

# BioCheck Fingerprint Search Engine

---

SDK Version 1.05.01

Copyright (C) 2003 by Biometrix Int.

## Disclaimer

Biometrix Int. does not warrant that this software will meet your requirements or that the operation of the software will be uninterrupted or error free. The warranty does not cover any media or documentation which has been subjected to damage or abuse by you or others. The software warranty does not cover any copy of the licensed software which has been altered or changed in any way.

ANY IMPLIED WARRANTIES INCLUDING ANY WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE ARE LIMITED TO THE TERM OF THE EXPRESS WARRANTIES. Some States do not allow limitations on how long an implied warranty lasts, so the above limitation may not apply to you.

The warranties set forth above are in lieu of any and all other express or implied warranties, whether oral, written, or implied, and the remedies set forth above are the sole and exclusive remedies.

Biometrix Int. is not responsible for any problems or damage caused by this software that may result from using this software. This includes, but is not limited to, computer hardware, computer software, operating systems, and any computer or computing accessories. End user agrees to hold Biometrix Int. harmless for any problems arising from the use of the software.

Biometrix Int. SHALL NOT IN ANY CASE BE LIABLE FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, INDIRECT OR OTHER SIMILAR DAMAGES ARISING FROM ANY BREACH OF THESE WARRANTIES EVEN IF Biometrix Int. OR HIS AGENTS OR DISTRIBUTORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Some states do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

In no case shall Biometrix Int. liability exceed the license fees paid for the right to use this software.

# Biometrix Int. BioCheck Fingerprint Search Engine End User License Agreement

Biometrix Int. ("Biometrix") is granting you (an individual or an entity, either of which is referred to herein as "Licensee") a license to use Biometrix Int.'s BioCheck Fingerprint Search Engine SDK, including computer software, hardware, associated media and printed materials only upon the condition that Licensee accepts all of the terms and conditions contained in this User License Agreement (the "Agreement"). If Licensee does not agree to the terms of this Agreement, Licensee should not install or use the BioCheck Fingerprint Search Engine SDK, rather, promptly return the BioCheck Fingerprint Search Engine SDK to the place of purchase.

## 1. Grant of License.

This Agreement grants Licensee a personal or company, limited, non-transferable, non-exclusive right to install and use one copy of the BioCheck Fingerprint Search Engine SDK Developer's Kit (SDK) on a single computer exclusively for the following purposes:

- a. Licensee may use the BioCheck Fingerprint Search Engine SDK solely for developing, designing, and testing Biometrix Int. software applications for use with Biometrix Int. products ("Applications").
- b. Licensee may modify the sample source code located in the BioCheck Fingerprint Search Engine SDK "samples" directories ("Sample Code") to design, develop and test Biometrix Int. Applications. Licensee may also reproduce and distribute the Sample Code in object code form along with any modifications Licensee makes to the Sample Code, provided that Licensee complies with the distribution requirements described below. For purposes of this section, "modifications" shall mean enhancements to the functionality of the Sample Code.
- c. Licensee may copy and redistribute the Sample Code provided that:
  - (a) it is distributed as part of an Application prepared by Licensee;
  - (b) Licensee's Application adds significant and primary functionality to the Sample Code;
  - (c) the Sample Code only operates in conjunction with the BioCheck Fingerprint Search Engine SDK;
  - (d) Licensee does not permit further redistribution of the Sample Code;
  - (e) Licensee does not use Biometrix Int.'s name, logo or trademarks to mark Licensee's Application;
  - (f) Licensee includes a valid copyright notice on Licensee's Application; and
  - (g) Licensee agrees to indemnify, hold harmless, and defend Biometrix Int. from and against any claims or lawsuits, including attorney's fees, that arise or result from the use or distribution of Licensee's Application.
- d. No other uses and/or distribution of the BioCheck Fingerprint Search Engine SDK or Sample Code are permitted without Biometrix Int.'s prior written consent. Biometrix Int. reserves all rights not expressly granted to Licensee.

## 2. Limitations or Restrictions.

Licensee may not: (i) modify or translate the BioCheck Fingerprint Search Engine SDK; (ii) reverse engineer, decompile, or disassemble the BioCheck Fingerprint Search Engine SDK, except and only to the extent this restriction is expressly permitted by applicable law; (iii) separate the BioCheck Fingerprint Search Engine SDK component parts to transfer to a third party; (iv) rent, lease, sell or lend the BioCheck Fingerprint Search Engine SDK.

## 3. Intellectual Property Rights.

Licensee acknowledges that no title to the intellectual property in the BioCheck Fingerprint Search Engine SDK and any components thereof are transferred to Licensee. All rights, title and copyrights in and to the BioCheck Fingerprint Search Engine SDK and any copies are owned by Biometrix Int. The BioCheck Fingerprint Search Engine SDK are protected by copyright laws and international copyright conventions and treaties.

LICENSEE'S EXCLUSIVE REMEDY UNDER THIS AGREEMENT SHALL NOT EXCEED THE REPLACEMENT COST OF A SINGLE COPY OF THE BIOCHECK FINGERPRINT SEARCH ENGINE SDK .

## 4. Warranty.

Biometrix Int.'s BioCheck Fingerprint Search Engine SDK warranty runs to the Licensee for a period of thirty (30) days. If the date of receipt by Licensee of BioCheck Fingerprint Search Engine SDK is the same as the date of purchase by Licensee of the BioCheck Fingerprint Search Engine SDK, the thirty-day warranty period begins on the date of purchase. If the date of receipt by Licensee of BioCheck Fingerprint Search Engine SDK is later than the date of purchase of the BioCheck Fingerprint Search Engine SDK by Licensee, then the thirty-day warranty period begins on the date of shipment by a distributor of the BioCheck Fingerprint Search Engine SDK to Licensee. During the thirty day warranty period, Biometrix Int. warrants that (a) the media on which the BioCheck Fingerprint Search Engine SDK or documentation covering BioCheck Fingerprint Search Engine SDK is delivered shall be free from material defects in workmanship and materials, and (b) the BioCheck Fingerprint Search Engine SDK, when installed, configured, used and maintained in accordance with Biometrix Int.'s then-current published installation, configuration, use and maintenance specifications, will, in its unaltered form, conform substantially to Biometrix Int.'s then-current published functional specifications for such BioCheck Fingerprint Search Engine SDK. Biometrix Int.'s sole obligation, for a breach of this warranty shall be for Biometrix Int. to replace the media in the case of breach of this Warranty. Biometrix Int. does not warrant that the BioCheck Fingerprint Search Engine SDK will meet Licensee's requirements, that the BioCheck Fingerprint Search Engine SDK will operate in combinations selected for use by Licensee or that use of the BioCheck Fingerprint Search Engine SDK will be uninterrupted or error-free. Because not all errors in software can or need be corrected, Biometrix Int. does not warrant that the BioCheck Fingerprint Search Engine SDK are error-free or that all software errors will be corrected.

## 5. LIMITATIONS OF WARRANTY.

THE BIOCHECK FINGERPRINT SEARCH ENGINE SDK IS DEEMED ACCEPTED BY LICENSEE. BIOMETRIX INT. DISCLAIMS ALL WARRANTIES NOT EXPRESSLY PROVIDED IN THIS AGREEMENT INCLUDING, WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. ANY AND ALL RISK ARISING OUT OF THE USE OR PERFORMANCE OF THE BIOCHECK BIOCHECK FINGERPRINT SEARCH ENGINE SDK REMAINS WITH LICENSEE. IN NO EVENT SHALL BIOMETRIX INT. OR ITS SUPPLIERS BE LIABLE FOR ANY CONSEQUENTIAL, INCIDENTAL, DIRECT, INDIRECT, SPECIAL, PUNITIVE, OR OTHER DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR OTHER PECUNIARY LOSS) ARISING OUT OF THIS AGREEMENT OR THE USE OF OR INABILITY TO USE THE BIOCHECK FINGERPRINT SEARCH ENGINE SDK , EVEN IF BIOMETRIX INT. HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME STATES/JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATION APPLIES ONLY TO THE EXTENT PERMITTED BY APPLICABLE LAW.

## 6. LIMITATION OF LIABILITY.

BIOMETRIX INT.'S TOTAL LIABILITY AND LICENSEE'S EXCLUSIVE REMEDY UNDER THIS AGREEMENT SHALL NOT EXCEED THE REPLACEMENT COST OF A SINGLE COPY OF THE BIOCHECK FINGERPRINT SEARCH ENGINE SDK .

## 7. Indemnification.

Licensee shall hold harmless, indemnify and defend Biometrix Int., its officers, directors and employees, from and against any claim, suit or proceeding and any losses, damages, fines and expenses (including attorneys' fees and costs) arising out of or relating to any claims that Licensee's use of the BioCheck Fingerprint Search Engine SDK in conjunction with any Licensee Application infringes the patent, copyright, trademark, trade secret, or other proprietary rights of any third party, or resulting from any breach of this Agreement by Licensee.

## 8. Termination.

Without prejudice to any other rights, Biometrix Int. may terminate this Agreement if Licensee fails to comply with any term or condition of this Agreement. Upon termination of this Agreement, Licensee shall immediately discontinue the use of the BioCheck Fingerprint Search Engine SDK and certify destruction of all full or partial copies of the BioCheck Fingerprint Search Engine SDK and related materials provided by Biometrix Int.

Licensee may also terminate this Agreement at any time by destroying the BioCheck Fingerprint Search Engine SDK and all copies thereof.

#### 9. General.

Licensee acknowledges that he or she has read this Agreement, understands it, and that by using the BioCheck Fingerprint Search Engine SDK Licensee agrees to be bound by the terms and conditions set forth in this Agreement. Licensee further agrees that the Agreement is the complete and exclusive statement of the understanding between Biometrix Int. and Licensee which supersedes any proposal or prior agreement, oral or written, and any other communication between Biometrix Int. and Licensee relating to the subject matter of this Agreement. This Agreement may not be modified except in writing duly signed by an authorized representative of Biometrix Int. and Licensee. If any provision of this Agreement is held to be unenforceable for any reason, such provision shall be reformed only to the extent necessary to make it enforceable, and such decision shall not affect the enforceability of such provision under other circumstances, or of the remaining provisions hereof under all circumstances.

THIS PUBLICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY MADE TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED INTO NEW EDITIONS OF THIS PUBLICATION. BIOMETRIX INT. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS PUBLICATION AT ANY TIME AND WITHOUT NOTICE.

# Part 1

## Scanner interface

**Overview:**

This part of the API contains the necessary functions to capture fingerprint images.

Usage is quite straightforward: Initialize the scanner with `CSInitReader`, get the image size with `CSGetImageSize`, then read images with `CSReadRawImage` as needed and terminate with a call to `CSCloseReader`. The only parameters you may have to adjust are gain and brightness, to allow for changes in environmental conditions such as lighting or moisture.

The demo program that comes with the SDK shows some of the possibilities.

This part of the SDK is separate from the recognition functions, because different sources of fingerprint images may be used. Again, the demo program gives an example, namely the offline enrollment and recognition of fingerprints from tiff-files as image source.

If you plan to write your own customized reader DLL and want to use the demo program as a template for your own applications, your DLL should export the functions described in this part of the document. Also, the name of the DLL should be "xxx41api.dll", xxx being an identification code of the scanner for use in the `biocheck.ini`. Otherwise, feel free to implement whatever suits the needs of your specific scanner device.

**File list:**

- biofpa41.h
- xxx41api.lib (xxx = bio, idm, son, eye, fpc, shi or non)
- xxx41api.dll
- device specific drivers and/or interface DLLs

The remainder of this document describes the functions exported by the `xxxfpapi.dll` in detail.

## CSInitReader

### Definition:

```
C++:
int __stdcall CSInitReader
(
    char *driver_path
);
```

### Visual Basic:

```
Public Declare Function CSInitReader Lib "xxx41api"
(
    ByVal driver As String
)
As Long
```

### Example:

```
Dim ret_code As Long

driver$ = "" & chr(0)
ret_code = CSInitReader
```

### Function:

Scanner device initialization

### Parameters:

driver\_path (IN)            Empty string ("").

### Return Codes:

FP\_READER\_OK (0):        successful  
other:                    Call the Win32 API GetLastError() for details

### Remarks:

This function must be called before any other scanner functions.  
Calling the function more than once does not result in an error.

## CSGetImageSize

### Definition:

```
C++:  
void __stdcall CSGetImageSize  
(  
    int *image_length,  
    int *image_width  
);
```

### Visual Basic:

```
Public Declare Sub CSGetImageSize Lib "xxx41api"  
(  
    image_length As Long,  
    image_width As Long  
)
```

### Example:

```
Dim width As Long, length As Long
```

```
CSGetImageSize length, width
```

### Function:

Returns width and length of the scanned fingerprint image.

### Parameters:

image\_width (OUT)

Pointer to an integer that will receive the width (columns) of the fingerprint image.

image\_length (OUT)

Pointer to an integer that will receive the length (rows) of the fingerprint image.

### Return Codes:

None.

## CSReadRawImage

### Definition:

```
C++:
int      __stdcall CSReadRawImage
(
    BYTE      *image,
    int       device_port,
    int       gain,
    int       brightness,
    BOOL      auto_adapt
);
```

### Visual Basic:

```
Public Declare Function CSReadRawImage Lib "xxx41api"
(
    ByVal image As Long,
    ByVal port As Long,
    ByVal gain As Long,
    ByVal brightness As Long,
    ByVal auto_adapt As Long
)
As Long
```

### Example:

```
Dim img(1 To MAX_IMAGE_SIZE) As Byte
Dim p_img As Long
Dim port As Long, gain As Long, brightness As Long
Dim ret_code As Long

p_img = VarPtr(img(1))
ret_code = CSReadRawImage(p_img, port, gain, brightness, 0)
If ret_code <> 0 Then
    MsgBox "Error reading image!", vbCritical
End If
```

### Function:

Read a fingerprint image from the scanner

### Parameters:

image (OUT)

Byte array [ image\_width \* image\_length ]. This array receives the fingerprint image from the scanner. Each byte corresponds to a pixel, with values from 0 (black) to 255 (white). Storage must be allocated and freed outside the function.

port (IN)

Addressable USB devices: 0,1,etc.

gain (IN)

Contrast. Values are scanner specific.

brightness (IN)

Brightness. Values are scanner specific.

auto\_adapt (IN)

reserved: must be 0

### Return Codes:

FP\_READER\_OK (0): successful

FP\_READER\_SETTINGS\_FAILED:

This error code usually means that the scanner is warming up and not ready. Retry during the time of 1-2 second. If the error code persists for longer there may be something amiss with the device.

FP\_READER\_NOT\_FOUND:

No scanner was found on the specified port

FP\_READER\_INSUFFICIENT\_MEMORY:

Internal memory access fault.

FP\_READER\_MEMORY\_ALLOC\_FAILURE:

Temporary image buffer allocation error.

FP\_READER\_INVALID\_BUFFER:

The image buffer passed to the function is invalid.

FP\_READER\_INVALID\_FRAME\_NUMBER:

The scanner was unable to read an image.

**Remarks:**

Gain and brightness may have to be adjusted depending on light conditions, room temperature and moisture (which may affect the texture of fingertips). Generally useful values are preset in the file biocheck.ini.

## CSGetErrorString

### Definition:

```
C++:  
void __stdcall CSGetErrorString  
(  
    int error_code,  
    char *error_text  
);
```

### Visual Basic:

```
Public Declare Sub CSGetErrorString Lib "xxx41api"  
(  
    ByVal rtc As Long,  
    ByVal err As String  
)
```

### Example:

```
Dim ret_code As Long  
Dim errstr As String  
  
errstr = String(255, " ") & Chr(0)  
CSGetErrorString ret_code, errstr  
MsgBox errstr, vbCritical
```

### Function:

Returns an error text to a given error code.

### Parameters:

error\_code (IN)  
Error code in the FP\_READER\_xxx range  
error\_text (OUT)  
Pointer to a buffer of length >= 64 that will receive a zero terminated string corresponding to the error code.

### Return Codes:

None.

### Remarks:

This function is intended mainly for debugging purposes, to provide an easy way to display meaningful error messages.

## CSCloseReader

### Definition:

**C++:**  
void \_\_stdcall CSCloseReader ();

### Visual Basic:

Public Declare Sub CSCloseReader Lib "xxx41api" ()

Example:  
CSCloseReader

### Function:

Close the scanner device.

### Parameters:

None.

### Return Codes:

None.

### Remarks:

Required by some scanner models. You should call this function before program exit.

# Part 2

## Recognition

**Overview:**

The functions described in this part of the BIOCHECK SDK perform the extraction of minutiae from raw fingerprint images, the generalization of features from different feature sets of the same print, and the matching of a fingerprint feature set with the stored generalized sets.

To prepare a fingerprint image for further processing, the function CSEExtractFeatures must be called to obtain a data set of the minutiae, or features, of the fingerprint. This data set consists of an array of values describing the location and direction of the features, and an overall classification of the fingerprint type.

For enrollment, the function CSGeneralization provides the means to summarize the features of several prints of the same finger into a more accurate, and therefore more reliable, set of data.

Finally, CSMatch does the matching of fingerprints: One set of test features is compared to one or more enrolled and generalized fingerprints and a match level returned.

For an example of how to use these functions look at the demo program.

**File list:**

- biovfa41.h
- biovfa41.lib
- biovfa41.dll
- ms2mtc41.dll

The remainder of this document describes functions and data structures in detail.

## Data structures

The fingerprint recognition functions return and receive fingerprint data in an internal, encrypted and compressed format. The following data structures represent the explicit fingerprint data and can be obtained by calling the CSDecrypt function. These explicit data are for display purposes only. Extraction, generalization and matching, as well as database storage and retrieval can be done by using only the internal data format.

### FPDATA

#### Definition:

```
C++:
typedef struct _FPDATA
{
    int     features_number;
    int     *feature_coordinatex;
    int     *feature_coordinatey;
    int     *feature_direction;
    int     *feature_curvature;
    int     dirs_block_width
    int     dirs_block_length
    BYTE    Dirs    [MAX_DIRS_BLOCK_COUNT]
                    [MAX_DIRS_BLOCK_COUNT];
}
FPDATA;
```

#### Visual Basic:

```
Public Type FPDATA
    features_number As Long
    feature_coordinatex As Long
    feature_coordinatey As Long
    feature_direction As Long
    feature_curvature As Long
    dirs_block_width As Integer
    dirs_block_height As Integer
    dirs _
        (1 To MAX_DIRS_BLOCK_COUNT, _
        1 To MAX_DIRS_BLOCK_COUNT) _
        As Byte
End Type
```

**Function:**

Describes the features of a fingerprint.

**Members:**

features\_number  
Number of extracted fingerprint features

feature\_coordinatex  
Pointer to an array containing the x-coordinates of the features

feature\_coordinatey  
Pointer to an array containing the y-coordinates of the features

feature\_direction  
Pointer to an array containing the direction of the features

feature\_curvature  
Pointer to an array containing the curvature of the features

dirs\_block\_count  
Size of the blocked directional image

Dirs  
Blocked directional image

**Remarks:**

Memory for the arrays containing fingerprint features must be allocated and freed outside of the functions. The constant MAX\_FEATURES defines the size that should be allocated.

## FPCLASSDATA

### Definition:

```
C++:
typedef struct _FPCLASSDATA
{
    unsigned long      G;
    SINGULARPOINTS    SP;
}
FPCLASSDATA;
```

### Visual Basic:

```
Public Type FPCLASSDATA
    g As Long
    sp As SINGULARPOINTS
End Type
```

### Function:

Describes the general class of a fingerprint.

### Members:

- G      Fingerprint class. This value can be used to speed the matching process when working with a large database by comparing fingerprints with similar classes first.
- SP     Describes the location, orientation and type of singular points.

## SINGULARPOINTS

### Definition:

```
C++:
typedef struct _SINGULARPOINTS
{
    int          Count;
    int          X[MAX_SINGULARPOINT_COUNT];
    int          Y[MAX_SINGULARPOINT_COUNT];
    int          D[MAX_SINGULARPOINT_COUNT];
    int          T[MAX_SINGULARPOINT_COUNT];
}
SINGULARPOINTS;
```

### Visual Basic:

```
Public Type SINGULARPOINTS
    Count As Long
    x(1 To MAX_SINGULARPOINT_COUNT) As Long
    y(1 To MAX_SINGULARPOINT_COUNT) As Long
    d(1 To MAX_SINGULARPOINT_COUNT) As Long
    t(1 To MAX_SINGULARPOINT_COUNT) As Long
End Type
```

### Function:

Describes the singular points of the finger print.

### Members:

Count	Number of singular points.
X, Y	Location
D	Direction
T	Type: 1 = delta, 2 = core, 3 = double core

## VF\_PARAMETERS

### Definition:

```
C++:
typedef struct _VF_PARAMETERS
{
    double FAR_MATCHING;
    double FAR_GENERALIZATION;
    double FAR_CHECK_DUPLICATE;
    int MATCHING_THRESHOLD;
    int GENERALIZATION_THRESHOLD;
    int CHECK_DUPLICATE_THRESHOLD;
    int MATCHING_SPEED;
}
VF_PARAMETERS;
```

### Visual Basic:

### Function:

Parameter settings that influence the performance of the extraction and matching functions.

### Members:

FAR_MATCHING:	Desired false acceptance rate for matching (FAR, the probability, in percent, that a fingerprint will be erroneously accepted). Useful values range between 0.1 and 0.001.
FAR_GENERALIZATION:	The FAR for generalization. Same as above
FAR_CHECK_DUPLICATE:	Reserved for future use.
MATCHING_THRESHOLD:	Internal threshold value derived from the FAR_MATCHING setting. The function CSetParameter computes this value automatically.
GENERALIZATION_THRESHOLD:	Same as MATCHING_THRESHOLD.
CHECK_DUPLICATE_THRESHOLD:	Reserved for future use.
MATCHING_SPEED:	0 to 4. 0 is lowest, 4 is highest speed.

## Functions

### CSInitializeAPI

#### Definition:

```
C++:
int    __stdcall CSInitializeAPI
(
    char    *program_name,
    int     *version
);
```

#### Visual Basic:

```
Public Declare Function CSInitializeAPI Lib "biovfa41"
(
    ByVal program As String,
    version As Long
)
As Long
```

#### Example:

```
Dim program As String
Dim version As Long
Dim ret_code As Long

program = "BiocheckSDK" & Chr(0)
ret_code = CSInitializeAPI ( program, version )
```

#### Function:

API initialization.

**Parameters:**

program\_name (IN)  
The registered name of the SDK, usually "BiocheckSDK"  
version (OUT)  
41, the ms2mtc41.dll major version.

**Return Codes:**

TRUE: successful  
FALSE: ms2mtc41.dll could not be loaded or is not registered.

**Remarks:**

If you are not using contexts for thread safety (see CSCreateHandle and CSDestroyHandle below), you must call this function prior to using any other API functions. Before the end of execution you must then call CSUnloadAPI to free resources allocated by CSInitializeAPI. You may not call these functions when using contexts.

If the function returns FALSE, an error code can be obtained by calling the WIN32 API function **GetLastError**.

## Error codes:

CS_DLL_LOAD_ERROR	Cannot load ms2mtc41.dll
CS_NOT_REGISTERED	DLL could not be registered

## CSUnloadAPI

### Definition:

C++:  
void \_\_stdcall CSUnloadAPI ();

### Visual Basic:

Public Declare Sub CSUnloadAPI Lib "biovfa41" ()

Example:

CSUnloadAPI

### Function:

Cleanup.

### Remarks:

Call this function before program termination, if initialization was done with CSInitializeAPI

## CSCreateHandle

### Definition:

```
C++:  
FMHANDLE __stdcall CSCreateHandle ();
```

### Visual Basic:

```
Public Declare Function CSCreateHandle Lib "biovfa41" ()  
As Long
```

### Example:

```
Dim context As Long  
  
context = CSCreateHandle ()
```

### Function:

Obtain a handle to a matching context.

### Parameters:

none

### Return Value:

Handle to a matching context.

### Remarks:

This function is needed when matching is done concurrently in different threads. Each thread must create and destroy its own handle. The function is not needed for single threaded matching operations, in which case you should use CSInitializeAPI and CSUnloadAPI with a NULL context.

## CSDestroyHandle

### Definition:

```
C++:  
void __stdcall CSDestroyHandle  
(  
    FMHANDLE    fmh  
);
```

### Visual Basic:

```
Public Declare Sub CSDestroyHandle Lib "biovfa41"  
(  
    handle As Long  
)
```

### Example:

```
CSDestroyHandle context
```

### Function:

Destroy the handle obtained by CSGlobalHandle.

### Parameters:

fmh (IN)  
FMHANDLE obtained by CSGlobalHandle

### Return Codes:

none

### Remarks:

Must called before leaving each thread that has created a handle.

## CSGetParameters

### Definition:

```
C++:  
void __stdcall CSGetParameters  
(  
    VF_PARAMETERS *vfp,  
    FMHANDLE context  
);
```

### Visual Basic:

```
Public Declare Sub CSGetParameters Lib "biovfa41"  
(  
    vfp As VF_PARAMETERS,  
    ByVal context As Long  
)
```

### Example:

```
Dim vfp As VF_PARAMETERS  
  
CSGetParameters vfp, context
```

### Function:

Get current parameter settings.

### Parameters:

vfp (OUT)  
VF\_PARAMETERS structure, as described above.  
context (IN):  
context obtained by CSGlobalHandle or NULL

### Return Codes:

none

## CSSetParameters

### Definition:

```
C++:
void __stdcall CSSetParameters
(
    VF_PARAMETERS *vfp,
    FMHANDLE context
);
```

### Visual Basic:

```
Public Declare Sub CSSetParameters Lib "biovfa41"
(
    vfp As VF_PARAMETERS,
    ByVal context As Long
)
```

### Example:

```
Dim vfp As VF_PARAMETERS

CSSetParameters vfp, context
```

### Function:

Save parameter settings.

### Parameters:

vfp (IN)  
VF\_PARAMETERS structure, as described above.

context (IN):  
context obtained by CSGlobalHandle or NULL

### Return Codes:

none

## CSAverageGradient

### Definition:

```
C++:
void __stdcall CSAverageGradient
(
    BYTE *image,
    int width,
    int length,
    int top,
    int bottom,
    int left,
    int right,
    int max_empty,
    FMHANDLE context
);
```

### Visual Basic:

```
Public Declare Function CSAverageGradient Lib "biovfa41"
(
    ByVal p_image As Long,
    ByVal width As Long,
    ByVal length As Long,
    ByVal top As Long,
    ByVal bottom As Long,
    ByVal left As Long,
    ByVal right As Long,
    ByVal max_empty As Long,
    ByVal context As Long
)
As Long
```

### Example:

```
Dim img(1 To MAX_IMAGE_SIZE) As Byte
Dim p_img As Long
Dim width As Long, length As Long
Dim top As Long, bottom As Long
Dim left As Long, right As Long
Dim avg_gradient As Long

p_img = VarPtr(img(1))
avg_gradient = CSAverageGradient
    ( p_img, width, length, top, bottom, left, right, context )
```

### Function:

Returns the average gradient of the image.

**Parameters:**

image (IN)  
Buffer containing image from scanner

width (IN)  
Image width (columns)

length (IN)  
Image length (rows)

top (IN)  
Gradient window top offset.

bottom (IN)  
Gradient window bottom offset.

left (IN)  
Gradient window left offset.

right (IN)  
Gradient window right offset.

max\_empty (IN)  
Maximum gradient indicating an empty image.

context (IN):  
context obtained by CSGlobalHandle or NULL

**Return Codes:**

Average Gradient.

**Remarks:**

This function can be used to decide if a fingerprint image is present.  
Values depend on the scanner used, the ambient light and humidity, etc.  
Optimal threshold values must therefore be obtained experimentally.

## CSExtractFeatures

### Definition:

```
C++:  
void __stdcall CSExtractFeatures  
(  
    BYTE *image,  
    int width,  
    int length,  
    int resolution,  
    DWORD settings,  
    BYTE *features,  
    FMHANDLE context  
);
```

### Visual Basic:

```
Public Declare Function CSExtractFeatures Lib "biovfa41"  
(  
    ByVal p_image As Long,  
    ByVal width As Long,  
    ByVal length As Long,  
    ByVal resolution As Long,  
    ByVal settings As Long,  
    ByVal p_features As Long,  
    ByVal context As Long  
)  
As Long
```

### Example:

```
Dim img(1 To MAX_IMAGE_SIZE) As Byte  
Dim p_img As Long  
Dim width As Long, length As Long  
Dim resolution As Long, settings As Long  
Dim features(1 To MAX_FEATURES_SIZE) As Byte  
Dim p_features As Long  
Dim ret_code As Long  
  
p_img = VarPtr(img(1))  
p_features = VarPtr(features(1))  
rtc = CSExtractFeatures  
    (p_img, width, length, resolution, settings, p_features, context)
```

### Function:

Processes raw image and returns fingerprint feature data.

**Parameters:**

image (IN)  
Buffer containing image from scanner

width (IN)  
Image width (columns)

length (IN)  
Image length (rows)

resolution (IN)  
Image resolution in DPI.

settings (IN)  
Reserved. Must be zero.

features (OUT)  
Features array in internal format: Receives fingerprint feature data.  
Must be allocated before calling CSExtractFeatures.

context (IN):  
context obtained by CSCreateHandle or NULL

**Return Codes:**

None.

**Remarks:**

The array pointers feature\_coordinate\_x, feature\_coordinate\_y, feature\_direction and feature\_curvature must be initialized with addresses of integer arrays of length MAX\_FEATURES before calling the function. Storage allocation is the responsibility of the caller. The array content may be undefined.

## CSGeneralization

### Definition:

```
C++:  
int __stdcall CSGeneralization  
(  
    BYTE *feature_array[3],  
    BYTE *gen_features,  
    FMHANDLE context  
);
```

### Visual Basic:

```
Public Declare Function CSGeneralization Lib "biovfa41"  
(  
    p_feature_array As Long,  
    ByVal p_gfeatures As Long,  
    ByVal context As Long  
)  
As Long
```

### Example:

```
Dim features_array(1 To MAX_FEATURES_SIZE, 1 To 3) As Byte  
Dim p_features_array(1 To 3) As Long  
Dim gen_features(1 To MAX_FEATURES_SIZE) As Byte  
Dim p_gen_features As Long  
Dim ret_code As Long  
  
For i = 1 To 3  
    p_features_array(i) = VarPtr(features_array(1, i))  
Next  
p_gen_features = VarPtr(gen_features(1))  
ret_code = CSGeneralization  
    (p_features_array(1), p_gen_features, context)
```

### Function:

Combines three fingerprint feature datasets into one generalized, noise reduced dataset.

**Parameters:**

feature\_array (IN)  
Array of pointers to arrays of internal data obtained from three different images of the same fingerprint.  
gen\_features (OUT)  
Array that receives the generalized fingerprint features.  
context (IN):  
context obtained by CSGlobalCreateHandle or NULL

**Return Codes:**

-1: Insufficient quality for generalization  
>= 0: Quality of generalization (values see CSGlobalMatch below)

**Remarks:**

The array pointers feature\_coordinatex, feature\_coordinatey, feature\_direction and feature\_curvature must be initialized with addresses of integer arrays of length MAX\_FEATURES before calling the function. Storage allocation is the responsibility of the caller. The array content may be undefined.

## CSMatch

### Definition:

```
C++:  
int      __stdcall CSMatch  
(  
    BYTE          *features,  
    int           start_run_stop_flag,  
    FMHANDLE      context;  
);
```

### Visual Basic:

```
Public Declare Function CSMatch Lib "biovfa41"  
(  
    ByVal p_features As Long,  
    ByVal start_run_stop_flag As Long,  
    ByVal context As Long  
)  
As Long
```

### Example:

```
Dim test_features(1 To MAX_FEATURES_SIZE) As Byte  
Dim p_test_features As Long  
Dim db_features(1 To MAX_FEATURES_SIZE) As Byte  
Dim p_db_features As Long  
Dim rtc As Long  
  
p_test_features = VarPtr(test_features(1))  
p_db_features = VarPtr(db_features(1))  
  
'Capture and extract "test_features"  
rtc = CSMatch(p_features, 1, context)  
  
While ... 'not eof or similar  
    'read "db_features" from database  
    rtc = CSMatch(p_db_features, 0, context)  
    If rtc <> 0 Then  
        MsgBox "Found"  
        Exit While  
    End If  
Wend  
rtc = CSMatch(0, 2, context)
```

**Function:**

Match test fingerprint features with enrolled features.

**Parameters:**

features (IN)

Array of internal fingerprint features.

start\_run\_stop\_flag (IN)

1: Initialize a matching sequence. The array 'features' contains data of the finger to be searched for.

0: Match with sequence of templates. This step can be done multiple times to match the features from the initialization step with a sequence of different templates, for instance from a database. The 'features' array contains the templates.

2: Finalize matching. Must be called before starting another sequence of matches. The 'features' array can be NULL.

context (IN):

context obtained by CSGlobalCreateHandle or NULL

**Return Value:**

Quality of recognition (see Remarks).

**Remarks:**

A return value of 0 indicates the matching has failed. Otherwise the matching quality is returned.

## CSEncrypt

### Definition:

```
C++:  
void __stdcall CSEncrypt  
(  
    Fpdata *fpd,  
    FPCLASSDATA *fpc,  
    BYTE *features  
    FMHANDLE context;  
);
```

### Visual Basic:

```
Public Declare Function CSEncrypt Lib "biovfa41"  
(  
    fpd As Fpdata,  
    fpc As FPCLASSDATA,  
    ByVal p_features As Long  
    ByVal context As Long  
)  
As Long
```

### Example:

```
Dim fpd As Fpdata, fpc As FPCLASSDATA  
Dim vx(1 To MAX_FEATURES) As Long  
Dim vy(1 To MAX_FEATURES) As Long  
Dim vd(1 To MAX_FEATURES) As Long  
Dim vc(1 To MAX_FEATURES) As Long  
Dim features(1 To MAX_FEATURES_SIZE) As Byte  
Dim p_features As Long  
  
fpd.feature_coordinate_x = VarPtr(vx(1))  
fpd.feature_coordinate_y = VarPtr(vy(1))  
fpd.feature_direction = VarPtr(vd(1))  
fpd.feature_curvature = VarPtr(vc(1))  
p_features = VarPtr(features(1))  
CSEncrypt fpd, fpc, p_features, context
```

### Function:

Encryption and compression of fingerprint template data into a byte array.

**Parameters:**

- fpdata (IN)  
Pointer to FPDATA structure containing fingerprint feature data.
- fpclassdata (IN)  
Pointer to FPCLASSDATA structure containing fingerprint class data.
- features (OUT)  
Pointer to receive the encrypted string. Must be allocated by user.
- context (IN):  
context obtained by CSGlobalHandle or NULL

**Remarks:**

This function can be used to obtain a condensed, encrypted representation of the fingerprint data, e.g. for storage in a database.

The returned string must be freed with the VirtualFree function.

The key parameter can be any null-terminated string and is used for encryption.

## CSDecrypt

### Definition:

```
C++:  
void __stdcall CSDecrypt  
(  
    FPDATA *fpc,  
    FPCLASSDATA *fpc,  
    BYTE *features  
    FMHANDLE context;  
);
```

### Visual Basic:

```
Public Declare Function CSDecrypt Lib "biovfa41"  
(  
    fpc As FPDATA,  
    fpc As FPCLASSDATA,  
    ByVal p_features As Long  
    ByVal context As Long  
)  
As Long
```

### Example:

```
Dim fpc As FPDATA, fpc As FPCLASSDATA  
Dim vx(1 To MAX_FEATURES) As Long  
Dim vy(1 To MAX_FEATURES) As Long  
Dim vd(1 To MAX_FEATURES) As Long  
Dim vc(1 To MAX_FEATURES) As Long  
Dim features(1 To MAX_FEATURES_SIZE) As Byte  
Dim p_features As Long  
  
fpc.feature_coordinate_x = VarPtr(vx(1))  
fpc.feature_coordinate_y = VarPtr(vy(1))  
fpc.feature_direction = VarPtr(vd(1))  
fpc.feature_curvature = VarPtr(vc(1))  
p_features = VarPtr(features(1))  
CSDecrypt fpc, fpc, p_features, context
```

### Function:

Decryption of internal data into explicit features and class data.

**Parameters:**

- fpdata (OUT)
  - Pointer to FPDATA structure receiving the fingerprint feature data.
  - The structure must be properly initialized (see CSExtractFeatures).
- fpclassdata (OUT)
  - Pointer to FPCLASSDATA structure receiving the fingerprint class data.
- features (IN)
  - Array with encrypted and compressed fingerprint feature data.
- context (IN):
  - context obtained by CSCreateHandle or NULL

**Remarks:**

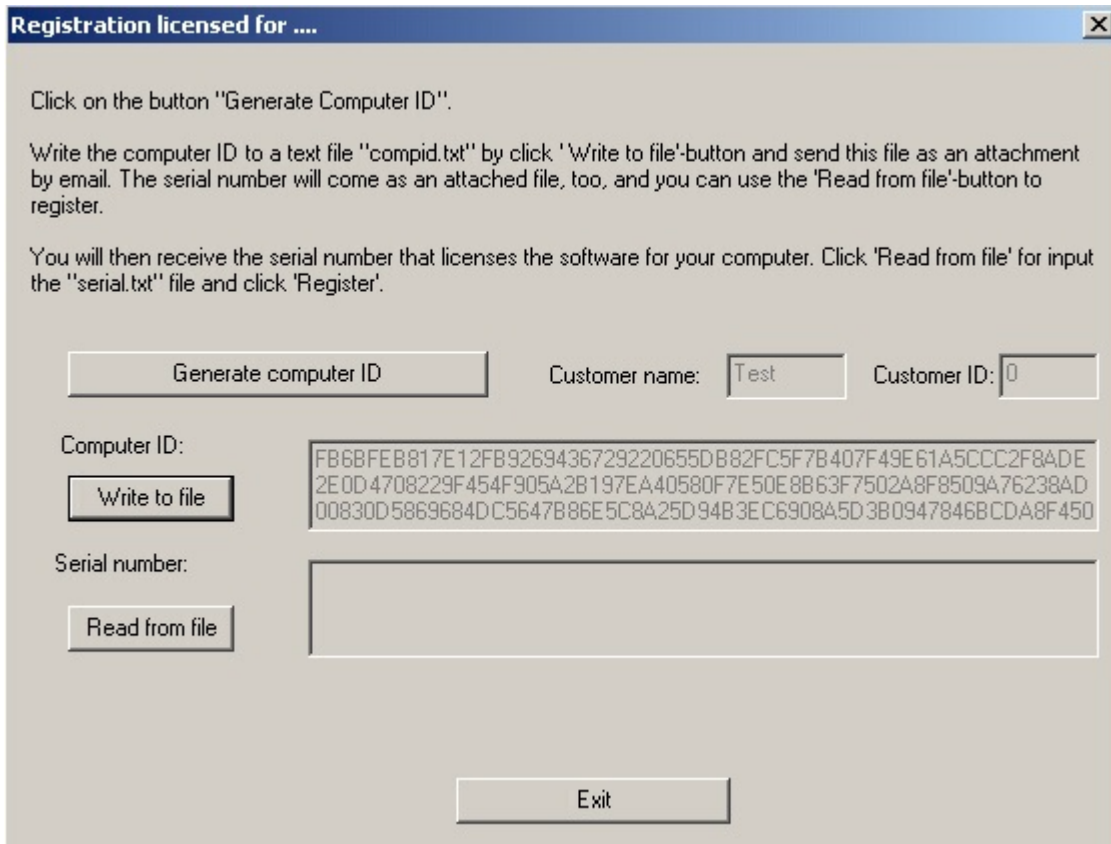
- This function decrypts a string encrypted by CSEncrypt and retrieves the fingerprint data.
- The key parameter must be the same as used for encryption.

# Part 3

## Registration

**Description:**

The Biocheck SDK needs to be registered before it can be used. The easiest way to do this is starting the C++ or VB demo program, which will display the registration dialog:



## CSSerialDlg

### Definition:

```
C++:  
int __stdcall CSSerialDlg  
(  
    char * program  
);
```

### Visual Basic:

```
Public Declare Function CSSerialDlg Lib "serial_dlg" _  
    (ByVal program As String) _  
    As Long
```

### Example:

```
Dim ret_code As Long  
  
program$ = "BiocheckSDK" & chr(0)  
ret_code = CSInitReader
```

### Function:

Display the registration dialog

### Parameters:

program (IN) "BiocheckSDK", if registration is for the SDK. Otherwise the name of the program product that needs to use biofva41.dll

### Return Codes:

any value Disregard. The return code is not related to the success or failure of the registration process.

### Remarks:

This function must be called when CSInitializeAPI returns FALSE and GetLastError returns CS\_NOT\_REGISTERED. The registration procedure is described below in detail.