

# API Specification

## ILO SID Biometric Interoperability Test

October 2007

Participants in ISBIT shall be required to provide Bion Biometrics, the ILO designated test lab, with an Application Programming Interface (API) that complies with the specifications detailed in this document.

**Note: All BIRs and raw BMP images produced by products for ISBIT will become the property of Bion Biometrics, which will safeguard them in accordance with all relevant privacy legislation under the terms of the personal information release forms signed by test subjects. In order to resolve interoperability issues or to support future offline tests, raw BMP images and/or corresponding SID-0002 BIRs produced by each vendor's product may be anonymously shared.**

## API and Platform Requirements

The API shall be submitted in the form of a compiled Win32 dynamic link library (DLL) which runs on Microsoft Windows XP SP2. The test software that imports the API functions is written in C# on the Microsoft .NET Framework 2.0 using the `DllImport` attribute.

The API specified by this document shall be implemented in a single base DLL file with the filename '`isbitapi.dll`'. Additional dynamic/shared library files may be submitted that support this base library file (i.e. the base DLL may have dependencies implemented in other libraries).

API functions specified to be used during both online and offline testing (**Enrol**, **VerifyProcess**, and **VerifyMatch**) shall not use any interactive mechanisms such as graphical user interface (GUI) calls, or anything requiring terminal interaction including calls to "standard input" or "standard output." These functions shall also run without the presence of the participant's biometric sensor and device drivers.

The API provided must not include multiple "modes" of operation, or algorithm variations.

The API shall access only that system memory that it allocates or that corresponds to the provided inputs and outputs. Furthermore, the API shall not communicate with any external processes, devices, or computers except those required for biometric capture. Modern desktop PCs with USB 2 and Firewire ports will be used for biometric capture in the lab.

## Installation

The API should install easily, and shall be executable on any number of machines without requiring additional hardware-based license control procedures. It is recommended that the API be installable using simple file copy methods, and not require the use of a separate installation program.

## Documentation

Complete documentation of any functionality or behaviour beyond what is specified in this document should be provided.

## Speed

On average, an **Enrol** operation should take no more than 7 seconds, a **VerifyProcess** operation should take no more than 1 second, and a **VerifyMatch** operation should take no more than 10 milliseconds to complete (using a 3 GHz Pentium IV).

# API Function Calls

## ImageSize

### C Prototype

```
int _stdcall ImageSize();
```

### Description

Returns the byte size of each uncompressed BMP image captured. The return value will be used by the calling application (test harness) to allocate an appropriately sized image buffer for the **Capture** function of the same product.

### Parameters

None.

### Return Values

The byte size of each raw fingerprint image in uncompressed BMP format.

## ITemplateMaxSize

### C Prototype

```
int _stdcall ITemplateMaxSize();
```

### Description

Returns the maximum byte size of an intermediate template that could be returned by the **VerifyProcess** function of the same product. The return value will be used by the calling application (test harness) to allocate an appropriately sized buffer for the **VerifyProcess** function of the same product.

### Parameters

None.

### Return Values

Buffer size for the **iTemplate** parameter of **VerifyProcess**.

## CaptureInit

### C Prototype

```
int _stdcall CaptureInit(const int showGUI);
```

### Description

Initializes biometric device before subsequent calls to **Capture**. Some devices require a perceptible duration for the automatic initialization and/or calibration of the sensor before running online capture transactions. The test harness will call this function once before each transaction consisting of multiple **Capture** attempts for enrolment or verification. This function will not be used during offline testing.

### Parameters

showGui (input): If the API provides an on-screen indicator via a window or GUI during the initialization period, a value of zero (0) will suppress the indication.

### Return Values

0	Success
-1	Failed to Initialize

If initialization for capture is successful, or if no initialization is required by the product, the API should return zero (success) when this function is called.

## CaptureEnd

### C Prototype

```
int _stdcall CaptureEnd();
```

### Description

Provides the opportunity for the API to perform any 'housekeeping chores' or resource de-allocation that may be required at the conclusion of a capture transaction. The test harness will call this function once after each transaction consisting of multiple **Capture** attempts for enrolment or verification. This function will not be used during offline testing.

### Parameters

None.

### Return Values

0	Success
-1	Failed to End Capture

If the capture session is shutdown successfully, or if no shutdown is required, the API should return zero (success) when this function is called.

# Capture

## C Prototype

```
int _stdcall Capture(  
    const unsigned char    finger,  
    const unsigned char    purpose,  
    unsigned char          *image);
```

## Description

Displays a window or GUI to prompt for placement of the finger corresponding to the **finger** parameter, and capture a single raw fingerprint image from the biometric device for either enrolment or verification as specified by the **purpose** parameter. This function will not be used during offline testing.

This function will be called once for each finger placement. Multiple placements will not be permitted during a single capture call, and a BMP image of the same size (as specified by the return value of the **ImageSize** function) must always be output, even if it is blank. If finger placement is automatically detected by the API, it must exit once the finger is removed from the sensor or the image has been acquired. If the API deems the image as unsuitable for the **purpose** indicated, it shall output the image and a return value of -1 (Failed to Acquire).

If, after 12 seconds, the administrator determines that the API fails to detect a legitimate finger placement, a button shall be provided in the GUI to allow the administrator to cancel the current capture operation, outputting an image and a return value of -2 (Cancelled by User). If the capture operation is cancelled, the presentation will not count as a failure-to-acquire by the test control software, and the image will be processed for the **purpose** indicated.

## Parameters

*finger* (input): A value from 1 to 10 corresponding to a valid finger position from SID-0002 or ANSI/NIST-ITL 1-2000, table 5.

*purpose* (input): A value of zero (0) will indicate a capture for the purpose of enrolment, while a non-zero value will indicate a capture for the purpose of verification.

*image* (output): The raw fingerprint image in an uncompressed Microsoft BMP format. A buffer will be allocated by the calling application to the size returned by the **ImageSize** function of the same product.

## Return Values

0	Success
-1	Failed to Acquire
-2	Cancelled by User

## Enrol

### C Prototype

```
int _stdcall Enrol(
    const unsigned char    fingerPrimary,
    const unsigned char    fingerSecondary,
    const unsigned char    *imagePrimary1,
    const unsigned char    *imagePrimary2,
    const unsigned char    *imagePrimary3,
    const unsigned char    *imageSecondary1,
    const unsigned char    *imageSecondary2,
    const unsigned char    *imageSecondary3,
    unsigned short         *birSize,
    unsigned char          *bir);
```

### Description

This function shall attempt to enrol both primary and secondary fingers as an SID-0002 BIR using up to three uncompressed BMP images for both the primary and secondary fingers captured on the same biometric device. This function will be used for both online and offline testing.

An SID-0002 conformant BIR should always be output. Therefore, if either the primary or the secondary finger could not be enrolled from the input images, the enrolled finger shall be designated as the primary fingerprint template and the secondary fingerprint template shall be 'unenrolled'. (see SID-0002 section 5.1.1 and Annex B) If neither the primary nor the secondary set of images could be enrolled, both the primary and secondary fingerprint templates of the BIR shall be 'unenrolled'. Return values -1, -2, and -3 will indicate that the enrolment of a different finger is required by the test harness.

### Parameters

*fingerPrimary* (input): A value from 1 to 10 corresponding to a valid finger position from SID-0002 or ANSI/NIST-ITL 1-2000, table 5.

*fingerSecondary* (input): A value from 1 to 10 corresponding to a valid finger position from SID-0002 or ANSI/NIST-ITL 1-2000, table 5.

*imagePrimary1*, *imagePrimary2*, *imagePrimary3* (input): Raw uncompressed BMP images from the same product corresponding to the finger identified by ***fingerPrimary***. May be set to null by the calling application.

*imageSecondary1*, *imageSecondary2*, *imageSecondary3* (input): Raw uncompressed BMP images from the same product corresponding to the finger identified by ***fingerSecondary***. May be set to null by the calling application.

*birSize* (output): The size of the BIR in bytes.

*bir* (output): SID-0002 BIR containing two fingerprint minutiae templates. A 566-byte buffer will be allocated by the calling application. The ***birSize*** parameter will specify the actual size.

### Return Values

0	Success
-1	Failed to Enrol Primary
-2	Failed to Enrol Secondary
-3	Failed to Enrol Primary and Secondary
-4	Unknown Image Format

# VerifyProcess

## C Prototype

```
int _stdcall VerifyProcess(  
    const unsigned char *image,  
    unsigned int        *iTemplateSize,  
    unsigned char       *iTemplate);
```

## Description

This function will process an input image (captured from the same product) into an intermediate (or proprietary) template to be used as an input to the same product's **VerifyMatch** function. This function will be used for both online and offline testing. This function is provided to enhance matching speed in the offline tests when many matches will be performed. It is assumed that in an operational verification, the system performing the verification would receive a live sample of the seafarer's fingerprint to compare with the BIR read from the SID. Since the live sample would not have to be converted to an SID-0002 conformant format, this function allows vendors to use a proprietary format for those verification images if they so choose.

## Parameters

*image* (input): Raw uncompressed BMP image from the same product.

*iTemplateSize* (output): The size of the intermediate template in bytes.

*iTemplate* (output): Intermediate template to be used as input to **VerifyMatch** function. A buffer will be allocated by the calling application to the size returned by the **ITemplateMaxSize** function of the same product.

## Return Values

0	Success
-1	Failed to Process Image
-2	Unknown Image Format

## VerifyMatch

### C Prototype

```
int _stdcall VerifyMatch(
    const unsigned int      iTemplateSize,
    const unsigned char    *iTemplate,
    const unsigned short   birSize,
    const unsigned char    *bir,
    const int              useSecondary,
    unsigned short         *score,
    int                    *match);
```

### Description

This will attempt to compare an intermediate template from the same product with either the primary or the secondary template within the input SID-0002 BIR. If the return value is non-zero, then the match and score parameters will be ignored. This function will be used for both online and offline testing.

### Parameters

*iTemplateSize* (input): The size of the intermediate template in bytes.

*iTemplate* (input): Intermediate template from the same product.

*birSize* (input): The size of the BIR in bytes.

*bir* (input): SID-0002 BIR containing two fingerprint minutiae templates.

*useSecondary* (input): A non-zero value shall indicate that the intermediate template should be matched with the secondary template of the SID-0002 BIR. A value of zero indicates that the intermediate template should be matched with the primary template of the SID-0002 BIR.

*score* (output): A similarity score resulting from the comparison of the intermediate template with the primary or secondary template of the SID-0002 BIR. The scores should range from a perfect non-match value of 0 (zero) to a perfect match value of 65,535.

*match* (output): A successful match (as determined by the internal threshold of the product) shall be indicated by a non-zero value, while an unsuccessful match will result in a value of zero (0).

### Return Values

0	Success
-1	Failed to Process Intermediate Template
-2	Failed to Process BIR
-3	Failed to Process Intermediate Template and BIR

## Release

### C Prototype

```
void _stdcall Release();
```

### Description

Frees all resources allocated by the API through prior function calls.

### Parameters

None.

### Return Values

None.