IREX

An Evaluation-based Program for the Development of Exchangeable Iris Imagery and Support for Compact Interoperable ISO/IEC 19794-6 Records

Iris Exchange (IREX) Evaluation 2008

Concept, Evaluation Plan and API

Patrick Grother

NIST

February 20, 2008

4.	Capture Device ID		2 B	0	Value will be provided to SDK. Value will be one of those in Table 11.
5.	Number of eyes imaged		1 B	1	
6.	Record header length		2 B	45	
7.	Iris image properties	Horz. orientation	2 b	ORIENTATION_BASE	
8.		Vert. orientation	2 b	ORIENTATION_BASE	
9.		Scan type	2 b	SCAN_TYPE_PROGRESSIVE, SCAN_TYPE_INTERLACE_FRAME	Rectilinear format only
10.		Occlusions	1 b	0	Polar format only
11.		Occlusion filling	1 b	0	Polar format only
12.		Boundary extraction	1 b	0	Polar format only
13.	Iris diameter (rect)		2 B	> 0	Rectilinear format only
14.	Image format		2 B	IMAGEFORMAT_MONO_RAW	IREX SDKs are neither permitted nor required to compress or decompress image data. Thus this field shall not be IMAGEFORMAT_MONO_JPEG, or IMAGEFORMAT_MONO_JPEG2000.
15.	Raw image width		2 B	> 0	
16.	Raw image height		2 B	> 0	
17.	Intensity depth		1 B	8	
18.	Image transformation (polar only)		1 B	1	TRANS_STD = 1 i.e. linear radial interpolation per [STD05, 6.5.4].
19.	Device unique identifier		16 B	'0000000000000000'	16 character zeroes '0' = 0x30, not '\0' = 0x00..
	Iris Biometric Subtype Header (Table 3 of ISO/IEC 19794-6:2005)				
20.	Eye		1 B	EYE_RIGHT, EYE_LEFT	
21.	Number of iris images of this eye		2 B	1	
	Iris Image Header (Table 4 of ISO/IEC 19794-6:2005)				
22.	Image number		2 B	1	This field is an index starting at 1
23.	Quality		1 B	0 ≤ Q ≤ 100	This field to be populated with value.
24.	Rotation angle of eye		2 B	ROT_ANGLE_UNDEF > 0	For rectilinear For polar unless next line is set non-zero
25.	Rotation uncertainty		2 B	ROT_UNCERTAIN_UNDEF > 0	For rectilinear and polar when rotation is not estimated. For polar when rotation is estimated.
26.	Mask value for upper eyelid		1B		If depth was > 8 bits these should be >8 bits too. For IREX 08, everything is 8 bits.
27.	Mask value for lower eyelid		1B		
28.	Mask value for sclera		1B		
29.	Auxiliary information length		2 B	LEN	See section 7.6.
30.	Auxiliary information		LEN		
31.	Image length		4 B	LEN	
32.	Image data		LEN		

Q9	Are three grey-level values needed (i.e. lines 26, 27 and 28). Or two?

This is identically the structure of the ISO/IEC 19794-6:2005 standard. Note that the standard treats rotation estimates and uncertainties differently for rectilinear and polar instances, and thus appropriate values shall be assigned.

7.5. Unsegmented Polar image record structure

The following structure is advanced as an implementation of the proposed unsegmented polar addition to the ISO standard [IRI07]. Yellow shading indicates new fields. **If adopted here, and used successfully, the records defined herein would form the basis of NIST contributions toward revision of ISO/IEC 19794-6.**

Table 9 – Structure for the proposed unsegmented polar format

	Section title and/or field name	L	IREX actual or required values	Remarks

	Iris Record Header (Table 2 of ISO/IEC 19794-6:2005)				
1.	Format Identifier		4 B	0x49495200	i.e. ASCII "IIR\0"
2.	Version Number		4 B	0x30313200	i.e. ASCII "013\0". This value is non-standard, and instituted by NIST to differentiate it from the properly standardized record in Table 7.
3.	Record Length		4 B	$57 \leq L \leq 2^{32} - 1$	Total length
4.	Capture Device ID		2 B	0	Value will be provided to SDK. Value will be one of those in Table 11.
5.	Number of eyes imaged		1 B	1	
6.	Record header length		2 B	45	
7.	Iris image properties	Horz. Orientation	2 b	ORIENTATION_BASE	
8.		Vert. orientation	2 b	ORIENTATION_BASE	
9.		Scan type	2 b	SCAN_TYPE_PROGRESSIVE, SCAN_TYPE_INTERLACE_FRAME	Rectilinear format only
10.		Occlusions	1 b		Polar format only
11.		Occlusion filling	1 b		Polar format only
12.		Boundary extraction	1 b		Polar format only
13.	Iris diameter (rect)		2 B		Rectilinear format only
14.	Image format		2 B	IMAGEFORMAT_MONO_RAW	IREX SDKs are neither permitted nor required to compress or decompress image data. Thus this field shall not be IMAGEFORMAT_MONO_JPEG, or IMAGEFORMAT_MONO_JPEG2000.
15.	Raw image width		2 B	> 0	
16.	Raw image height		2 B	> 0	
17.	Intensity depth		1 B	8	
18.	Image transformation (polar only)		1 B	1	TRANS_STD = 1 i.e. linear radial interpolation per [STD05, 6.5.4].
19.	Device unique identifier		16 B	'0000000000000000'	16 character zeroes '0' = 0x30, not '\0' = 0x00
	Iris Biometric Subtype Header (Table 3 of ISO/IEC 19794-6:2005)				
20.	Eye		1 B	EYE_RIGHT, EYE_LEFT	
21.	Number of iris images of this eye		2 B	1	
	Iris Image Header (Table 4 of ISO/IEC 19794-6:2005)				
22.	Image number		2 B	1	This field is an index starting at 1
23.	Quality		1 B	$0 \leq Q \leq 100$	This field to be populated with value.
24.	Rotation angle of eye		2 B	ROT_ANGLE_UNDEF	Values will not be provided to SDK
25.	Rotation uncertainty		2 B	ROT_UNCERTAIN_UNDEF	
26.	X coordinate of inner + outer circle centers		2 B	$0 \leq x < W$	Coordinate system is zero oriented with (0,0) at the top left corner. The inner and outer circle centers are concentric
27.	Y coordinate of inner + outer circle centers		2 B	$0 \leq y < H$	
28.	Inner circle radius		2 B	$0 \leq r$	
29.	Outer circle radius		2 B	$0 \leq r$	
30.	Auxiliary information length		2 B	LEN	See section 7.6.
31.	Auxiliary information		LEN		
32.	X coordinate of the center of the ellipse approximating the pupil boundary		2 B		
33.	Y coordinate of the center of the ellipse approximating the pupil boundary		2 B		
34.	X coordinate of the intersection point of the semi-major axis with the ellipse approximating the pupil		2 B		
35.	Y coordinate of the intersection point of the semi-major axis with the ellipse approximating the pupil		2 B		
36.	X coordinate of the intersection point of the semi-minor axis with the ellipse approximating the pupil		2 B		

37.	Y coordinate of the intersection point of the semi-minor axis with the ellipse approximating the pupil	2 B		
38.	X coordinate of the center of the ellipse approximating the iris boundary	2 B		
39.	Y coordinate of the center of the ellipse approximating the iris boundary	2 B		
40.	X coordinate of the intersection point of the semi-major axis with the ellipse approximating the iris	2 B		
41.	Y coordinate of the intersection point of the semi-major axis with the ellipse approximating the iris	2 B		
42.	X coordinate of the intersection point of the semi-minor axis with the ellipse approximating the iris	2 B		
43.	Y coordinate of the intersection point of the semi-minor axis with the ellipse approximating the iris	2 B		
44.	Image length	4 B	LEN	
45.	Unsegmented polar-transformed image data	LEN		

7.6. Auxiliary Data

Regarding the Auxiliary Data block on Lines 30-31 of Table 9, and Lines 29-30 of Table 8, NIST does not intend to allow purely vendor-defined data (e.g. a proprietary template) to be placed in such a block because it is non-interoperable[8]. NIST instead proposes to standardize the contents of this block in two parts, as follows.

- Eight-connected Freeman chain code (FCC) of the closed-path inner boundary.
- Eight-connected Freeman chain code (FCC) of the closed-path outer boundary.

Eight-connected FCCs allow encoding of an arbitrary path in 3 bits per pixel. For an iris of radius 100 pixels, and a pupil of radius 40 pixels, such encoding would require, with a small header, around 220 and 90 bytes respectively.

Table 10 - Format for Freeman chain code

#	Field	Length
1	X-coordinate of first pixel in closed path	2B
2	Y-coordinate of first pixel in closed path	2B
3	Number of elements in chain code	N
4	Bit-packed Freeman chain code, zero padded to nearest octet if (3N % 8 != 0)	≤ 3N/8 + 1

Q12	Do you have alternatives or additions to FCC? These should be precisely documented, non-proprietary, and available to be specified in this document.
Q15	Do you want Freeman chain codes? (If yes, ellipse info lines 32-43 would be deleted). If not, why not?
Q18	Do you want a vendor-defined auxiliary data block?

[8] The issue of allowing Extended Data has been discussed with respect to other biometric standards and has been deprecated because it is only valuable when the producer and consumer of the data are manufactured by the same company.

8. PC-based API specification

8.1. Overview

This section describes the IREX API. All SDK's submitted to IREX 08 shall implement the functions below here as required by the classes of participation listed in Table 4.

8.2. Testing interface

8.2.1. Requirement

IREX participants shall submit an SDK which presents the "C" prototyped interface given in the following subsections.

8.2.2. Sensor identifiers

The following sensors will be identified to the SDK using the two byte unsigned integer values in Table 11.

Table 11 – Sensor identifiers

#	Sensor Manufacturer and Model	Identifier
1	LG 2200	0x2A16
2	LG 3000	0x2A1E
3	Unknown	0x0000

Presence on this table indicates NIST's intention to use images captured by these devices. NIST may add to this table in due course. Also, please see NIST's call for images on Page 2.

8.2.3. Geometric or other alterations to images

In some cases poor images will need to be corrected. For example, the following action is described in Section 2 of [CAM07].

> For those images in which the iris was partly outside of the original image frame, the missing pixels were replaced with black ones. For those in which the algorithms detected that the gaze was directed away from the camera, as gauged by projective deformation of the eye shape, a corrective affine transformation was automatically applied which effectively "rotated" the eye in its socket back into orthographic perspective on-axis with the camera.

Such steps are allowed and are likely to allow downstream feature extractors and matchers to give better performance. NIST takes no position on whether these or other operations (e.g histogram equalization) should be applied.

Any such processing shall be conducted during the preparation of the ROI-masked rectilinear instance of section 8.2.5, or the unsegmented polar instance of section 8.2.6. It shall not be performed in the preparation of the rectilinear instance of section 8.2.4.

8.2.4. Conversion of raw imagery to standard rectilinear image

To support the standard, all submissions to IREX 08 shall implement a function to execute the packaging operation of Table 12. While this action is merely syntactic, prior interoperability tests conducted by NIST have revealed a not uncommon inability to reliably instantiate conformant records.

The output instances are used as inputs to subsequent functions (polar processing, or feature extraction). NIST will apply compression-decompression to the image data contained in these records.

Table 12 – Preparation of standard rectilinear records

Input	Action	Output
Raw raster, dimensions, and parameters needed to instantiate	Package the input into a conformant record.	Conformant ISO/IEC 19794-6 rectilinear record. These records

the standardized record	Do not alter the image data.	will be checked for conformance.

The function shall be implemented with the API call specified in Table 13.

Table 13 – IREX API preparation of standard rectilinear record

Prototype	INT32 convert_raster_to_rectilinear(BYTE *uncompressed_raster_data, const UINT16 image_width, const UINT16 image_height, const BYTE horz_orientation, const BYTE vert_orientation, const BYTE scan_type, const BYTE image_format, const BYTE intensity_depth, const UINT16 nist_encoded_device_id, BYTE *quality BYTE * ISO_19794_6_rectlinear_image);	
Description	This function takes a raw input image and outputs the corresponding ISO/IEC 19794-6 rectilinear record. This function executes only a syntactic repackaging of the input data. It shall not alter the image data.	
Input Parameters	uncompressed_raster_data	The uncompressed raw image used for template creation.
	image_width	The number of pixels indicating the width of the image.
	image_height	The number of pixels indicating the height of the image.
	horz_orientation	NIST anticipates setting these values to ORIENTATION_BASE, per [STD05, 6.5.4].
	vert_orientation	
	scan_type	Progressive or interlaced. Values per the standard.
	image_format	NIST anticipates using only unprocessed uncompressed 8 bit grayscale data, so the image format will be 0x0002, and the intensity depth will be 8, both per [STD05].
	intensity_depth	
	nist_encoded_device_id	A two byte unsigned integer value from Table 11
Output Parameters	Quality	A [0,100] quality value representing
	ISO_19794_6_rectlinear_image	The output rectilinear image, per Table 7
Return Value	0	Success
	Other	Vendor-defined

8.2.5. Conversion of rectilinear to ROI-masked rectilinear

To assess viability of the standard's polar format, participating submissions to IREX 08 shall execute the conversion operation of Table 14. In addition to the ROI-masked output image, the function shall return center coordinates. This:

- supports measurements of the pixel-level displacement between segmentation algorithms, and
- allows NIST to run execute JPEG cropping as in section 2 of [CAM07]
- allows NIST to run JPEG 2000 ROI tests (as has been considered for ISO/IEC 19794-5 Token face images)

Table 14 – Preparation of ROI-masked records

Input	Action	Output
Conformant ISO/IEC 19794-6 rectilinear record.	Find iris-eyelid and iris-sclera boundaries and mask those regions with vendor fixed values. Perform corrective geometry actions (sec XX).	ROI-masked record, per Table 7. Radius and center coordinates of the outer circle.

The function shall be implemented with the API call specified in Table 15.

Table 15 – IREX API for creation of ROI-masked records

Prototype	INT32 convert_rectilinear_to_ROI_masked_rectilinear(const BYTE *ISO_19794_6_rectlinear_image, UINT16 *x_iris_center, UINT16 *y_iris_center, UINT16 *iris_radius BYTE * roi_masked_rectilinear_image);	
Description	This function takes a conformant ISO/IEC 19794-6 rectilinear image and outputs the corresponding ROI-masked image. The coordinates of the pupil and iris centers are returned also. The memory for the output structure is allocated before the call i.e. the implementation shall not allocate memory for the result. The function returns either success (0) or failure (non-zero). Failure indicates a failure to convert the image. The result will nevertheless be a conformant instance with zero irides and zero images.	
Input Parameters	ISO_19794_6_rectlinear_image	The uncompressed raw image used for template creation.
Output Parameters	x_iris_center	Horizontal and vertical locations of the iris center.
	y_iris_center	
	iris_radius	Radius of a circle containing the entire iris.
	roi_masked_rectilinear_image	The output polar image, per Table 8.
Return Value	0	Success
	1	Involuntary failure to produce an output record - e.g. could not find iris-sclera boundary.
	2	Elective refusal to not produce an output record - e.g. on quality grounds.
	3	Cannot parse input data (i.e. assertion that input record is non-conformant)
	Other	Vendor-defined failure

The number of times a non-zero error codes is returned will be counted, reported and appropriately factored into analyses.

8.2.6. Conversion of rectilinear to unsegmented polar

To examine the viability of the proposed unsegmented polar format, participating submissions to IREX 08 shall execute the conversion of operation of Table 16. The function shall be implemented with the API call specified in Table 17.

In addition to the raw input record, three additional parameters are passed in.

- To allow surveys over the radial and circumferential sampling rates the function takes as input number-of-samples arguments.
- To parameterize a loosening of the segmentation, the function takes an outer radius multiplier, specified as a percentage of the supplier's best computation of the circle enclosing all regions of the iris. This is introduced per recommendation to "over-segment the iris by 15% of the algorithm-determined iris boundary before converting it to Polar format" [LG]. NIST will set the default to be 115, and survey over this parameter.

Q19	Is a NIST survey over a "radius multiplier" worth the trouble? Alternatives could be for IREX to mandatorily fix: a factor (e.g. 115), or a margin (e.g. 8 pixels).

In addition to the required polar instance output, the function shall return center coordinates. This:

- supports measurements of the pixel-level displacement between segmentation algorithms, and
- allows NIST to run execute JPEG cropping as in section 2 of [CAM07]
- allows NIST to run JPEG 2000 ROI tests (as has been considered for ISO/IEC 19794-5 Token face images)

Table 16 – Preparation of unsegmented polar records

Input	Action	Output

Conformant ISO/IEC 19794-6 rectilinear record, desired circumferential and radial numbers of samples, and radius multiplier.	Find suitable concentric inner and outer circles. Execute forward polar transform.	Proposed unsegmented polar record, per Table 9.

Table 17 – IREX API for creation of unsegmented polar records

Prototype	INT32 convert_rectilinear_to_unsegmented_polar(const BYTE *ISO_19794_6_rectlinear_image, const UNIT16 num_samples_radially, const UINT16 num_samples_circumferentially, const UINT16 outer_radius_multiplier, BYTE * unseg_polar_image);	
Description	This function takes a conformant ISO/IEC 19794-6 rectilinear image and outputs the corresponding unsegmented polar image. The memory for the template is allocated before the call i.e. the implementation shall not allocate memory for the result. The function returns either success (0) or failure (non-zero). Failure indicates a failure to convert the image. The result will nevertheless be a conformant instance with zero irides and zero images.	
Input Parameters	ISO_19794_6_rectlinear_image	The uncompressed raw image as prepared by the function in Table 13.
	num_samples_radially	The number of sample along a spoke. The output polar data shall have this height.
	num_samples_circumferentially	The number of "spokes" around the iris. The output polar data shall have this width.
	outer_radius_multiplier	NIST proposes to set the default value to 115 indicating the SDK should add 15% to its computation of the minimal outer circle.
Output Parameters	unseg_polar_image	The output template, per Table 9.
Return Value	0	Success
	1	Involuntary failure to extract features (e.g. could not find iris in the input-image)
	2	Elective refusal to produce a template (e.g. insufficient iris area)
	3	Cannot parse input data (i.e. assertion that input record is non-conformant)
	Other	Vendor-defined failure.

The number of times a non-zero error codes is returned will be counted, reported and appropriately factored into analyses.

8.2.7. Template creation

This function converts any of the input data structures to an opaque proprietary template. The function will need to look at the header of the input record to determine the content. This routine will need to heed the hexadecimal "Version Number" values in the Table 7 and Table 9 records.

Table 18 – IREX API template creation

Prototype	INT32 convert_image _to_template(const BYTE *input_record, UINT16 *template_size, BYTE *proprietary_template);
Description	This function takes either — a conformant ISO/IEC 19794-6 rectilinear image, or — an ROI-masked image, or — an unsegmented polar image, and outputs a proprietary template. The implementation should inspect the input header to determine which kind of imagery is being provided, per the version number values given in section 7.2. The memory for the output template is allocated before the call i.e. the implementation shall not allocate memory for the result. In all cases, even when unable to extract features, the output shall be a template record that may be

	passed to the match_templates function without error. That is this routine must internally encode "template creation failed" and the matcher must transparently handle this.	
Input Parameters	input_record	An input image presented either as instance of Table 7, Table 8, or Table 9..
Output Parameters	Template_size	The size, in bytes, of the output template
	proprietary_template	The output template. The format is entirely unregulated.
Return Value	0	Success
	1	Involuntary failure to extract features (e.g. could not find iris in the input-image)
	2	Elective refusal to produce a template (e.g. insufficient iris area)
	3	Cannot parse input data (i.e. assertion that input record is non-conformant)
	Other	Vendor-defined failure

The number of times a non-zero error codes is returned will be counted, reported and appropriately factored into analyses.

8.2.8. Template comparison

This function compares two proprietary templates and returns a real-valued distance score.

Table 19 – IREX API template matching

Prototype	INT32 match_templates(const BYTE *verification_template, const UINT16 verification_template_size, const BYTE *enrollment_template, const UINT16 enrollment_template_size, double *dissimilarity);	
Description	This function compares two opaque proprietary templates and outputs a non-negative match score. The returned score is a distance measure. It need not satisfy the metric properties. NIST will allocate memory for this parameter before the call. the function shall assign the value -1 to the score.	
Input Parameters	verification_template	A template from create_template().
	verification_template_size	The size, in bytes, of the input verification template $0 \le N \le 2^{16} - 1$
	enrollment_template	A template from create_template().
	enrollment_template_size	The size, in bytes, of the input enrollment template $0 \le N \le 2^{16} - 1$
Output Parameters	dissimilarity	A dissimilarity score resulting from comparison of the templates, on the range [0,DBL_MAX]. See section 8.2.9.
Return Value	0	Success
	Other	Vendor-defined failure

8.2.9. Dissimilarity score

The template comparison function shall return a measure of the dissimilarity between the persons whose iris data is contained in the two templates. So, smaller values indicate more likelihood that the two samples are from the same person. There is no requirement for values to obey the metric property.

This deviates from many prior NIST tests which have used "larger-is-more-genuine" semantics.

8.2.10. Implementation identifiers

The implementation shall support the self-identification function of Table 20. This function is required to support internal NIST book-keeping. The version numbers should be distinct between any versions which offer different algorithmic functionality.

Table 20 – IREX API get_pids function

Prototype	INT32 get_pid(UINT32 *nist_assigned_identifier);

Description	This function retrieves an identifier that the provider must request from NIST, and compile into the source code. NIST will assign the identifier that will uniquely identify the supplier and the SDK version number.	
Output Parameters	nist_assigned_identifier	A PID which identifies the SDK under test. The memory for the identifier is allocated by NIST's calling application, and shall not be allocated by the SDK.
Return Value	0	Success
	Other	Vendor-defined failure

8.3. Software and Documentation

8.3.1. SDK Library and Platform Requirements

Participants shall provide NIST with binary code only (i.e. no source code) – supporting files such as header (".h") files notwithstanding. Such files shall not contain intellectual property of the company nor any material that is otherwise proprietary. It is preferred that the SDK be submitted in the form of a single static library file (ie. ".LIB" for Windows or ".a" for Linux). However, dynamic/shared library files are permitted.

If dynamic/shared library files are submitted, it is preferred that the API interface specified by this document be implemented in a single "core" library file with the base filename 'libIREX' (for example, 'libIREX.dll' for Windows or 'libIREX.so' for Linux). Additional dynamic/shared library files may be submitted that support this "core" library file (i.e. the "core" library file may have dependencies implemented in these other libraries).

8.3.2. Linking

NIST will link the provided library file(s) to a C language test driver application developed by NIST. The runtime environment shall be either

- The cygwin[9] layer running on a Windows Server 2003 OS.
- RedHat Linux Enterprise 4 or 5 platforms.

Both will use GNU's gcc compiler, version 3.3.3. These use libc. The link command might be:

- `gcc –o irextest irextest.c -L. –lIREX`

Participants are required to provide their library in a format that is linkable using GCC with the NIST test driver, which is compiled with GCC. All compilation and testing will be performed on x86 platforms. Thus, participants are strongly advised to verify library-level compatibility with GCC (on an equivalent platform) prior to submitting their software to NIST to avoid linkage problems later on (e.g. symbol name and calling convention mismatches, incorrect binary file formats, etc.).

Dependencies on external dynamic/shared libraries such as compiler-specific development environment libraries are discouraged. If absolutely necessary, external libraries must be provided to NIST upon prior approval by the Test Liaison.

8.3.3. Installation and Usage

The SDK must install easily (i.e. one installation step with no participant interaction required) to be tested, and shall be executable on any number of machines without requiring additional machine-specific license control procedures or activation.

The SDK's usage shall be unlimited. The SDK shall neither implement nor enforce any usage controls or limits based on licenses, execution date/time, number of executions, presence of temporary files, etc.

It is recommended that the SDK be installable using simple file copy methods, and not require the use of a separate installation program. Contact the Test Liaison for prior approval if an installation program is absolutely necessary.

[9] According to http://www.cygwin.com/ is a Linux-like environment for Windows. It consists of two parts: A DLL (cygwin1.dll) which acts as a Linux API emulation layer providing substantial Linux API functionality; a collection of tools which provide Linux look and feel.

8.3.4. Documentation

Participants shall provide complete documentation of the SDK and detail any additional functionality or behavior beyond that specified here. The documentation must define all (non-zero) vendor-defined error or warning return codes..

8.3.5. Modes of operation

Individual SDKs provided shall not include multiple “modes” of operation, or algorithm variations. No switches or options will be tolerated within one library. For example, the use of two different “coders” by an iris feature extractor must be split across two separate SDK libraries, and two separate submissions.

8.4. Runtime behavior

8.4.1. Speed

The following limits are instituted to constrain NIST's total IREX computational workload.

- The mean template match operation shall not exceed 10 milliseconds.
- The mean template creation operation shall not exceed 1.2 seconds (using a 2GHz Pentium IV).
- The mean iris segmentation operation (e.g. polar) shall not exceed 1.2 seconds (using a 2GHz Pentium IV).

Q21	Should these times be lowered (for operational relevance) or raised (for improved algorithmic function)?

8.4.2. Interactive behavior

The SDK will be tested in non-interactive “batch” mode (i.e. without terminal support). Thus, the submitted library shall not use any interactive functions such as graphical user interface (GUI) calls, or any other calls which require terminal interaction e.g. reads from “standard input”.

8.4.3. Error codes and status messages

The SDK will be tested in non-interactive “batch” mod, without terminal support. Thus, the submitted library shall run quietly, i.e. it should not write messages to "standard error" and shall not write to “standard output”.

8.4.4. Exception Handling

The application should include error/exception handling so that in the case of a fatal error, the return code is still provided to the calling application.

8.4.5. External communication

Processes running on NIST hosts shall not side-effect the runtime environment in any manner, except for memory allocation and release. Implementations shall not write any data to external resource (e.g. server, file, connection, or other process), nor read from such. If detected, NIST will take appropriate steps, including but not limited to, cessation of evaluation of all implementations from the supplier, notification to the provider, and documentation of the activity in published reports.

8.4.6. Stateful behavior

All components in this test shall be stateless. This applies to segmentation, feature extraction and matching. Thus, all functions should give identical output, for a given input, independent of the runtime history. NIST will institute appropriate tests to detect stateful behavior. If detected, NIST will take appropriate steps, including but not limited to, cessation of evaluation of all implementations from the supplier, notification to the provider, and documentation of the activity in published reports.

A.7 Reporting of results

A.7.1 Reports

The Government will combine appropriate results into one or more IREX reports. Together these will contain, at a minimum, descriptive information concerning IREX, descriptions of each experiment, and aggregate test results. NIST will include

- DET performance metrics as the primary indicators of one-to-one verification accuracy,
- ISO/IEC 19795-4 interoperability matrices as the primary measures of interoperability, and
- Image generation, template generation, and matching timing statistics.

NIST may compute and report other aggregate statistics.

NIST intends to release Phase 1 results to the participant only.

NIST intends to publish Phase 2 results in one or more NIST Interagency Reports. The Phase 2 reports will:

- contain the names of Phase 1 participants,
- not contain the results from Phase 1 participants' implementations,
- contain the names of Phase 2 participants, and
- contain the results of all Phase 2 participants' implementations which will associated with the participants names.

A.7.2 Pre-publication review

Participants will have an opportunity to review and comment on the reports. Participants' comments will be either incorporated into the main body of the report (if it is decided NIST reported in error) or published as an addendum. Comments will be attributed to the participant.

A.7.3 Citation of the report

Subsequent to publication of our reports Participants may decide to use the results for their own purposes. Such results shall be accompanied by the following phrase: "Results shown from the Iris Exchange Test (IREX) do not constitute endorsement of any particular system by the U. S. Government." Such results shall also be accompanied by the URL of the IREX Report on the IREX website, http://iris.nist.gov/irex.

A.7.4 Rights and ownership of the data

Any data generated, deduced, measured or otherwise obtained during IREX (excepting the submitted SDK itself), as well as any documentation required by the Government from the participants, becomes the property of the Government. Participants will not possess a proprietary interest in the data and/or submitted documentation.

A.8 Return of the supplied materials

A.8.1 Returning software to vendors

NIST will not return any supplied software, documentation, or other material to vendors.

A.8.2 Returning cards to vendors

NIST will not return cards to the provider. NIST will destroy the cards within ninety days of publication of the results for that card or notification to the vendor that the card is inoperable. This requirement is needed because template data on the card is protected and because NIST has no mechanism to assure deletion of templates from the card. However, NIST to support debugging NIST may, at its sole discretion, return cards during the initial acceptance testing phase.