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.

# 1.1.283 SaveCardToXmlA Function

Read eID card and save the information to XML file

**Pascal**

```
function SaveCardToXmlA(FileName: PAnsiChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PAnsiChar | 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.

# 1.1.284 SaveCardToXmlEx Function

Read eID card and save the information to XML file

**Pascal**

```
function SaveCardToXmlEx(ReaderNumber: integer; FileName: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader. |
| FileName: PChar | 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.

## 1.1.285 SaveCardToXmlExA Function

Read eID card and save the information to XML file

**Pascal**

```
function SaveCardToXmlExA(ReaderNumber: integer; FileName: PAnsiChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader. |
| FileName: PAnsiChar | 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.

## 1.1.286 SaveCardToXmlExW Function

Read eID card and save the information to XML file

**Pascal**

```
function SaveCardToXmlExW(ReaderNumber: integer; FileName: PWideChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader. |
| FileName: PWideChar | 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.

## 1.1.287 SaveCardToXmlW Function

Read eID card and save the information to XML file

**Pascal**

```
function SaveCardToXmlW(FileName: PWideChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PWideChar | 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.

## 1.1.288 SaveIdentity Function

Saves indentity information to a file

**Pascal**

```
procedure SaveIdentity(FileName: PChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PChar | 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 95) to read this information from the file to EidIdentityW structure

## 1.1.289 SaveIdentityA Function

Saves indentity information to a file

**Pascal**

```
procedure SaveIdentityA(FileName: PAnsiChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PAnsiChar | 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 95) to read this information from the file to EidIdentityW structure

## 1.1.290 SaveIdentityW Function

Saves indentity infornation to a file

**Pascal**

```
procedure SaveIdentityW(FileName: PWideChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PWideChar | 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 95) to read this information from the file to EidIdentityW structure

## 1.1.291 SaveNonRepudiationCertificate Function

Save Non Repudiation Certificate to a file

**Pascal**

```
procedure SaveNonRepudiationCertificate(FileName: PChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PChar | File name to store the certificate |

**Description**

Read Non Repudiation Certificate from the card and save it to a file

## 1.1.292 SaveNonRepudiationCertificateA Function

Save Non Repudiation Certificate to a file

**Pascal**

```
procedure SaveNonRepudiationCertificateA(FileName: PAnsiChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PAnsiChar | File name to store the certificate |

**Description**

Read Non Repudiation Certificate from the card and save it to a file

## 1.1.293 SaveNonRepudiationCertificateW Function

Save Non Repudiation Certificate to a file

**Pascal**

```
procedure SaveNonRepudiationCertificateW(FileName: PWideChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PWideChar | File name to store the certificate |

**Description**

Read Non Repudiation Certificate from the card and save it to a file

## 1.1.294 SavePersonToCsv Function

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

**Pascal**

```
function SavePersonToCsv(FileName: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PChar | 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.

# 1.1.295 SavePersonToCsvA Function

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

**Pascal**

```
function SavePersonToCsvA(FileName: PAnsiChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PAnsiChar | 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.

# 1.1.296 SavePersonToCsvEx Function

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

**Pascal**

```
function SavePersonToCsvEx(ReaderNumber: integer; FileName: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader. |
| FileName: PChar | 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.

# 1.1.297 SavePersonToCsvExA Function

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

**Pascal**

```
function SavePersonToCsvExA(ReaderNumber: integer; FileName: PAnsiChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader. |
| FileName: PAnsiChar | 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.

## 1.1.298 SavePersonToCsvExW Function

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

**Pascal**

```
function SavePersonToCsvExW(ReaderNumber: integer; FileName: PWideChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader. |
| FileName: PWideChar | 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.

## 1.1.299 SavePersonToCsvW Function

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

**Pascal**

```
function SavePersonToCsvW(FileName: PWideChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PWideChar | 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.

# 1.1.300 SavePhoto Function

Saves photo to a file

**Pascal**

```
procedure SavePhoto(Photo: PeidPicture; FileName: PChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Photo: PeidPicture | The pointer to EidPicture structure |
| FileName: PChar | Destination file name |

**Description**

Saves the raw picture data to a file

# 1.1.301 SavePhotoA Function

Saves photo to a file

**Pascal**

```
procedure SavePhotoA(Photo: PeidPicture; FileName: PAnsiChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Photo: PeidPicture | The pointer to EidPicture structure |
| FileName: PAnsiChar | Destination file name |

**Description**

Saves the raw picture data to a file

# 1.1.302 SavePhotoAsBitmap Function

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.

**Pascal**

```
function SavePhotoAsBitmap(FileName: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PChar | File name to store the photo |

**Returns**

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

## 1.1.303 SavePhotoAsBitmapA Function

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.

**Pascal**

```
function SavePhotoAsBitmapA(FileName: PAnsiChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PAnsiChar | File name to store the photo |

**Returns**

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

## 1.1.304 SavePhotoAsBitmapEx Function

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.

**Pascal**

```
function SavePhotoAsBitmapEx(ReaderNumber: integer; FileName: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | the zero-based index of the card reader. |
| FileName: PChar | File name to store the photo |

**Returns**

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

## 1.1.305 SavePhotoAsBitmapExA Function

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.

**Pascal**

```
function SavePhotoAsBitmapExA(ReaderNumber: integer; FileName: PAnsiChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | the zero-based index of the card reader. |
| FileName: PAnsiChar | File name to store the photo |

**Returns**

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

## 1.1.306 SavePhotoAsBitmapExW Function

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.

**Pascal**

```
function SavePhotoAsBitmapExW(ReaderNumber: integer; FileName: PWideChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | the zero-based index of the card reader. |
| FileName: PWideChar | File name to store the photo |

**Returns**

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

## 1.1.307 SavePhotoAsBitmapW Function

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.

**Pascal**

```
function SavePhotoAsBitmapW(FileName: PWideChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PWideChar | File name to store the photo |

**Returns**

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

## 1.1.308 SavePhotoAsJpeg Function

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.

**Pascal**

```
function SavePhotoAsJpeg(FileName: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PChar | File name to store the photo |

**Returns**

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

## 1.1.309 SavePhotoAsJpegA Function

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.

**Pascal**

```
function SavePhotoAsJpegA(FileName: PAnsiChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PAnsiChar | File name to store the photo |

**Returns**

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

## 1.1.310 SavePhotoAsJpegEx Function

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.

**Pascal**

```
function SavePhotoAsJpegEx(ReaderNumber: integer; FileName: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader. |
| FileName: PChar | File name to store the photo |

**Returns**

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

## 1.1.311 SavePhotoAsJpegExA Function

Save the picture from the card to JPG file 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.

**Pascal**

```
function SavePhotoAsJpegExA(ReaderNumber: integer; FileName: PAnsiChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader. |
| FileName: PAnsiChar | File name to store the photo |

**Returns**

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

## 1.1.312 SavePhotoAsJpegExW Function

Save the picture from the card to JPG file 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.

**Pascal**

```
function SavePhotoAsJpegExW(ReaderNumber: integer; FileName: PWideChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader. |
| FileName: PWideChar | File name to store the photo |

**Returns**

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

## 1.1.313 SavePhotoAsJpegW Function

Save the picture from the card to JPG file 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.

**Pascal**

```
function SavePhotoAsJpegW(FileName: PWideChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PWideChar | File name to store the photo |

**Returns**

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

## 1.1.314 SavePhotoW Function

Saves photo to a file

**Pascal**

```
procedure SavePhotoW(Photo: PeidPicture; FileName: PWideChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Photo: PeidPicture | The pointer to EidPicture structure |
| FileName: PWideChar | Destination file name |

**Description**

Saves the raw picture data to a file

## 1.1.315 SaveRootCaCertificate Function

Save Root Ca Certificate to a file

**Pascal**

```
procedure SaveRootCaCertificate(FileName: PChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PChar | File name to store the certificate |

**Description**

Read Root CA certificate from the card and save it to a file

## 1.1.316 SaveRootCaCertificateA Function

Save Root Ca Certificate to a file

**Pascal**

```
procedure SaveRootCaCertificateA(FileName: PAnsiChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PAnsiChar | File name to store the certificate |

**Description**

Read Root CA certificate from the card and save it to a file

## 1.1.317 SaveRootCaCertificateW Function

Save Root Ca Certificate to a file

**Pascal**

```
procedure SaveRootCaCertificateW(FileName: PWideChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PWideChar | File name to store the certificate |

**Description**

Read Root CA certificate from the card and save it to a file

## 1.1.318 SaveRrnCertificate Function

Save RRN Certificate to a file

**Pascal**

```
procedure SaveRrnCertificate(FileName: PChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PChar | File name to store the certificate |

**Description**

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

## 1.1.319 SaveRrnCertificateA Function

Save RRN Certificate to a file

**Pascal**

```
procedure SaveRrnCertificateA(FileName: PAnsiChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PAnsiChar | File name to store the certificate |

**Description**

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

## 1.1.320 SaveRrnCertificateW Function

Save RRN Certificate to a file

**Pascal**

```
procedure SaveRrnCertificateW(FileName: PWideChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PWideChar | File name to store the certificate |

**Description**

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

## 1.1.321 SelectReader Function

When more than 1 reader connected, select the reader with specified number The first reader has number 0

**Pascal**

```
function SelectReader(ReaderNumber: integer): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The reader index, starting from 0 |

**Returns**

TRUE if the reader is selected, FALSE if the reader with specified number does not exist

## 1.1.322 SelectReaderByName Function

Select active smart card reader by providing the reader name

**Pascal**

```
function SelectReaderByName(ReaderName: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderName: PChar | 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

## 1.1.323 SelectReaderByNameA Function

Select active smart card reader by providing the reader name

**Pascal**

```
function SelectReaderByNameA(ReaderName: PAnsiChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderName: PAnsiChar | the name of the card reader |

**Returns**

TRUE if the reader is selected, FALSE if the reader with specified number does not exist

# 1.1.324 SelectReaderByNameW Function

Select active smart card reader by providing the reader name

**Pascal**

```
function SelectReaderByNameW(ReaderName: PWideChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Returns**

TRUE if the reader is selected, FALSE if the reader with specified number does not exist

# 1.1.325 SetCallback Function

Activates callback procedure for card status change event

**Pascal**

```
procedure SetCallback(callback: TReaderCallback; userContext: Pointer); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| callback: TReaderCallback | The pointer to callback procedure |
| userContext: Pointer | 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

# 1.1.326 SetMWCompatibility Function

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

**Pascal**

```
procedure SetMWCompatibility; stdcall;
```

**File**

SwelioEngine (see page 162)

**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

## 1.1.327 SetStartup Function

Register application to run when Windows starts

**Pascal**

```
procedure SetStartup(const AppName: PChar; const AppPath: PChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| const AppName: PChar | The name of the application |
| const AppPath: PChar | The path to the application executable |

## 1.1.328 SetStartupA Function

Register application to run when Windows starts

**Pascal**

```
procedure SetStartupA(const AppName: PAnsiChar; const AppPath: PAnsiChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| const AppName: PAnsiChar | The name of the application |
| const AppPath: PAnsiChar | The path to the application executable |

## 1.1.329 SetStartupW Function

Register application to run when Windows starts

**Pascal**

```
procedure SetStartupW(const AppName: PWideChar; const AppPath: PWideChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| const AppName: PWideChar | The name of the application |
| const AppPath: PWideChar | The path to the application executable |

# 1.1.330 SetSupportSIS Function

Activates or deactivates SIS card support by engine

**Pascal**

```
procedure SetSupportSIS(Value: BOOL); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Value: BOOL | 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.

# 1.1.331 ShellCopyFile Function

Copies file to the new location

**Pascal**

```
procedure ShellCopyFile(oldName: PChar; NewName: PChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| oldName: PChar | The source file name |
| NewName: PChar | The destination file name |

**Description**

Copies file to the new location using Windows shell copy routine

# 1.1.332 ShellCopyFileA Function

Copies file to the new location

**Pascal**

```
procedure ShellCopyFileA(OldName: PAnsiChar; NewName: PAnsiChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| OldName: PAnsiChar | The source file name |

| | |
|---|---|
| NewName: PAnsiChar | The destination file name |

**Description**

Copies file to the new location using Windows shell copy routine

## 1.1.333 ShellCopyFileW Function

Copies file to the new location

**Pascal**

```
procedure ShellCopyFileW(oldName: PWideChar; NewName: PWideChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| oldName: PWideChar | The source file name |
| NewName: PWideChar | The destination file name |

**Description**

Copies file to the new location using Windows shell copy routine

## 1.1.334 ShowError Function

Shows Dialog with the text message corresponding to the Windows error code

**Pascal**

```
procedure ShowError(ErrorCode: DWORD); stdcall;
```

**File**

SwelioEngine (see page 162)

## 1.1.335 ShutdownWindows Function

Logs off the interactive user, shuts down the system.

**Pascal**

```
function ShutdownWindows(Flags: UINT): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Returns**

If the function succeeds returns TRUE, otherwise returns FALSE

**Description**

Logs off the interactive user, shuts down the system, or shuts down and restarts the system. It sends the WM_QUERYENDSESSION message to all applications to determine if they can be terminated.

This function accepts the following parameter:

flags : The shutdown type. This parameter must include one of the following values:

| Value | Meaning |
|---|---|
| EWX_LOGOFF | Shuts down all processes running in the logon session of the process that called the ExitWindowsEx function. Then it logs the user off. |
| EWX_POWEROFF | Shuts down the system and turns off the power. The system must support the power-off feature. |
| EWX_REBOOT | Shuts down the system and then restarts the system. |
| EWX_RESTARTAPPS | Shuts down the system and then restarts it |
| EWX_SHUTDOWN | Shuts down the system to a point at which it is safe to turn off the power. |

## 1.1.336 StartEngine Function

Activates the Swelio Engine.

**Pascal**

```
function StartEngine: BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Returns**

Returns TRUE if the Swelio Engine is successfully started; otherwize returns FALSE

**Description**

This procedure must be called first before any other functions from Swelio library can be used.

## 1.1.337 StopEngine Function

Deactivates the Swelio Engine

**Pascal**

```
procedure StopEngine; stdcall;
```

**File**

SwelioEngine (see page 162)

**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.

## 1.1.338 StripFileName Function

Replaces environment variable names with values

**Pascal**

```
procedure StripFileName(const FileName: PChar; FullName: PChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| const FileName: PChar | The source file name |
| FullName: PChar | The expanded file name |

**Description**

This function expands environment-variable strings and replaces them with their defined values in the file name.

## 1.1.339 StripFileNameA Function

Replaces environment variable names with values

**Pascal**

```
procedure StripFileNameA(FileName: PAnsiChar; FullName: PAnsiChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PAnsiChar | The source file name |
| FullName: PAnsiChar | The expanded file name |

**Description**

This function expands environment-variable strings and replaces them with their defined values in the file name.

## 1.1.340 StripFileNameW Function

Replaces environment variable names with values

**Pascal**

```
procedure StripFileNameW(const FileName: PWideChar; FullName: PWideChar); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| const FileName: PWideChar | The source file name |
| FullName: PWideChar | The expanded file name |

**Description**

This function expands environment-variable strings and replaces them with their defined values in the file name.

## 1.1.341 SuspendWindows Function

Suspends Windows

**Pascal**

```
function SuspendWindows: BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

## 1.1.342 TurnMonitorOff Function

Turns the monitor off

**Pascal**

```
procedure TurnMonitorOff; stdcall;
```

**File**

SwelioEngine (see page 162)

## 1.1.343 TurnMonitorOn Function

Turns the monitor on

**Pascal**

```
procedure TurnMonitorOn; stdcall;
```

**File**

SwelioEngine (see page 162)

## 1.1.344 UpdateWindowPosition Function

Updated the window position

**Pascal**

```
procedure UpdateWindowPosition(Handle: THandle; X: integer; Y: integer); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Handle: THandle | The handle of the window |
| X: integer | New horizontal coordinate |
| Y: integer | New vertical coordinate |

## 1.1.345 VerifyPin Function

Verify PIN code

**Pascal**

```
function VerifyPin(Value: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Value: PChar | PIN code to verify |

**Returns**

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

## 1.1.346 VerifyPinA Function

Verify PIN code

**Pascal**

```
function VerifyPinA(Value: PAnsiChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Value: PAnsiChar | PIN code to verify |

**Returns**

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

## 1.1.347 VerifyPinEx Function

Verify PIN code

**Pascal**

```
function VerifyPinEx(ReaderNumber: integer; Value: PChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader. |
| Value: PChar | PIN code to verify |

**Returns**

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

## 1.1.348 VerifyPinExA Function

Verify PIN code

**Pascal**

```
function VerifyPinExA(ReaderNumber: integer; Value: PAnsiChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader. |
| Value: PAnsiChar | PIN code to verify |

**Returns**

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

## 1.1.349 VerifyPinExW Function

Verify PIN code

**Pascal**

```
function VerifyPinExW(ReaderNumber: integer; Value: PWideChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| ReaderNumber: integer | The zero-based index of the card reader. |
| Value: PWideChar | PIN code to verify |

**Returns**

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

## 1.1.350 VerifyPinW Function

Verify PIN code

**Pascal**

```
function VerifyPinW(Value: PWideChar): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Value: PWideChar | PIN code to verify |

**Returns**

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

## 1.1.351 VerifySignature Function

Verifies the signature from the specified hash value.

**Pascal**

```
function VerifySignature(Certificate: PEidCertificate; Buffer: PBYTE; BufferSize: integer;
Signature: PBYTE; signatureSize: DWORD): BOOL; stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| Certificate: PEidCertificate | The public certificate |
| Buffer: PBYTE | The hash buffer |
| BufferSize: integer | The size of the hash buffer |
| Signature: PBYTE | The signature to be verified. |
| signatureSize: DWORD | 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

## 1.1.352 WriteBufferToFile Function

Writes the memory buffer to file

**Pascal**

```
procedure WriteBufferToFile(FileName: PChar; Buffer: PBYTE; BufferSize: integer); stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PChar | The name of the file |
| Buffer: PBYTE | The address of the memory block |
| BufferSize: integer | The size of the memory block |

**Description**

This function stores the content of the memory buffer to the file.

## 1.1.353 WriteBufferToFileA Function

Writes the memory buffer to file

**Pascal**

```
procedure WriteBufferToFileA(FileName: PAnsiChar; Buffer: PBYTE; BufferSize: integer);
stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PAnsiChar | The name of the file |
| Buffer: PBYTE | The address of the memory block |
| BufferSize: integer | The size of the memory block |

**Description**

This function stores the content of the memory buffer to the file.

## 1.1.354 WriteBufferToFileW Function

Writes the memory buffer to file

**Pascal**

```
procedure WriteBufferToFileW(FileName: PWideChar; Buffer: PBYTE; BufferSize: integer);
stdcall;
```

**File**

SwelioEngine (see page 162)

**Parameters**

| Parameters | Description |
|---|---|
| FileName: PWideChar | The name of the file |
| Buffer: PBYTE | The address of the memory block |
| BufferSize: integer | 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 |
|---|---|---|
| | TCardEventType (see page 147) | The type of the reader event |

**Records**

| | Name | Description |
|---|---|---|
| | tagEidAddressA (see page 142) | EID address information, stored on the card |
| | tagEidAddressW (see page 142) | EID address information, stored on the card |
| | tagEidCertificate (see page 143) | Certificate, stored on EID card |
| | tagEidIdentityA (see page 143) | Identity information stored on EID card |
| | tagEidIdentityW (see page 144) | Identity information stored on EID card |
| | tagEidPicture (see page 145) | Raw picture data from EID card |
| | tagSISRecordA (see page 146) | Public information stored on Belgian SIS card. The SIS card is the social security card of each Belgian resident (Belgian or foreigner) |
| | tagSISRecordW (see page 146) | Public information stored on Belgian SIS card. The SIS card is the social security card of each Belgian resident (Belgian or foreigner) |

## 1.2.1 tagEidAddressA Record

EID address information, stored on the card

**Pascal**

```
tagEidAddressA = record
  Street: array[0..EID_MAX_STREET_LEN] of AnsiChar;
  Zip: array[0..EID_MAX_ZIP_LEN] of AnsiChar;
  Municipality: array[0..EID_MAX_MUNICIPALITY_LEN] of AnsiChar;
end;
```

**File**

SwelioEngine (see page 162)

**Members**

| Members | Description |
|---|---|
| Street: array[0..EID_MAX_STREET_LEN] of AnsiChar; | Street name |
| Zip: array[0..EID_MAX_ZIP_LEN] of AnsiChar; | ZIP code |
| Municipality: array[0..EID_MAX_MUNICIPALITY_LEN] of AnsiChar; | Municipality |

## 1.2.2 tagEidAddressW Record

EID address information, stored on the card

**Pascal**

```
tagEidAddressW = record
  Street: array[0..EID_MAX_STREET_LEN] of WideChar;
  Zip: array[0..EID_MAX_ZIP_LEN] of WideChar;
  Municipality: array[0..EID_MAX_MUNICIPALITY_LEN] of WideChar;
end;
```

**File**

SwelioEngine (see page 162)

**Members**

| Members | Description |
|---|---|
| Street: array[0..EID_MAX_STREET_LEN] of WideChar; | Street name |
| Zip: array[0..EID_MAX_ZIP_LEN] of WideChar; | ZIP code |

| | |
|---|---|
| Municipality: array[0..EID_MAX_MUNICIPALITY_LEN] of WideChar; | Municipality |

## 1.2.3 tagEidCertificate Record

Certificate, stored on EID card

**Pascal**

```
tagEidCertificate = record
  Certificate: array[0..EID_MAX_CERT_LEN] of byte;
  CertificateLength: integer;
end;
```

**File**

SwelioEngine (see page 162)

**Members**

| Members | Description |
|---|---|
| Certificate: array[0..EID_MAX_CERT_LEN] of byte; | Certificate raw data buffer |
| CertificateLength: integer; | Certificate data length |

## 1.2.4 tagEidIdentityA Record

Identity information stored on EID card

**Pascal**

```
tagEidIdentityA = record
  CardNumber: array[0..EID_MAX_CARD_NUMBER_LEN] of AnsiChar;
  ChipNumber: array[0..EID_MAX_CHIP_NUMBER_LEN] of AnsiChar;
  ValidityDateBegin: array[0..EID_MAX_DATE_BEGIN_LEN] of AnsiChar;
  ValidityDateEnd: array[0..EID_MAX_DATE_END_LEN] of AnsiChar;
  Municipality: array[0..EID_MAX_DELIVERY_MUNICIPALITY_LEN] of AnsiChar;
  NationalNumber: array[0..EID_MAX_NATIONAL_NUMBER_LEN] of AnsiChar;
  Name: array[0..EID_MAX_NAME_LEN] of AnsiChar;
  FirstName1: array[0..EID_MAX_FIRST_NAME1_LEN] of AnsiChar;
  FirstName2: array[0..EID_MAX_FIRST_NAME2_LEN] of AnsiChar;
  Nationality: array[0..EID_MAX_NATIONALITY_LEN] of AnsiChar;
  BirthLocation: array[0..EID_MAX_BIRTHPLACE_LEN] of AnsiChar;
  BirthDate: array[0..EID_MAX_BIRTHDATE_LEN] of AnsiChar;
  Sex: array[0..EID_MAX_SEX_LEN] of AnsiChar;
  NobleCondition: array[0..EID_MAX_NOBLE_CONDITION_LEN] of AnsiChar;
  DocumentType: Longint;
  WhiteCane: BOOL;
  YellowCane: BOOL;
  ExtendedMinority: BOOL;
end;
```

**File**

SwelioEngine (see page 162)

**Members**

| Members | Description |
|---|---|
| CardNumber: array[0..EID_MAX_CARD_NUMBER_LEN] of AnsiChar; | Electronic ID card number |
| ChipNumber: array[0..EID_MAX_CHIP_NUMBER_LEN] of AnsiChar; | Electronic ID card physical chip number |

| | |
|---|---|
| ValidityDateBegin: array[0..EID_MAX_DATE_BEGIN_LEN] of AnsiChar; | Card validity date begin |
| ValidityDateEnd: array[0..EID_MAX_DATE_END_LEN] of AnsiChar; | Card validity date end |
| Municipality: array[0..EID_MAX_DELIVERY_MUNICIPALITY_LEN] of AnsiChar; | Card delivery municipality |
| NationalNumber: array[0..EID_MAX_NATIONAL_NUMBER_LEN] of AnsiChar; | National number |
| Name: array[0..EID_MAX_NAME_LEN] of AnsiChar; | Surname |
| FirstName1: array[0..EID_MAX_FIRST_NAME1_LEN] of AnsiChar; | First name (2 first given names) |
| FirstName2: array[0..EID_MAX_FIRST_NAME2_LEN] of AnsiChar; | First name part 2 (first letter of the 3rd given name). |
| Nationality: array[0..EID_MAX_NATIONALITY_LEN] of AnsiChar; | Nationality |
| BirthLocation: array[0..EID_MAX_BIRTHPLACE_LEN] of AnsiChar; | Birth location |
| BirthDate: array[0..EID_MAX_BIRTHDATE_LEN] of AnsiChar; | Birth date |
| Sex: array[0..EID_MAX_SEX_LEN] of AnsiChar; | Sex |
| NobleCondition: array[0..EID_MAX_NOBLE_CONDITION_LEN] of AnsiChar; | Noble condition |
| DocumentType: Longint; | Document type code (Belgian citizen card, Kids Card, Foreigner card) |
| WhiteCane: BOOL; | White cane (blind people) |
| YellowCane: BOOL; | Yellow cane (partially sighted people) |
| ExtendedMinority: BOOL; | Extended minority |

## 1.2.5 tagEidIdentityW Record

Identity information stored on EID card

**Pascal**

```
tagEidIdentityW = record
  CardNumber: array[0..EID_MAX_CARD_NUMBER_LEN] of WideChar;
  ChipNumber: array[0..EID_MAX_CHIP_NUMBER_LEN] of WideChar;
  ValidityDateBegin: array[0..EID_MAX_DATE_BEGIN_LEN] of WideChar;
  ValidityDateEnd: array[0..EID_MAX_DATE_END_LEN] of WideChar;
  Municipality: array[0..EID_MAX_DELIVERY_MUNICIPALITY_LEN] of WideChar;
  NationalNumber: array[0..EID_MAX_NATIONAL_NUMBER_LEN] of WideChar;
  Name: array[0..EID_MAX_NAME_LEN] of WideChar;
  FirstName1: array[0..EID_MAX_FIRST_NAME1_LEN] of WideChar;
  FirstName2: array[0..EID_MAX_FIRST_NAME2_LEN] of WideChar;
  Nationality: array[0..EID_MAX_NATIONALITY_LEN] of WideChar;
  BirthLocation: array[0..EID_MAX_BIRTHPLACE_LEN] of WideChar;
  BirthDate: array[0..EID_MAX_BIRTHDATE_LEN] of WideChar;
  Sex: array[0..EID_MAX_SEX_LEN] of WideChar;
  NobleCondition: array[0..EID_MAX_NOBLE_CONDITION_LEN] of WideChar;
  DocumentType: Longint;
  WhiteCane: BOOL;
  YellowCane: BOOL;
  ExtendedMinority: BOOL;
end;
```

**File**

SwelioEngine (see page 162)

**Members**

| Members | Description |
|---|---|
| CardNumber: array[0..EID_MAX_CARD_NUMBER_LEN] of WideChar; | Electronic ID card number |
| ChipNumber: array[0..EID_MAX_CHIP_NUMBER_LEN] of WideChar; | Electronic ID card physical chip number |
| ValidityDateBegin: array[0..EID_MAX_DATE_BEGIN_LEN] of WideChar; | Card validity date begin |
| ValidityDateEnd: array[0..EID_MAX_DATE_END_LEN] of WideChar; | Card validity date end |
| Municipality: array[0..EID_MAX_DELIVERY_MUNICIPALITY_LEN] of WideChar; | Card delivery municipality |
| NationalNumber: array[0..EID_MAX_NATIONAL_NUMBER_LEN] of WideChar; | National number |
| Name: array[0..EID_MAX_NAME_LEN] of WideChar; | Surname |
| FirstName1: array[0..EID_MAX_FIRST_NAME1_LEN] of WideChar; | First name (2 first given names) |
| FirstName2: array[0..EID_MAX_FIRST_NAME2_LEN] of WideChar; | First name part 2 (first letter of the 3rd given name). |
| Nationality: array[0..EID_MAX_NATIONALITY_LEN] of WideChar; | Nationality |
| BirthLocation: array[0..EID_MAX_BIRTHPLACE_LEN] of WideChar; | Birth location |
| BirthDate: array[0..EID_MAX_BIRTHDATE_LEN] of WideChar; | Birth date |
| Sex: array[0..EID_MAX_SEX_LEN] of WideChar; | Sex |
| NobleCondition: array[0..EID_MAX_NOBLE_CONDITION_LEN] of WideChar; | Noble condition |
| DocumentType: Longint; | Document type code (Belgian citizen card, Kids Card, Foreigner card) |
| WhiteCane: BOOL; | White cane (blind people) |
| YellowCane: BOOL; | Yellow cane (partially sighted people) |
| ExtendedMinority: BOOL; | Extended minority |

## 1.2.6 tagEidPicture Record

Raw picture data from EID card

**Pascal**

```
tagEidPicture = record
  Picture: array[0..EID_MAX_PICTURE_LEN] of byte;
  PictureLength: integer;
end;
```

**File**

SwelioEngine (see page 162)

**Members**

| Members | Description |
|---|---|
| Picture: array[0..EID_MAX_PICTURE_LEN] of byte; | Picture raw data buffer |
| PictureLength: integer; | Picture raw data buffer length |

## 1.2.7 tagSISRecordA Record

Public information stored on Belgian SIS card. The SIS card is the social security card of each Belgian resident (Belgian or foreigner)

**Pascal**

```
tagSISRecordA = record
  Name: array[0..SIS_MAX_NAME_LEN] of AnsiChar;
  FirstName: array[0..SIS_MAX_FIRSTNAMES_LEN] of AnsiChar;
  Initial: array[0..SIS_MAX_INITIAL_LEN] of AnsiChar;
  Sex: array[0..SIS_MAX_SEX_LEN] of AnsiChar;
  BirthDate: array[0..SIS_FIELD_MAX_BIRTHDATE_LEN] of AnsiChar;
  SocialSecurityNumber: array[0..SIS_FIELD_MAX_SOCIAL_SECURITY_NUMBER_LEN] of AnsiChar;
  CaptureDate: array[0..SIS_FIELD_MAX_CAPTUREDATE_LEN] of AnsiChar;
  ValidityDateBegin: array[0..SIS_FIELD_MAX_VALIDBEGIN_LEN] of AnsiChar;
  ValidityDateEnd: array[0..SIS_FIELD_MAX_VALIDEND_LEN] of AnsiChar;
  CardNumber: array[0..SIS_FIELD_MAX_CARDNUMBER_LEN] of AnsiChar;
  CardName: array[0..SIS_MAX_CARDNAME_LEN] of AnsiChar;
end;
```

**File**

SwelioEngine (see page 162)

**Members**

| Members | Description |
|---|---|
| Name: array[0..SIS_MAX_NAME_LEN] of AnsiChar; | Family name of the card owner |
| FirstName: array[0..SIS_MAX_FIRSTNAMES_LEN] of AnsiChar; | First name of the card owner |
| Initial: array[0..SIS_MAX_INITIAL_LEN] of AnsiChar; | Initial of the card owner |
| Sex: array[0..SIS_MAX_SEX_LEN] of AnsiChar; | Sex of the card owner |
| BirthDate: array[0..SIS_FIELD_MAX_BIRTHDATE_LEN] of AnsiChar; | Birth date of the card owner |
| SocialSecurityNumber: array[0..SIS_FIELD_MAX_SOCIAL_SECURITY_NUMBER_LEN] of AnsiChar; | Social security number of the card owner |
| CaptureDate: array[0..SIS_FIELD_MAX_CAPTUREDATE_LEN] of AnsiChar; | Date of issue |
| ValidityDateBegin: array[0..SIS_FIELD_MAX_VALIDBEGIN_LEN] of AnsiChar; | Card validity begin date |
| ValidityDateEnd: array[0..SIS_FIELD_MAX_VALIDEND_LEN] of AnsiChar; | Card validity end date |
| CardNumber: array[0..SIS_FIELD_MAX_CARDNUMBER_LEN] of AnsiChar; | Card number |
| CardName: array[0..SIS_MAX_CARDNAME_LEN] of AnsiChar; | Name of the card |

## 1.2.8 tagSISRecordW Record

Public information stored on Belgian SIS card. The SIS card is the social security card of each Belgian resident (Belgian or foreigner)

**Pascal**

```
tagSISRecordW = record
  Name: array[0..SIS_MAX_NAME_LEN] of WideChar;
  FirstName: array[0..SIS_MAX_FIRSTNAMES_LEN] of WideChar;
```

```
    Initial: array[0..SIS_MAX_INITIAL_LEN] of WideChar;
    Sex: array[0..SIS_MAX_SEX_LEN] of WideChar;
    BirthDate: array[0..SIS_FIELD_MAX_BIRTHDATE_LEN] of WideChar;
    SocialSecurityNumber: array[0..SIS_FIELD_MAX_SOCIAL_SECURITY_NUMBER_LEN] of WideChar;
    CaptureDate: array[0..SIS_FIELD_MAX_CAPTUREDATE_LEN] of WideChar;
    ValidityDateBegin: array[0..SIS_FIELD_MAX_VALIDBEGIN_LEN] of WideChar;
    ValidityDateEnd: array[0..SIS_FIELD_MAX_VALIDEND_LEN] of WideChar;
    CardNumber: array[0..SIS_FIELD_MAX_CARDNUMBER_LEN] of WideChar;
    CardName: array[0..SIS_MAX_CARDNAME_LEN] of WideChar;
  end;
```

**File**

SwelioEngine (see page 162)

**Members**

| Members | Description |
|---|---|
| Name: array[0..SIS_MAX_NAME_LEN] of WideChar; | Family name of the card owner |
| FirstName: array[0..SIS_MAX_FIRSTNAMES_LEN] of WideChar; | First name of the card owner |
| Initial: array[0..SIS_MAX_INITIAL_LEN] of WideChar; | Initial of the card owner |
| Sex: array[0..SIS_MAX_SEX_LEN] of WideChar; | Sex of the card owner |
| BirthDate: array[0..SIS_FIELD_MAX_BIRTHDATE_LEN] of WideChar; | Birth date of the card owner |
| SocialSecurityNumber: array[0..SIS_FIELD_MAX_SOCIAL_SECURITY_NUMBER_LEN] of WideChar; | Social security number of the card owner |
| CaptureDate: array[0..SIS_FIELD_MAX_CAPTUREDATE_LEN] of WideChar; | Date of issue |
| ValidityDateBegin: array[0..SIS_FIELD_MAX_VALIDBEGIN_LEN] of WideChar; | Card validity begin date |
| ValidityDateEnd: array[0..SIS_FIELD_MAX_VALIDEND_LEN] of WideChar; | Card validity end date |
| CardNumber: array[0..SIS_FIELD_MAX_CARDNUMBER_LEN] of WideChar; | Card number |
| CardName: array[0..SIS_MAX_CARDNAME_LEN] of WideChar; | Name of the card |

## 1.2.9 TCardEventType Enumeration

The type of the reader event

**Pascal**

```
TCardEventType = (
  ewtUnknownEvent,
  ewtCardInsert,
  ewtCardRemove,
  ewtReadersChange
);
```

**File**

SwelioEngine (see page 162)

**Members**

| Members | Description |
|---|---|
| ewtUnknownEvent | Unknown event |
| ewtCardInsert | Card was inserted in the reader |
| ewtCardRemove | Card was removed from the reader |
| ewtReadersChange | The readers list changed |

# 1.3 Types

The following table lists types in this documentation.

**Types**

| Name | Description |
|---|---|
| PEIDAddress (see page 148) | EID address information, stored on the card |
| PEIDAddressA (see page 148) | EID address information, stored on the card |
| PEIDAddressW (see page 149) | EID address information, stored on the card |
| PEIDCertificate (see page 149) | Certificate, stored on EID card |
| PEIDIdentity (see page 149) | Identity information stored on EID card |
| PEIDIdentityA (see page 149) | Identity information stored on EID card |
| PEIDIdentityW (see page 150) | Identity information stored on EID card |
| PEIDPicture (see page 150) | Raw picture data from EID card |
| PSISRecordA (see page 150) | Public information stored on Belgian SIS card |
| PSISRecordW (see page 150) | Public information stored on Belgian SIS card |
| TEIDAddress (see page 150) | EID address information, stored on the card |
| TEIDAddressA (see page 151) | EID address information, stored on the card |
| TEIDAddressW (see page 151) | EID address information, stored on the card |
| TEIDCertificate (see page 151) | Certificate, stored on EID card |
| TEIDIdentity (see page 151) | Identity information stored on EID card |
| TEIDIdentityA (see page 152) | Identity information stored on EID card |
| TEIDIdentityW (see page 152) | Identity information stored on EID card |
| TEIDPicture (see page 152) | Raw picture data from EID card |
| TReaderCallback (see page 152) | The smart card reader callback procedure |
| TSISRecord (see page 152) | Public information stored on Belgian SIS card |
| TSISRecordA (see page 153) | Public information stored on Belgian SIS card |
| TSISRecordW (see page 153) | Public information stored on Belgian SIS card |

## 1.3.1 PEIDAddress Type

EID address information, stored on the card

**Pascal**

```
PEIDAddress = ^TEIDAddressA;
```

**File**

SwelioEngine (see page 162)

## 1.3.2 PEIDAddressA Type

EID address information, stored on the card

**Pascal**

```
PEIDAddressA = ^TEIDAddressA;
```

**File**

SwelioEngine (see page 162)

## 1.3.3 PEIDAddressW Type

EID address information, stored on the card

**Pascal**

```
PEIDAddressW = ^TEIDAddressW;
```

**File**

SwelioEngine (see page 162)

## 1.3.4 PEIDCertificate Type

Certificate, stored on EID card

**Pascal**

```
PEIDCertificate = ^TEIDCertificate;
```

**File**

SwelioEngine (see page 162)

## 1.3.5 PEIDIdentity Type

Identity information stored on EID card

**Pascal**

```
PEIDIdentity = ^TEIDIdentityA;
```

**File**

SwelioEngine (see page 162)

## 1.3.6 PEIDIdentityA Type

Identity information stored on EID card

**Pascal**

```
PEIDIdentityA = ^TEIDIdentityA;
```

**File**

SwelioEngine (see page 162)

## 1.3.7 PEIDIdentityW Type

Identity information stored on EID card

**Pascal**

```
PEIDIdentityW = ^TEIDIdentityW;
```

**File**

SwelioEngine (see page 162)

## 1.3.8 PEIDPicture Type

Raw picture data from EID card

**Pascal**

```
PEIDPicture = ^TEIDPicture;
```

**File**

SwelioEngine (see page 162)

## 1.3.9 PSISRecordA Type

Public information stored on Belgian SIS card

**Pascal**

```
PSISRecordA = ^TSISRecordA;
```

**File**

SwelioEngine (see page 162)

## 1.3.10 PSISRecordW Type

Public information stored on Belgian SIS card

**Pascal**

```
PSISRecordW = ^TSISRecordW;
```

**File**

SwelioEngine (see page 162)

## 1.3.11 TEIDAddress Type

EID address information, stored on the card

**Pascal**

```
TEIDAddress = TEIDAddressA;
```

**File**

SwelioEngine (see page 162)

## 1.3.12 TEIDAddressA Type

EID address information, stored on the card

**Pascal**

```
TEIDAddressA = tagEidAddressA;
```

**File**

SwelioEngine (see page 162)

## 1.3.13 TEIDAddressW Type

EID address information, stored on the card

**Pascal**

```
TEIDAddressW = tagEidAddressW;
```

**File**

SwelioEngine (see page 162)

## 1.3.14 TEIDCertificate Type

Certificate, stored on EID card

**Pascal**

```
TEIDCertificate = tagEidCertificate;
```

**File**

SwelioEngine (see page 162)

## 1.3.15 TEIDIdentity Type

Identity information stored on EID card

**Pascal**

```
TEIDIdentity = TEIDIdentityA;
```

**File**

SwelioEngine (see page 162)

## 1.3.16 TEIDIdentityA Type

Identity information stored on EID card

**Pascal**

```
TEIDIdentityA = tagEidIdentityA;
```

**File**

SwelioEngine (see page 162)

## 1.3.17 TEIDIdentityW Type

Identity information stored on EID card

**Pascal**

```
TEIDIdentityW = tagEidIdentityW;
```

**File**

SwelioEngine (see page 162)

## 1.3.18 TEIDPicture Type

Raw picture data from EID card

**Pascal**

```
TEIDPicture = tagEidPicture;
```

**File**

SwelioEngine (see page 162)

## 1.3.19 TReaderCallback Type

The smart card reader callback procedure

**Pascal**

```
TReaderCallback = procedure (var ReaderNumber : DWORD; var EventCode : DWORD; UserContext :
Pointer);
```

**File**

SwelioEngine (see page 162)

## 1.3.20 TSISRecord Type

Public information stored on Belgian SIS card

**Pascal**

```
TSISRecord = TSISRecordA;
```

**File**

SwelioEngine (see page 162)

## 1.3.21 TSISRecordA Type

Public information stored on Belgian SIS card

**Pascal**

```
TSISRecordA = tagSISRecordA;
```

**File**

SwelioEngine (see page 162)

## 1.3.22 TSISRecordW Type

Public information stored on Belgian SIS card

**Pascal**

```
TSISRecordW = tagSISRecordW;
```

**File**

SwelioEngine (see page 162)

# 1.4 Constants

The following table lists constants in this documentation.

**Constants**

| Name | Description |
|---|---|
| EID_MAX_BIRTHDATE_LEN (see page 154) | Maximum length of the birthdate |
| EID_MAX_BIRTHPLACE_LEN (see page 154) | Maximum length of the birthplace |
| EID_MAX_CARD_NUMBER_LEN (see page 155) | Maximum length of the card number field |
| EID_MAX_CERT_LEN (see page 155) | Maximum length of the certificate data |
| EID_MAX_CHIP_NUMBER_LEN (see page 155) | Maximum length of the chip number field |
| EID_MAX_DATE_BEGIN_LEN (see page 155) | Maximum length of the begin date field |
| EID_MAX_DATE_END_LEN (see page 156) | Maximum length of the end date field |
| EID_MAX_DELIVERY_MUNICIPALITY_LEN (see page 156) | Maximum length of the name of the devivery municipality |
| EID_MAX_DOCUMENT_TYPE_LEN (see page 156) | Maximum length of the document type field |
| EID_MAX_FIRST_NAME1_LEN (see page 156) | Maximum length of the first name |

| | |
|---|---|
| EID_MAX_FIRST_NAME2_LEN (see page 156) | Maximum length of the first name |
| EID_MAX_MUNICIPALITY_LEN (see page 157) | Maximum length of the municipality name field |
| EID_MAX_NAME_LEN (see page 157) | Maximum length of the surname |
| EID_MAX_NATIONAL_NUMBER_LEN (see page 157) | Maximum length of the national number |
| EID_MAX_NATIONALITY_LEN (see page 157) | Maximum length of the nationality |
| EID_MAX_NOBLE_CONDITION_LEN (see page 158) | Maximum length of the noble condition field |
| EID_MAX_PICTURE_LEN (see page 158) | Maximum length of the picture data |
| EID_MAX_SEX_LEN (see page 158) | Maximum length of the sex field |
| EID_MAX_SPECIAL_STATUS_LEN (see page 158) | Maximum length of the special status field |
| EID_MAX_STREET_LEN (see page 158) | Maximum length of the street name field |
| EID_MAX_ZIP_LEN (see page 159) | Maximum length of the ZIP code field |
| SIS_FIELD_MAX_BIRTHDATE_LEN (see page 159) | Maximum length of the birth date field |
| SIS_FIELD_MAX_CAPTUREDATE_LEN (see page 159) | Maximum length of the capture date field |
| SIS_FIELD_MAX_CARDNUMBER_LEN (see page 159) | Maximum length of the car number field |
| SIS_FIELD_MAX_SOCIAL_SECURITY_NUMBER_LEN (see page 160) | Maximum length of the social security number field |
| SIS_FIELD_MAX_VALIDBEGIN_LEN (see page 160) | Maximum length of the start validity date field |
| SIS_FIELD_MAX_VALIDEND_LEN (see page 160) | Maximum length of the end validity date field |
| SIS_MAX_CARDNAME_LEN (see page 160) | Maximum length of the card name field |
| SIS_MAX_FIRSTNAMES_LEN (see page 161) | Maximum length of the first name field |
| SIS_MAX_INITIAL_LEN (see page 161) | Maximum length of the initial field |
| SIS_MAX_NAME_LEN (see page 161) | Maximum length of the surname field |
| SIS_MAX_SEX_LEN (see page 161) | Maximum length of the sex field |

## 1.4.1 EID_MAX_BIRTHDATE_LEN Constant

Maximum length of the birthdate

**Pascal**

```
EID_MAX_BIRTHDATE_LEN = 12;
```

**File**

SwelioEngine (see page 162)

## 1.4.2 EID_MAX_BIRTHPLACE_LEN Constant

Maximum length of the birthplace

**Pascal**

```
EID_MAX_BIRTHPLACE_LEN = 80;
```

**File**

SwelioEngine (see page 162)

## 1.4.3 EID_MAX_CARD_NUMBER_LEN Constant

Maximum length of the card number field

**Pascal**

```
EID_MAX_CARD_NUMBER_LEN = 12;
```

**File**

SwelioEngine (see page 162)

## 1.4.4 EID_MAX_CERT_LEN Constant

Maximum length of the certificate data

**Pascal**

```
EID_MAX_CERT_LEN = 2048;
```

**File**

SwelioEngine (see page 162)

## 1.4.5 EID_MAX_CHIP_NUMBER_LEN Constant

Maximum length of the chip number field

**Pascal**

```
EID_MAX_CHIP_NUMBER_LEN = 32;
```

**File**

SwelioEngine (see page 162)

## 1.4.6 EID_MAX_DATE_BEGIN_LEN Constant

Maximum length of the begin date field

**Pascal**

```
EID_MAX_DATE_BEGIN_LEN = 10;
```

**File**

SwelioEngine (see page 162)

## 1.4.7 EID_MAX_DATE_END_LEN Constant

Maximum length of the end date field

**Pascal**

```
EID_MAX_DATE_END_LEN = 10;
```

**File**

SwelioEngine (see page 162)

## 1.4.8 EID_MAX_DELIVERY_MUNICIPALITY_LEN Constant

Maximum length of the name of the devivery municipality

**Pascal**

```
EID_MAX_DELIVERY_MUNICIPALITY_LEN = 80;
```

**File**

SwelioEngine (see page 162)

## 1.4.9 EID_MAX_DOCUMENT_TYPE_LEN Constant

Maximum length of the document type field

**Pascal**

```
EID_MAX_DOCUMENT_TYPE_LEN = 2;
```

**File**

SwelioEngine (see page 162)

## 1.4.10 EID_MAX_FIRST_NAME1_LEN Constant

Maximum length of the first name

**Pascal**

```
EID_MAX_FIRST_NAME1_LEN = 95;
```

**File**

SwelioEngine (see page 162)

## 1.4.11 EID_MAX_FIRST_NAME2_LEN Constant

Maximum length of the first name

**Pascal**

```
EID_MAX_FIRST_NAME2_LEN = 3;
```

**File**

SwelioEngine (see page 162)

## 1.4.12 EID_MAX_MUNICIPALITY_LEN Constant

Maximum length of the municipality name field

**Pascal**

```
EID_MAX_MUNICIPALITY_LEN = 67;
```

**File**

SwelioEngine (see page 162)

## 1.4.13 EID_MAX_NAME_LEN Constant

Maximum length of the surname

**Pascal**

```
EID_MAX_NAME_LEN = 110;
```

**File**

SwelioEngine (see page 162)

## 1.4.14 EID_MAX_NATIONAL_NUMBER_LEN Constant

Maximum length of the national number

**Pascal**

```
EID_MAX_NATIONAL_NUMBER_LEN = 11;
```

**File**

SwelioEngine (see page 162)

## 1.4.15 EID_MAX_NATIONALITY_LEN Constant

Maximum length of the nationality

**Pascal**

```
EID_MAX_NATIONALITY_LEN = 85;
```

**File**

SwelioEngine (see page 162)

## 1.4.16 EID_MAX_NOBLE_CONDITION_LEN Constant

Maximum length of the noble condition field

**Pascal**

```
EID_MAX_NOBLE_CONDITION_LEN = 50;
```

**File**

SwelioEngine (see page 162)

## 1.4.17 EID_MAX_PICTURE_LEN Constant

Maximum length of the picture data

**Pascal**

```
EID_MAX_PICTURE_LEN = 4096;
```

**File**

SwelioEngine (see page 162)

## 1.4.18 EID_MAX_SEX_LEN Constant

Maximum length of the sex field

**Pascal**

```
EID_MAX_SEX_LEN = 1;
```

**File**

SwelioEngine (see page 162)

## 1.4.19 EID_MAX_SPECIAL_STATUS_LEN Constant

Maximum length of the special status field

**Pascal**

```
EID_MAX_SPECIAL_STATUS_LEN = 2;
```

**File**

SwelioEngine (see page 162)

## 1.4.20 EID_MAX_STREET_LEN Constant

Maximum length of the street name field

**Pascal**

```
EID_MAX_STREET_LEN = 80;
```

**File**

SwelioEngine (see page 162)

## 1.4.21 EID_MAX_ZIP_LEN Constant

Maximum length of the ZIP code field

**Pascal**

```
EID_MAX_ZIP_LEN = 4;
```

**File**

SwelioEngine (see page 162)

## 1.4.22 SIS_FIELD_MAX_BIRTHDATE_LEN Constant

Maximum length of the birth date field

**Pascal**

```
SIS_FIELD_MAX_BIRTHDATE_LEN = 8;
```

**File**

SwelioEngine (see page 162)

## 1.4.23 SIS_FIELD_MAX_CAPTUREDATE_LEN Constant

Maximum length of the capture date field

**Pascal**

```
SIS_FIELD_MAX_CAPTUREDATE_LEN = 8;
```

**File**

SwelioEngine (see page 162)

## 1.4.24 SIS_FIELD_MAX_CARDNUMBER_LEN Constant

Maximum length of the car number field

**Pascal**

```
SIS_FIELD_MAX_CARDNUMBER_LEN = 10;
```

**File**

SwelioEngine (see page 162)

## 1.4.25 SIS_FIELD_MAX_SOCIAL_SECURITY_NUMBER_LEN Constant

Maximum length of the social security number field

**Pascal**

```
SIS_FIELD_MAX_SOCIAL_SECURITY_NUMBER_LEN = 11;
```

**File**

SwelioEngine (see page 162)

## 1.4.26 SIS_FIELD_MAX_VALIDBEGIN_LEN Constant

Maximum length of the start validity date field

**Pascal**

```
SIS_FIELD_MAX_VALIDBEGIN_LEN = 8;
```

**File**

SwelioEngine (see page 162)

## 1.4.27 SIS_FIELD_MAX_VALIDEND_LEN Constant

Maximum length of the end validity date field

**Pascal**

```
SIS_FIELD_MAX_VALIDEND_LEN = 8;
```

**File**

SwelioEngine (see page 162)

## 1.4.28 SIS_MAX_CARDNAME_LEN Constant

Maximum length of the card name field

**Pascal**

```
SIS_MAX_CARDNAME_LEN = 6;
```

**File**

SwelioEngine (see page 162)

## 1.4.29 SIS_MAX_FIRSTNAMES_LEN Constant

Maximum length of the first name field

**Pascal**

```
SIS_MAX_FIRSTNAMES_LEN = 24;
```

**File**

SwelioEngine (see page 162)

## 1.4.30 SIS_MAX_INITIAL_LEN Constant

Maximum length of the initial field

**Pascal**

```
SIS_MAX_INITIAL_LEN = 1;
```

**File**

SwelioEngine (see page 162)

## 1.4.31 SIS_MAX_NAME_LEN Constant

Maximum length of the surname field

**Pascal**

```
SIS_MAX_NAME_LEN = 48;
```

**File**

SwelioEngine (see page 162)

## 1.4.32 SIS_MAX_SEX_LEN Constant

Maximum length of the sex field

**Pascal**

```
SIS_MAX_SEX_LEN = 1;
```

**File**

SwelioEngine (see page 162)

# 1.5 Files

The following table lists files in this documentation.

**Units**

| Name | Description |
|---|---|
| SwelioEngine.pas (see page 162) | |

## 1.5.1 SwelioEngine.pas

**Constants**

| Name | Description |
|---|---|
| EID_MAX_BIRTHDATE_LEN (see page 154) | Maximum length of the birthdate |
| EID_MAX_BIRTHPLACE_LEN (see page 154) | Maximum length of the birthplace |
| EID_MAX_CARD_NUMBER_LEN (see page 155) | Maximum length of the card number field |
| EID_MAX_CERT_LEN (see page 155) | Maximum length of the certificate data |
| EID_MAX_CHIP_NUMBER_LEN (see page 155) | Maximum length of the chip number field |
| EID_MAX_DATE_BEGIN_LEN (see page 155) | Maximum length of the begin date field |
| EID_MAX_DATE_END_LEN (see page 156) | Maximum length of the end date field |
| EID_MAX_DELIVERY_MUNICIPALITY_LEN (see page 156) | Maximum length of the name of the devivery municipality |
| EID_MAX_DOCUMENT_TYPE_LEN (see page 156) | Maximum length of the document type field |
| EID_MAX_FIRST_NAME1_LEN (see page 156) | Maximum length of the first name |
| EID_MAX_FIRST_NAME2_LEN (see page 156) | Maximum length of the first name |
| EID_MAX_MUNICIPALITY_LEN (see page 157) | Maximum length of the municipality name field |
| EID_MAX_NAME_LEN (see page 157) | Maximum length of the surname |
| EID_MAX_NATIONAL_NUMBER_LEN (see page 157) | Maximum length of the national number |
| EID_MAX_NATIONALITY_LEN (see page 157) | Maximum length of the nationality |
| EID_MAX_NOBLE_CONDITION_LEN (see page 158) | Maximum length of the noble condition field |
| EID_MAX_PICTURE_LEN (see page 158) | Maximum length of the picture data |
| EID_MAX_SEX_LEN (see page 158) | Maximum length of the sex field |
| EID_MAX_SPECIAL_STATUS_LEN (see page 158) | Maximum length of the special status field |
| EID_MAX_STREET_LEN (see page 158) | Maximum length of the street name field |
| EID_MAX_ZIP_LEN (see page 159) | Maximum length of the ZIP code field |
| SIS_FIELD_MAX_BIRTHDATE_LEN (see page 159) | Maximum length of the birth date field |
| SIS_FIELD_MAX_CAPTUREDATE_LEN (see page 159) | Maximum length of the capture date field |
| SIS_FIELD_MAX_CARDNUMBER_LEN (see page 159) | Maximum length of the car number field |
| SIS_FIELD_MAX_SOCIAL_SECURITY_NUMBER_LEN (see page 160) | Maximum length of the social security number field |
| SIS_FIELD_MAX_VALIDBEGIN_LEN (see page 160) | Maximum length of the start validity date field |
| SIS_FIELD_MAX_VALIDEND_LEN (see page 160) | Maximum length of the end validity date field |

| | |
|---|---|
| SIS_MAX_CARDNAME_LEN (see page 160) | Maximum length of the card name field |
| SIS_MAX_FIRSTNAMES_LEN (see page 161) | Maximum length of the first name field |
| SIS_MAX_INITIAL_LEN (see page 161) | Maximum length of the initial field |
| SIS_MAX_NAME_LEN (see page 161) | Maximum length of the surname field |
| SIS_MAX_SEX_LEN (see page 161) | Maximum length of the sex field |

**Enumerations**

| | Name | Description |
|---|---|---|
| | TCardEventType (see page 147) | The type of the reader event |

**Functions**

| | Name | Description |
|---|---|---|
| | ActivateCard (see page 10) | Established communication between the card and the reader |
| | ActivateCardEx (see page 11) | Established communication between the card and the reader |
| | AddRemoveMessageFilter (see page 11) | Adds or removes a message from the User Interface Privilege Isolation (UIPI) message filter. |
| | AllocateBuffer (see page 12) | Allocates the buffer in memory |
| | AllocateDefaultHWND (see page 12) | This function creates the invisible tool window |
| | AllocateDefaultHWNDA (see page 12) | This function creates the invisible tool window |
| | AllocateDefaultHWNDW (see page 13) | This function creates the invisible tool window |
| | AllocateHWND (see page 13) | This function creates the invisible tool window using the provided window procedure |
| | AllocateHWNDA (see page 13) | This function creates the invisible tool window using the provided window procedure |
| | AllocateHWNDW (see page 14) | This function creates the invisible tool window using the provided window procedure |
| | AllocateLayeredWindow (see page 14) | This function creates the layered window using the provided window class name |
| | AllocateLayeredWindowA (see page 14) | This function creates the layered window using the provided window class name |
| | AllocateLayeredWindowW (see page 15) | This function creates the layered window using the provided window class name |
| | AllocateWindowClass (see page 15) | This function creates the standard window using the provided window class name |
| | AllocateWindowClassA (see page 15) | This function creates the standard window using the provided window class name |
| | AllocateWindowClassW (see page 16) | This function creates the standard window using the provided window class name |
| | BringWindowToFront (see page 16) | This function brings the specified window to the top of the z-order. |
| | CardDecryptFile (see page 16) | Decrypt file using Belgian Id card |
| | CardDecryptFileA (see page 17) | Decrypt file using Belgian Id card |
| | CardDecryptFileW (see page 17) | Decrypt file using Belgian Id card |
| | CardEncryptFile (see page 17) | Encrypt file using Belgian Id card |
| | CardEncryptFileA (see page 18) | Encrypt file using Belgian Id card |
| | CardEncryptFileW (see page 18) | Encrypt file using Belgian Id card |
| | CardSignCadesT (see page 18) | Sign data with eID card according to CADES-T standard |
| | CardSignCMS (see page 19) | Sign data with eID card according to CMS standard |
| | CertSignCadesT (see page 19) | Sign data with certificate according to CADES-T standard |
| | CertSignCMS (see page 20) | Sign data with certificate according to CMS standard |

| | | |
|---|---|---|
| | CheckMD5 (see page 20) | Checks the MD5 hash value of the memory buffer |
| | CheckSHA1 (see page 21) | Checks the SHA1 hash value of the memory buffer |
| | ClearFileAttributes (see page 21) | This function sets the file attributes to normal. |
| | ClearFileAttributesA (see page 22) | This function sets the file attributes to normal. |
| | ClearFileAttributesW (see page 22) | This function sets the file attributes to normal. |
| | ClearUnusedMemory (see page 22) | Clears unused memory and minimized the application memory usage |
| | CreateUnicodeFile (see page 23) | Creates UNICODE file |
| | CreateUnicodeFileA (see page 23) | Creates UNICODE file |
| | CreateUnicodeFileW (see page 23) | Creates UNICODE file |
| | CurrentIPAddress (see page 24) | Returns the IP address |
| | CurrentIPAddressA (see page 24) | Returns the IP address |
| | CurrentIPAddressW (see page 24) | Returns the IP address |
| | DeactivateCard (see page 24) | Terminates a cennection between a smart card and a reader |
| | DeactivateCardEx (see page 25) | Terminates a connection between a smart card and a reader |
| | DeallocateBuffer (see page 25) | Deallocates the memory buffer |
| | DeallocateHWND (see page 25) | This function destroys the specified window. |
| | DeallocateHWNDA (see page 26) | This function destroys the specified window. |
| | DeallocateHWNDW (see page 26) | This function destroys the specified window. |
| | DecryptFileAES (see page 26) | Decrypts file using AES algorithm. |
| | DecryptFileAESA (see page 27) | Decrypts file using AES algorithm. |
| | DecryptFileAESW (see page 27) | Decrypts file using AES algorithm. |
| | DeleteToRecycleBin (see page 28) | Deletes file to WIndows Recycle Bin |
| | DeleteToRecycleBinA (see page 28) | Deletes file to the Windows Recycle Bin |
| | DeleteToRecycleBinW (see page 28) | Deletes file to the Windows Recycle Bin |
| | DestroyImageBuffer (see page 29) | Destroys the memory buffer |
| | DirectoryExists (see page 29) | Determines whether a specified directory exists. |
| | DirectoryExistsA (see page 30) | Determines whether a specified directory exists. |
| | DirectoryExistsW (see page 30) | Determines whether a specified directory exists. |
| | DisplayCertificate (see page 30) | Displays the dialog window with certificate information |
| | DocumentTypeToString (see page 31) | Returns the textual representation of the card type (in English) |
| | DrawLayeredWindow (see page 31) | Repaints the surface of the layered window |
| | EmptyRecycleBin (see page 32) | Empties the recycle bin |
| | EncodeCertificate (see page 32) | Performs Base64 encoding of the certificate |
| | EncodePhoto (see page 32) | Performs Base64 encoding of the photo |
| | EncryptFileAES (see page 33) | Encrypts file using AES algorithm. |
| | EncryptFileAESA (see page 33) | Encrypts file using AES algorithm. |
| | EncryptFileAESW (see page 34) | Encrypts file using AES algorithm. |
| | FileClose (see page 34) | Concludes input/output (I/O) to a file opened using the FileCreateRewrite (see page 36) function. |

| | | |
|---|---|---|
| | FileCloseA (see page 34) | Concludes input/output (I/O) to a file opened using the FileCreateRewrite (see page 36) function. |
| | FileCloseW (see page 35) | Concludes input/output (I/O) to a file opened using the FileCreateRewrite (see page 36) function. |
| | FileCopy (see page 35) | The CopyFile function copies an existing file to a new file. |
| | FileCopyA (see page 36) | The CopyFile function copies an existing file to a new file. |
| | FileCopyW (see page 36) | The CopyFile function copies an existing file to a new file. |
| | FileCreateRewrite (see page 36) | Creates new or overwrites existing file |
| | FileCreateRewriteA (see page 37) | Creates new or overwrites existing file |
| | FileCreateRewriteW (see page 37) | Creates new or overwrites existing file |
| | FileDelete (see page 37) | Deletes a file from disk. |
| | FileDeleteA (see page 38) | Deletes a file from disk. |
| | FileDeleteW (see page 38) | Deletes a file from disk. |
| | FileExists (see page 38) | Tests whether a specified file exists. |
| | FileExistsA (see page 39) | Tests whether a specified file exists. |
| | FileExistsW (see page 39) | Tests whether a specified file exists. |
| | FileExtensionIs (see page 39) | Checks the file extension |
| | FileExtensionIsA (see page 40) | Checks the file extension |
| | FileExtensionIsW (see page 40) | Checks the file extension |
| | FileGetSize (see page 41) | Retrieves the size of a specified file. |
| | FileGetSizeA (see page 41) | Retrieves the size of a specified file. |
| | FileGetSizeW (see page 42) | Retrieves the size of a specified file. |
| | FileIsExe (see page 42) | Checks if the file is a Windows executable |
| | FileIsExeA (see page 42) | Checks if the file is a Windows executable |
| | FileIsExeW (see page 43) | Checks if the file is a Windows executable |
| | FileIsIcon (see page 43) | Checks if the file is a Windows icon (.ico) file |
| | FileIsIconA (see page 43) | Checks if the file is a Windows icon (.ico) file |
| | FileIsIconW (see page 44) | Checks if the file is a Windows icon (.ico) file |
| | FileIsImage (see page 44) | Checks if the file is an image file |
| | FileIsImageA (see page 44) | Checks if the file is an image file |
| | FileIsImageW (see page 45) | Checks if the file is an image file |
| | FileOrFolderExists (see page 45) | Checks if the file or folder with the given name exists |
| | FileOrFolderExistsA (see page 46) | Checks if the file or folder with the given name exists |
| | FileOrFolderExistsW (see page 46) | Checks if the file or folder with the given name exists |
| | FileRename (see page 46) | Renames the file |
| | FileRenameA (see page 47) | Renames the file |
| | FileRenameW (see page 47) | Renames the file |
| | FileWrite (see page 47) | Writes string to the file |
| | FileWriteA (see page 48) | Writes string to the file |
| | FileWriteChar (see page 48) | Writes one character to the file |
| | FileWriteCharA (see page 48) | Writes one character to the file |
| | FileWriteCharW (see page 49) | Writes one character to the file |
| | FileWriteNewLine (see page 49) | Writes new line sequence to the file |
| | FileWriteNewLineA (see page 49) | Writes new line sequence to the file |
| | FileWriteNewLineW (see page 49) | Writes new line sequence to the file |
| | FileWriteW (see page 50) | Writes string to the file |

| | | |
|---|---|---|
| | FormatCardNumber (see page 50) | Format card number string for better visualization |
| | FormatEIDDate (see page 50) | Converts the national number value to its formatted String representation |
| | FormatNationalNumber (see page 51) | Format the national number string for better visualization |
| | FullPath (see page 51) | Gets the full path to the file based on file name |
| | FullPathA (see page 51) | Gets the full path to the file based on file name |
| | FullPathW (see page 52) | Gets the full path to the file based on file name |
| | GenerateAuthenticationSignature (see page 52) | Generate authentication signature |
| | GenerateAuthenticationSignatureA (see page 53) | Generate authentication signature |
| | GenerateAuthenticationSignatureEx (see page 53) | Generate authentication signature |
| | GenerateAuthenticationSignatureExA (see page 54) | Generate authentication signature |
| | GenerateAuthenticationSignatureExW (see page 54) | Generate authentication signature |
| | GenerateAuthenticationSignatureW (see page 55) | Generate authentication signature |
| | GenerateBMP (see page 55) | Generates Windows Bitmap file with QR Code image |
| | GenerateBMPA (see page 56) | Generates Windows Bitmap file with QR Code image |
| | GenerateBMPW (see page 56) | Generates Windows Bitmap file with QR Code image |
| | GenerateNonRepudiationSignature (see page 56) | Generate non repudiation signature |
| | GenerateNonRepudiationSignatureA (see page 57) | Generate non repudiation signature |
| | GenerateNonRepudiationSignatureEx (see page 57) | Generate non repudiation signature |
| | GenerateNonRepudiationSignatureExA (see page 58) | Generate non repudiation signature |
| | GenerateNonRepudiationSignatureExW (see page 59) | Generate non repudiation signature |
| | GenerateNonRepudiationSignatureW (see page 59) | Generate non repudiation signature |
| | GeneratePNG (see page 60) | Generates PNG file with QR Code image |
| | GeneratePNGA (see page 60) | Generates PNG file with QR Code image |
| | GeneratePNGW (see page 60) | Generates PNG file with QR Code image |
| | GenerateQRCode (see page 61) | Read eID card and save the identity information and address to PNG QR Code file |
| | GenerateQRCodeA (see page 61) | Read eID card and save the identity information and address to PNG QR Code file |
| | GenerateQRCodeEx (see page 62) | Read eID card and save the identity information and address to PNG QR Code file |
| | GenerateQRCodeExA (see page 62) | Read eID card and save the identity information and address to PNG QR Code file |
| | GenerateQRCodeExW (see page 63) | Read eID card and save the identity information and address to PNG QR Code file |
| | GenerateQRCodeW (see page 63) | Read eID card and save the identity information and address to PNG QR Code file |
| | GetCardSerialNumber (see page 63) | Gets the card serial number |
| | GetEncodedCertificateSize (see page 64) | Returns the size of the Base64 encoded certificate |

| | | |
|---|---|---|
| | GetEncodedPhotoSize (see page 64) | Calculates buffer size for Base64 encoded photo |
| | GetFileMD5 (see page 65) | Gets the MD5 hash value for the file |
| | GetFileMD5A (see page 65) | Gets the MD5 hash value for the file |
| | GetFileMD5W (see page 65) | Gets the MD5 hash value for the file |
| | GetFilesCount (see page 66) | Calculates the number of files in the given folder |
| | GetFilesCountA (see page 66) | Calculates the number of files in the given folder |
| | GetFilesCountW (see page 67) | Calculates the number of files in the given folder |
| | GetFileSHA1 (see page 67) | Gets the SHA1 hash value for the file |
| | GetFileSHA1A (see page 67) | Gets the SHA1 hash value for the file |
| | GetFileSHA1W (see page 68) | Gets the SHA1 hash value for the file |
| | GetHBitmap (see page 68) | Generates Windows Bitmap in memory with QR Code image |
| | GetHBitmapA (see page 69) | Generates Windows Bitmap in memory with QR Code image |
| | GetHBitmapW (see page 69) | Generates Windows Bitmap in memory with QR Code image |
| | GetISOCode (see page 70) | Returns the country ISO code based on the nationality string |
| | GetISOCodeA (see page 70) | Returns the country ISO code based on the nationality string |
| | GetISOCodeW (see page 71) | Returns the country ISO code based on the nationality string |
| | GetMD5 (see page 71) | Gets the MD5 hash value for the content of the memory buffer |
| | GetPNG (see page 72) | Writes PNG image to the memory buffer. |
| | GetPNGA (see page 72) | Writes PNG image to the memory buffer. |
| | GetPNGW (see page 73) | Writes PNG image to the memory buffer. |
| | GetReaderIndex (see page 73) | Returns the zero-based reader index with specified name |
| | GetReaderIndexA (see page 73) | Returns the zero-based reader index with specified name |
| | GetReaderIndexW (see page 74) | Returns the zero-based reader index with specified name |
| | GetReaderName (see page 74) | Returns the name of the reader |
| | GetReaderNameA (see page 75) | Returns the name of the reader |
| | GetReaderNameLen (see page 75) | Returns the length of the reader name |
| | GetReaderNameLenA (see page 75) | Returns the length of the reader name |
| | GetReaderNameLenW (see page 76) | Returns the length of the reader name |
| | GetReaderNameW (see page 76) | Returns the name of the reader |
| | GetReadersCount (see page 76) | Get number of card readers connected to PC |
| | GetSelectedReaderIndex (see page 77) | Returns the index of the active smart card reader |
| | GetSHA1 (see page 77) | Gets the SHA1 hash value for the content of the memory buffer |
| | GetStartup (see page 78) | Checks if the application is registered to run when Windows starts |
| | GetStartupA (see page 78) | Checks if the application is registered to run when Windows starts |
| | GetStartupW (see page 78) | Checks if the application is registered to run when Windows starts |
| | GetSupportSIS (see page 78) | Checks if the SIS cards are supported by the engine |
| | HibernateWindows (see page 79) | Hibernates Windows |
| | IsAnimatedGIF (see page 79) | Checks if the file is an animated GIF image file |
| | IsAnimatedGIFA (see page 79) | Checks if the file is an animated GIF image file |
| | IsAnimatedGIFW (see page 80) | Checks if the file is an animated GIF image file |
| | IsCardPresent (see page 80) | Checks if the card is present in the card reader |
| | IsCardPresentEx (see page 80) | Checks if the card is present in the card reader |
| | IsConnectedToInternet (see page 81) | Checks if PC is connected to Internet |
| | IsDirectory (see page 81) | Verifies that a path is a valid directory. |
| | IsDirectoryA (see page 81) | Verifies that a path is a valid directory. |
| | IsDirectoryW (see page 82) | Verifies that a path is a valid directory. |
| | IsEIDCard (see page 82) | Check if Belgian EID card is inserted into card reader |
| | IsEIDCardEx (see page 82) | Check if Belgian EID card is inserted into card reader |

| | | |
|---|---|---|
| | IsEngineActive (see page 83) | Checks if the Swelio Engine is activated |
| | IsFemale (see page 83) | Checks if the card owner is female |
| | IsFemaleA (see page 83) | Checks if the card owner is female |
| | IsFemaleW (see page 84) | Checks if the card owner is female |
| | IsMale (see page 84) | Checks if the card owner is male |
| | IsMaleA (see page 85) | Checks if the card owner is male |
| | IsMaleW (see page 85) | Checks if the card owner is male |
| | IsMediaCenter (see page 85) | Checks if the Media Center version of Windows is installed |
| | IsMetroActive (see page 86) | Checks if metro interface is active |
| | IsMultiTouchReady (see page 86) | Checks if the system is multi touch ready |
| | IsNativeWin64 (see page 86) | Checks if the application is native 64 bit executable |
| | IsSISCard (see page 86) | Check if Belgian SIS card is inserted into card reader |
| | IsSISCardEx (see page 87) | Check if Belgian SIS card is inserted into card reader |
| | IsTabletPC (see page 87) | Checks if the application is running on the Tablet PC |
| | IsUnicodeFile (see page 87) | Checks if the file is UNICODE file |
| | IsUnicodeFileA (see page 88) | Checks if the file is UNICODE file |
| | IsUnicodeFileW (see page 88) | Checks if the file is UNICODE file |
| | IsValidFileName (see page 88) | Checks if provided string is a valid file name |
| | IsValidFileNameA (see page 89) | Checks if provided string is a valid file name |
| | IsValidFileNameW (see page 89) | Checks if provided string is a valid file name |
| | IsValidPathName (see page 90) | Checks if provided string is a valid file path |
| | IsValidPathNameA (see page 90) | Checks if provided string is a valid file path |
| | IsValidPathNameW (see page 90) | Checks if provided string is a valid file path |
| | IsWindows7 (see page 91) | Checks if PC is running Windows 7 or better |
| | IsWindows8 (see page 91) | Checks if PC is Running Windows 8 or better |
| | IsWindowsVista (see page 91) | Checks if PC is running Windows Vista or better |
| | IsWindowsXP (see page 91) | Checks if PC is running Windows XP |
| | IsWindowsXPSP2 (see page 92) | Checks if PC is running Windows XP with Service Pack 2 installed |
| | IsWow64 (see page 92) | Checks if the 32 bit application runs on 64 bit Windows |
| | LayeredWndProc (see page 92) | The default window procedure for the layered window |
| | LayeredWndProcA (see page 92) | The default window procedure for the layered window |
| | LayeredWndProcW (see page 93) | The default window procedure for the layered window |
| | LoadCertificate (see page 93) | Reads the certificate from a file |
| | LoadCertificateA (see page 93) | Reads the certificate from a file |
| | LoadCertificateW (see page 94) | Reads the certificate from a file |
| | LoadIdentity (see page 94) | Reads the raw identity information from a file |
| | LoadIdentityA (see page 94) | Reads the raw identity information from a file |
| | LoadIdentityW (see page 95) | Reads the raw identity information from a file |
| | LoadPhoto (see page 95) | Loads photo from a file |
| | LoadPhotoA (see page 95) | Loads photo from a file |
| | LoadPhotoW (see page 96) | Loads photo from a file |
| | MakeSoundFromFile (see page 96) | Plays the wave sound from the file |
| | MakeSoundFromFileA (see page 96) | Plays the wave sound from the file |
| | MakeSoundFromFileW (see page 97) | Plays the wave sound from the file |
| | MakeSoundFromResource (see page 97) | Plays the wave sound from the resource |
| | MakeSoundFromResourceA (see page 97) | Plays the wave sound from the resource |
| | MakeSoundFromResourceW (see page 98) | Plays the wave sound from the resource |

| | | |
|---|---|---|
| | PortAvailable (see page 98) | Checks if the port with specified number is available |
| | ReadAddress (see page 98) | Read address information from Belgian eID card |
| | ReadAddressA (see page 99) | Read address information from Belgian eID card |
| | ReadAddressEx (see page 99) | Read address information from Belgian eID card |
| | ReadAddressExA (see page 100) | Read address information from Belgian eID card |
| | ReadAddressExW (see page 100) | Read address information from Belgian eID card |
| | ReadAddressW (see page 100) | Read address information from Belgian eID card |
| | ReadAuthenticationCertificate (see page 101) | Read Authentication Certificate to memory |
| | ReadBufferFromFile (see page 101) | Reads the content of the file to the memory buffer |
| | ReadBufferFromFileA (see page 101) | Reads the content of the file to the memory buffer |
| | ReadBufferFromFileW (see page 102) | Reads the content of the file to the memory buffer |
| | ReadCaCertificate (see page 102) | Read Ca Certificate to memory |
| | ReadIdentity (see page 103) | Read identity information from Belgian eID card |
| | ReadIdentityA (see page 103) | Read identity information from Belgian eID card |
| | ReadIdentityEx (see page 103) | Read identity information from Belgian eID card |
| | ReadIdentityExA (see page 104) | Read identity information from Belgian eID card |
| | ReadIdentityExW (see page 104) | Read identity information from Belgian eID card |
| | ReadIdentityW (see page 104) | Read identity information from Belgian eID card |
| | ReadNonRepudiationCertificate (see page 105) | Read Non Repudiation Certificate to memory |
| | ReadPhoto (see page 105) | Reads a photo from a card |
| | ReadPhotoAsBitmap (see page 106) | Reads the picture from the card, converts it to bitmap and returns the bitmap handle Decription: 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 106) | Reads the picture from the card, converts it to bitmap and returns the bitmap handle Decription: 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 106) | Reads a photo from a card |
| | ReadRootCaCertificate (see page 107) | Read Root Ca Certificate to memory |
| | ReadRrnCertificate (see page 107) | Read Rrn Certificate to memory |
| | ReadSISCard (see page 107) | Read Belgian SIS card. |
| | ReadSISCardA (see page 108) | Read Belgian SIS card. |
| | ReadSISCardEx (see page 108) | Read Belgian SIS card. |
| | ReadSISCardExA (see page 109) | Read Belgian SIS card. |
| | ReadSISCardExW (see page 109) | Read Belgian SIS card. |
| | ReadSISCardW (see page 110) | Read Belgian SIS card. |
| | ReloadReadersList (see page 110) | Reloads the list of the available card readers |
| | RemoveCallback (see page 110) | Remove callback procedure for card events |
| | RemoveStartup (see page 111) | Removes the application from the list of the automatically started applications |
| | RemoveStartupA (see page 111) | Removes the application from the list of the automatically started applications |
| | RemoveStartupW (see page 111) | Removes the application from the list of the automatically started applications |
| | RestoreWindowSubclass (see page 112) | Restores window standard procedure |
| | RestoreWindowSubclassA (see page 112) | Restores window standard procedure |

| | | |
|---|---|---|
| | RestoreWindowSubclassW (see page 112) | Restores window standard procedure |
| | SaveAuthenticationCertificate (see page 112) | Save Authentication Certificate to a file |
| | SaveAuthenticationCertificateA (see page 113) | Save Authentication Certificate to a file |
| | SaveAuthenticationCertificateW (see page 113) | Save Authentication Certificate to a file |
| | SaveCaCertificate (see page 113) | Save Ca Certificate to a file |
| | SaveCaCertificateA (see page 114) | Save Ca Certificate to a file |
| | SaveCaCertificateW (see page 114) | Save Ca Certificate to a file |
| | SaveCardToXml (see page 115) | Read eID card and save the information to XML file |
| | SaveCardToXmlA (see page 115) | Read eID card and save the information to XML file |
| | SaveCardToXmlEx (see page 115) | Read eID card and save the information to XML file |
| | SaveCardToXmlExA (see page 116) | Read eID card and save the information to XML file |
| | SaveCardToXmlExW (see page 116) | Read eID card and save the information to XML file |
| | SaveCardToXmlW (see page 117) | Read eID card and save the information to XML file |
| | SaveIdentity (see page 117) | Saves indentity infornation to a file |
| | SaveIdentityA (see page 117) | Saves indentity infornation to a file |
| | SaveIdentityW (see page 118) | Saves indentity infornation to a file |
| | SaveNonRepudiationCertificate (see page 118) | Save Non Repudiation Certificate to a file |
| | SaveNonRepudiationCertificateA (see page 118) | Save Non Repudiation Certificate to a file |
| | SaveNonRepudiationCertificateW (see page 119) | Save Non Repudiation Certificate to a file |
| | SavePersonToCsv (see page 119) | Read eID card and save the identity information and address to CSV file |
| | SavePersonToCsvA (see page 120) | Read eID card and save the identity information and address to CSV file |
| | SavePersonToCsvEx (see page 120) | Read eID card and save the identity information and address to CSV file |
| | SavePersonToCsvExA (see page 120) | Read eID card and save the identity information and address to CSV file |
| | SavePersonToCsvExW (see page 121) | Read eID card and save the identity information and address to CSV file |
| | SavePersonToCsvW (see page 121) | Read eID card and save the identity information and address to CSV file |
| | SavePhoto (see page 122) | Saves photo to a file |
| | SavePhotoA (see page 122) | Saves photo to a file |
| | SavePhotoAsBitmap (see page 122) | 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. |
| | SavePhotoAsBitmapA (see page 123) | 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. |
| | SavePhotoAsBitmapEx (see page 123) | 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. |

| | | |
|---|---|---|
| | SavePhotoAsBitmapExA (see page 123) | 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 124) | 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 124) | 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. |
| | SavePhotoAsJpeg (see page 125) | 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. |
| | SavePhotoAsJpegA (see page 125) | 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. |
| | SavePhotoAsJpegEx (see page 125) | 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 126) | 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 126) | 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. |
| | SavePhotoAsJpegW (see page 127) | 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. |
| | SavePhotoW (see page 127) | Saves photo to a file |
| | SaveRootCaCertificate (see page 127) | Save Root Ca Certificate to a file |
| | SaveRootCaCertificateA (see page 128) | Save Root Ca Certificate to a file |
| | SaveRootCaCertificateW (see page 128) | Save Root Ca Certificate to a file |
| | SaveRrnCertificate (see page 128) | Save RRN Certificate to a file |
| | SaveRrnCertificateA (see page 129) | Save RRN Certificate to a file |
| | SaveRrnCertificateW (see page 129) | Save RRN Certificate to a file |
| | SelectReader (see page 129) | When more than 1 reader connected, select the reader with specified number The first reader has number 0 |
| | SelectReaderByName (see page 130) | Select active smart card reader by providing the reader name |
| | SelectReaderByNameA (see page 130) | Select active smart card reader by providing the reader name |
| | SelectReaderByNameW (see page 131) | Select active smart card reader by providing the reader name |
| | SetCallback (see page 131) | Activates callback procedure for card status change event |
| | SetMWCompatibility (see page 131) | Set the compatibility mode with the old version of the oficial EID MiddleWare |
| | SetStartup (see page 132) | Register application to run when Windows starts |

| | | |
|---|---|---|
| | SetStartupA (see page 132) | Register application to run when Windows starts |
| | SetStartupW (see page 132) | Register application to run when Windows starts |
| | SetSupportSIS (see page 133) | Activates or deactivates SIS card support by engine |
| | ShellCopyFile (see page 133) | Copies file to the new location |
| | ShellCopyFileA (see page 133) | Copies file to the new location |
| | ShellCopyFileW (see page 134) | Copies file to the new location |
| | ShowError (see page 134) | Shows Dialog with the text message corresponding to the Windows error code |
| | ShutdownWindows (see page 134) | Logs off the interactive user, shuts down the system. |
| | StartEngine (see page 135) | Activates the Swelio Engine. |
| | StopEngine (see page 135) | Deactivates the Swelio Engine |
| | StripFileName (see page 135) | Replaces environment variable names with values |
| | StripFileNameA (see page 136) | Replaces environment variable names with values |
| | StripFileNameW (see page 136) | Replaces environment variable names with values |
| | SuspendWindows (see page 137) | Suspends Windows |
| | TurnMonitorOff (see page 137) | Turns the monitor off |
| | TurnMonitorOn (see page 137) | Turns the monitor on |
| | UpdateWindowPosition (see page 137) | Updated the window position |
| | VerifyPin (see page 138) | Verify PIN code |
| | VerifyPinA (see page 138) | Verify PIN code |
| | VerifyPinEx (see page 138) | Verify PIN code |
| | VerifyPinExA (see page 139) | Verify PIN code |
| | VerifyPinExW (see page 139) | Verify PIN code |
| | VerifyPinW (see page 139) | Verify PIN code |
| | VerifySignature (see page 140) | Verifies the signature from the specified hash value. |
| | WriteBufferToFile (see page 140) | Writes the memory buffer to file |
| | WriteBufferToFileA (see page 141) | Writes the memory buffer to file |
| | WriteBufferToFileW (see page 141) | Writes the memory buffer to file |

**Records**

| | Name | Description |
|---|---|---|
| | tagEidAddressA (see page 142) | EID address information, stored on the card |
| | tagEidAddressW (see page 142) | EID address information, stored on the card |
| | tagEidCertificate (see page 143) | Certificate, stored on EID card |
| | tagEidIdentityA (see page 143) | Identity information stored on EID card |
| | tagEidIdentityW (see page 144) | Identity information stored on EID card |
| | tagEidPicture (see page 145) | Raw picture data from EID card |
| | tagSISRecordA (see page 146) | Public information stored on Belgian SIS card. The SIS card is the social security card of each Belgian resident (Belgian or foreigner) |
| | tagSISRecordW (see page 146) | Public information stored on Belgian SIS card. The SIS card is the social security card of each Belgian resident (Belgian or foreigner) |

**Types**

| Name | Description |
|---|---|
| PEIDAddress (see page 148) | EID address information, stored on the card |
| PEIDAddressA (see page 148) | EID address information, stored on the card |
| PEIDAddressW (see page 149) | EID address information, stored on the card |
| PEIDCertificate (see page 149) | Certificate, stored on EID card |
| PEIDIdentity (see page 149) | Identity information stored on EID card |
| PEIDIdentityA (see page 149) | Identity information stored on EID card |

| | |
|---|---|
| PEIDIdentityW (see page 150) | Identity information stored on EID card |
| PEIDPicture (see page 150) | Raw picture data from EID card |
| PSISRecordA (see page 150) | Public information stored on Belgian SIS card |
| PSISRecordW (see page 150) | Public information stored on Belgian SIS card |
| TEIDAddress (see page 150) | EID address information, stored on the card |
| TEIDAddressA (see page 151) | EID address information, stored on the card |
| TEIDAddressW (see page 151) | EID address information, stored on the card |
| TEIDCertificate (see page 151) | Certificate, stored on EID card |
| TEIDIdentity (see page 151) | Identity information stored on EID card |
| TEIDIdentityA (see page 152) | Identity information stored on EID card |
| TEIDIdentityW (see page 152) | Identity information stored on EID card |
| TEIDPicture (see page 152) | Raw picture data from EID card |
| TReaderCallback (see page 152) | The smart card reader callback procedure |
| TSISRecord (see page 152) | Public information stored on Belgian SIS card |
| TSISRecordA (see page 153) | Public information stored on Belgian SIS card |
| TSISRecordW (see page 153) | Public information stored on Belgian SIS card |

# Index

## A



## B



## C

## D



## E

## F

## G

## H



## I

## L



## M



## P



## R

## S

## T

## U



## V



## W