Aratek IrisEngine SDK

Development Guide

Aratek Biometrics Co.,Ltd.

www.aratek.co

directory

1 INTRODUCTION

1.1 Overview

This document describes how applications running on Windows/Linux platforms can use the IrisEngine SDK

1.2 Revision history

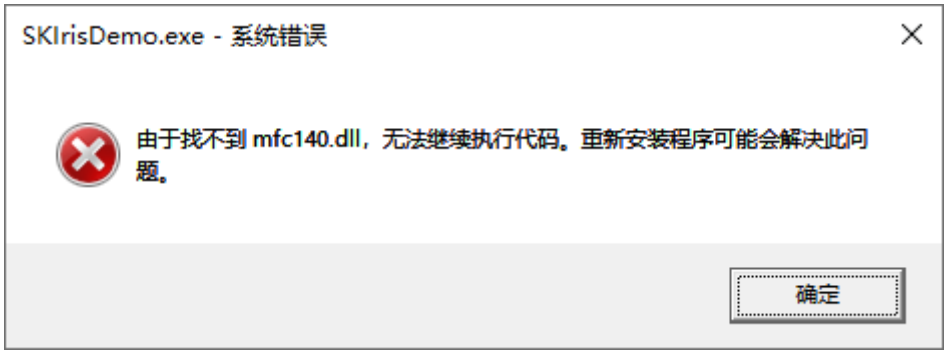
Version	Date	Content
1.0	2019-10-30	Initial version
1.2	2020-03-19	Adjust and improve interfaces
1.3	2020-04-20	Join [Prerequisite for Use]
1.4	2020-12-07	Additional Instructions for installing the VS2017 release package
1.5	2021-04-09	Add instructions for device specific parameters
1.6	2022-03-20	* Remove requirements and instructions for registering components * Add an interface to get device information in json format * Add an interface to get algorithm version and release date
1.7	2022-04-20	* Correct the note about DIR_DATA/DIR_EYEDATA * Add a description about load_user_dir * Add a description about the error code * Updated instructions about set preview
1.8	2022-06-10	*Add a description of the Linux platform
1.9	2022-08-24	*Add instructions for the following functions: IrisEng_load_dev_params IrisEng_fetch_enroll_data IrisEng_fetch_snap_data
1.11	2023-09-11	Instructions for joining Callback
1.12	2024-02-26	*Change IrisEng_set_preview to IrisEng_set_preview2 *Support ISO-IEC-19794-6-2011 by below: #on/off "generate_IIBDR":"on", #jpg/bmp/j2k/png "image_save_format":"png", #uncropped, vga, cropped "image_save_type":"cropped",

2 PREREQUISITES

2.1 Windows platform

2.1.1 Install the VS2017 release package

Run the sample program SKIrisDemo if you get an error like the following:



You'll need to install the redistributable packages for Visual Studio 2017, found on the downloads page

x86: vc_redist.x86.exe

x64: vc_redist.x64.exe

Download and install

3 SDK INTERFACE FUNCTIONS

3.1 IrisEng_initiate

Start the iris runtime environment and initialize the SDK, during this process, the user iris data under the data directory specified in the configuration parameters will be loaded into the SDK.

3.1.1 Function Definition

```
int IrisEng_initiate(const char *configuration,
                     IrisEventCB event_callback);
```

Parameter list:

Parameters	Explanation
configuration	Parameter setting string in Json format, see 3.1.3 for details
event_callback	Device event callback function, mainly to provide hotplug event reporting

Return value:

0 - success, non-0 - failure, see the return value definition in irisengine_export.h

Any other interface function that returns an int can refer to this return value

3.1.2 Sample code

```
char conf_buf[] =
"{\
        \"enroll_mode\":\"two_eye\",\
        \"identify_mode\":\"any_eye\",\
        \"find_same\":\"off\" \
}";
int ret = IrisEng_initiate(conf_buf, eventCallback);
if(ret != IRIS_ENG_OK) {
        getErrorInfo(ret, msg);
        MessageBox(msg, " initialization failed", MB_ICONERROR);
}
else {
        MessageBox(msg, " initialization success");
}
```

3.1.3 Generic setting parameters

Parameters	Range of values	Explanation
DIR_APP	Available directories	The root directory used by this SDK, for example: "DIR_APP":"E:\\temp" By default, this directory is %ProgramData%\Simbok\SKIris\
DIR_DATA	Available directories	The directory where the data will be stored,for example: "DIR_DATA":"E:\\temp\\data" By default, this directory is DIR_APP\data
DIR_EYEDATA	Available directories	The directory in which the iris data is stored, for example: "DIR_EYEDATA":"E:\\temp\\data\\eyedata" by default, this directory is DIR_DATA\data When "load_user_dir": "on", the SDK automatically loads all user data subdirectories from this directory. After the user registration is completed, the user data subdirectory will also be generated in this director
enroll_mode	two_eye left_eye right_eye	Register for both eyes, left eye and right eye
identify_mode	two_eye left_eye right_eye any_eye	Corresponding to both eyes, left eye, right eye, any eye
find_same	on/off	Specify whether to look for users with the same iris characteristics in the current user database when registering on- Look for duplicates, off- do not look
load_user_dir	on/off	on -- SDK automatically loads all user data subdirectories from DIR_EYEDATA directory into runtime memory on startup, this is the default

		off -- The SDK does not actively load the user data directory at startup. It is usually used when the upper application uses a database to manage user data, then the upper application can load the user data into the SDK by calling the IrisEng_add_user (see 3.10) interface.3.10
save_user_dir	on/off	Specifies whether the image and feature data should be saved in the user subdirectory under DIR_EYEDATA when the user is registered The default is on
save_snap_file	on/off	Specify whether the identified image should be saved in the DIR_EYEDATA/snapshot subdirectory when the user identification is successful The default is off

3.1.4 Error Code Definition

When the return value of the call interface is not 0, it means that an error has occurred. The specific error code and meaning are as follows:

Macro Definition	Value	Explanation
IRIS_ENG_OK	0	Success
IRIS_ENG_ERR_NO_DEV	1	No Aratek dedicated acquisition module was found to access the system
IRIS_ENG_ERR_DEV_CONNECT	2	Failed to connect to the acquisition module
IRIS_ENG_ERR_DEV_ILLEGAL	3	The connected acquisition module does not go through the normal factory Settings
IRIS_ENG_ERR_DEV_CAPTURE	4	Failed to start image acquisition
IRIS_ENG_ERR_DEV_HOTPLUG_ENABLE	6	Failed to initiate module hotplug monitoring
IRIS_ENG_ERR_DEV_FRAME_PARAM	11	Failed to set image frame parameters
IRISDEV_ERR_EU_CTRL	12	Failed to control IR /LED/ ranging of acquisition module
IRISDEV_ERR_EXPIRE_DEV	15	Expired acquisition mods
IRIS_ENG_ERR_NOT_INIT	20	The SDK has not been initialized

IRIS_ENG_ERR_ALREADY_INIT	21	SDK initialized (cannot be re-initialized)
IRIS_ENG_ERR_WRONG_CONFIG	22	Wrong configuration parameters (must be in valid json format)
IRIS_ENG_ERR_MAX_USER_NUM	23	Exceed the maximum number of users
IRIS_ENG_ERR_USER_ID_ILLEGAL	24	The user ID is not valid
IRIS_ENG_ERR_USER_EXIST	25	The user ID already exists
IRIS_ENG_ERR_NO_USER	26	No user exists
IRIS_ENG_ERR_ALREADY_RUN	27	Have been in some running state (registered/identified)
IRIS_ENG_ERR_USER_NOT_EXIST	28	The specified user ID does not exist
IRIS_ENG_ERR_SOUND_PLAY	30	Sound play failure
IRIS_ENG_ERR_SYSTEM	50	System-level errors (memory allocation, threading, etc.)

The error codes listed above apply to the return values of all SDK interfaces.

You can also refer to inc\irisengine_export.h in the SDK for detailed macro definitions

3.2 IrisEng_release

Release the iris runtime environment

3.2.1 Function Definition

```
int IrisEng_release();
```

3.2.2 Sample code

slightly

3.3 IrisEng_load_dev_params

Depending on the structure and application of different device modules, some special internal parameter values need to be used

The function is called immediately after IrisEng_initiate.

3.3.1 Function Definition

```
int IrisEng_load_dev_params(const char *params);
```

Parameter list:

Parameters	Explanation
params	Params parameter setting string in Json format, see 3.1.3 for details The first level key is always the device type name The content corresponds exactly to the param_dev.cfg file in the SDK package

3.3.1 Sample code

slightly

3.4 IrisEng_enroll

Register Iris users

3.4.1 Function Definition

```
int IrisEng_enroll(const char *user_id, int overwrite,
                   CallBack enroll_callback);
```

Parameter list:

Parameters	Explanation
user_id	Registered user ID
overwrite	1-- overwrite the existing user (whether or not the user already exists), 0-- Do not overwrite (return an error if the user already exists)
enroll_callback	The resulting callback function, see 3.8

3.4.2 Sample code

```
    void callBack(const char* user_id, iris_callback_result *result)
    {
        char msg[1024];

        sprintf(msg, "result:%d, eye:%d, finish:%d", result->result,
result->eye, result->finish);
```

```
        std::cout << msg << std::endl;

        if(result->finish) {
            switch(state) {
            case ENROLL:
                if(result->result == IRIS_RESULT_OK) {
                    sprintf(msg, "%s enrolled", user_id);
                }
                else if(result->result == IRIS_RESULT_FAIL) {
                    sprintf(msg, "find user '%s' with same iris trait",
user_id);
                }
                else if(result->result == IRIS_RESULT_TIMEOUT) {
                    sprintf(msg, "enroll timeout");
                }
                state = NOTHING;
                break;

            case IDENTIFY:
            case IDENTIFY2:
                if(result->result == IRIS_RESULT_OK) {
                    sprintf(msg, "%s identified", user_id);
                }
                else if(result->result == IRIS_RESULT_FAIL) {
                    sprintf(msg, "no user identified");
                }
                else if(result->result == IRIS_RESULT_FAIL_LR_MATCH) {
                    sprintf(msg, "find different user with same
left/right iris");
                }
                else if(result->result == IRIS_RESULT_TIMEOUT) {
                    sprintf(msg, "identify timeout");
                }
                if(state == IDENTIFY) {
                    state = NOTHING;
                }
                break;

            default:
                break;
            }
            std::cout << msg << std::endl;
        }
    }


    int ret = IrisEng_enroll( strName, 1, myCallback );
    if (IRIS_ENG_OK != ret) {
```

```
        return;
    }
```

3.5 IrisEng_fetch_enroll_data

Pull the user data that has just completed enrollment from the SDK cache

3.5.1 Function Definition

```
int IrisEng_fetch_enroll_data( const char *user_id,
                               char *data,
                               int len);
```

Parameter list:

Parameters	Explanation
user_id	NULL- identification function Not NULL- A validation function for that user_id
data	Pre-allocated address space to store fetched user data
len	Len the size of the address space pointed to by data

3.5.2 Sample code

```
const char* user = "aaa";
int data_len = IrisEng_get_enroll_data_len(user);
char* data = (char*)malloc(data_len+1);
int ret = IrisEng_fetch_enroll_data(user, data, data_len+1);
```

3.6 IrisEng_identify

Enable iris recognition

3.6.1 Function Definition

```
int IrisEng_identify(const char *user_id, int continuous,
                     CallBack identify_callback);
```

Parameter list:

Parameters	Explanation
user_id	NULL - Used for identification Not NULL - Used for validation of this user_id
continuous	1- continuous identification, 0- single identification

identify_callback	The resulting callback function, see 3.8

3.6.2 Sample code

```
int ret = IrisEng_identify( NULL, 1, myCallback );
if (IRIS_ENG_OK != ret) {
        getErrorInfo(ret, msg);
        MessageBox(msg, " recognition failed ", MB_ICONERROR);
        return;
}
```

3.7 IrisEng_fetch_snap_data

Fetches the user data from the SDK cache that has just been successfully identified

3.7.1 Function Definition

```
int IrisEng_fetch_snap_data(char *data, int len);
```

3.7.2 Sample code

```
int data_len = IrisEng_get_snap_data_len();
char* data = (char*)malloc(data_len+1);
int ret = IrisEng_fetch_snap_data(data, data_len+1);
```

3.8 CallBack

The resulting callback function to register/identify

3.8.1 Function Definition

```
typedef void (*CallBack)(const char* user_id,
                         iris_callback_result *cb_result);
```

Parameter list:

Parameters	Explanation
user_id	Registered/identified user ID
cb_result	result: Indicates the result, 0 is success. See the IRIS_RESULT definition in irisengine_export.h for details eye: 0- left eye, 1- right eye finish: Indicates if this callback is the last callback registered/identified

	For example: 1. When both eyes are registered, 2 callbacks occur and the result could be: result:0, eye:0,finish:0 result:0, eye:1,finish:1 It means that both left and right eyes are successfully registered 2. Only one callback occurs for either eye recognition, and the result could be: result:0, eye:1,finish:1 It means that the right eye was recognized successfully

3.8.2 Instructions

Do not perform long task processing in this callback, and do not call interfaces such as stop or start of registration/identification.

3.9 IrisEng_stop

Stop running iris registration/recognition

3.9.1 Function Definition

```
int IrisEng_stop();
```

3.9.2 Sample code

slightly

3.10 IrisEng_add_user

Add user data dynamically to the SDK running memory.

In general, this interface is used when load_user_dir is off.

3.10.1 Function Definition

```
int IrisEng_add_user(const char *user_id,
```

```
                                const unsigned char *left_iris,
                                const unsigned char *right_iris);
```

Parameter list:

Parameters	Explanation
user_id	The user ID you added
left_iris	Left iris feature, which can be NULL
right_iris	Right iris feature, which can be NULL

3.10.2 Sample code

slightly

3.11 IrisEng_fetch_user

Fetches the user data from the SDK running memory

3.11.1 Function Definition

```
int IrisEng_fetch_user(const char *user_id,
                       bool *left_exist, unsigned char *left_iris,
                        bool *right_exist, unsigned char *right_iris);
```

Parameter list:

Parameters	Explanation
user_id	The user ID to be fetched
left_exist	Returns whether the left eye data exists
right_exist	Returns whether the right eye data exists
left_iris	Left iris feature
right_iris	Right iris feature

3.11.1 Sample code

slightly

3.12 IrisEng_delete_user

Delete the user data from the SDK

3.12.1 Function Definition

```
int IrisEng_delete_user(const char *user_id, int dir_removed);
```

Parameter list:

Parameters	Explanation
user_id	The deleted user ID
dir_removed	1-- Remove the user's data directory, 0-- do not remove the directory Data directory: The user's subdirectory under the DIR_EYEDATA directory in 3.1.3

3.12.2 Sample code

slightly

3.13 IrisEng_delete_all_user

Delete all user data from the SDK

3.13.1 Function Definition

```
int IrisEng_delete_all_user(int dir_removed);
```

Parameter list:

See instructions in 3.12.1

3.13.2 Sample code

slightly

3.14 IrisEng_get_user_dir

Get the directory where the user's iris data is located.

This directory can be specified by the upper application at initialization time through the DIR_EYEDATA configuration option. If it is not specified, the SDK will use the default directory and get the directory through this interface

3.14.1 Function Definition

```
int IrisEng_get_user_dir(char* user_dir);
```

Parameter list:

Parameters	Explanation
user_dir	Returns the directory of the user's data (i.e., EYEDATA) When passed as a parameter, this parameter points to the memory address allocated in advance

3.14.2 Sample code

slightly

3.15 IrisEng_get_user_list

Get the number of users and the entire user list string.

3.15.1 Function Definition

```
int IrisEng_get_user_list(int *user_num, char* list);
```

Parameter list:

Parameters	Explanation
user_num	Returns the number of users contained in the users directory
list	Returns a comma-separated list of all users as a string, e.g. "a1,a2,a3"

3.15.2 Sample code

```
char* l_user_list = NULL;
int l_user_num;

IrisEng_get_user_list(&l_user_num, NULL);
if(l_user_num == 0)
    return;
l_user_list = (char*)calloc(l_user_num, IRIS_USER_ID_LENGTH);
irisEng_get_user_list(&l_user_num, l_user_list);
```

3.16 IrisEng_change_configure

Some individual parameter configurations can be dynamically adjusted while the SDK is running.

3.16.1 Function Definition

```
int IrisEng_change_configure(const char *configuration);
```

Parameter list:

See 3.1.3

3.16.2 Sample code

slightly

3.17 IrisEng_get_image_size

Iriseng_get_image_size Gets the pixel width and height of the iris image.

3.17.1 Function Definition

```
int IrisEng_get_image_size(int *height, int *width);
```

Parameter list:

Parameters	Explanation
height	Iris image height (in pixels)
width	Iris image width (in pixels)

3.17.2 Sample code

slightly

3.18 IrisEng_set_preview2

设置图像实时预览相关的参数

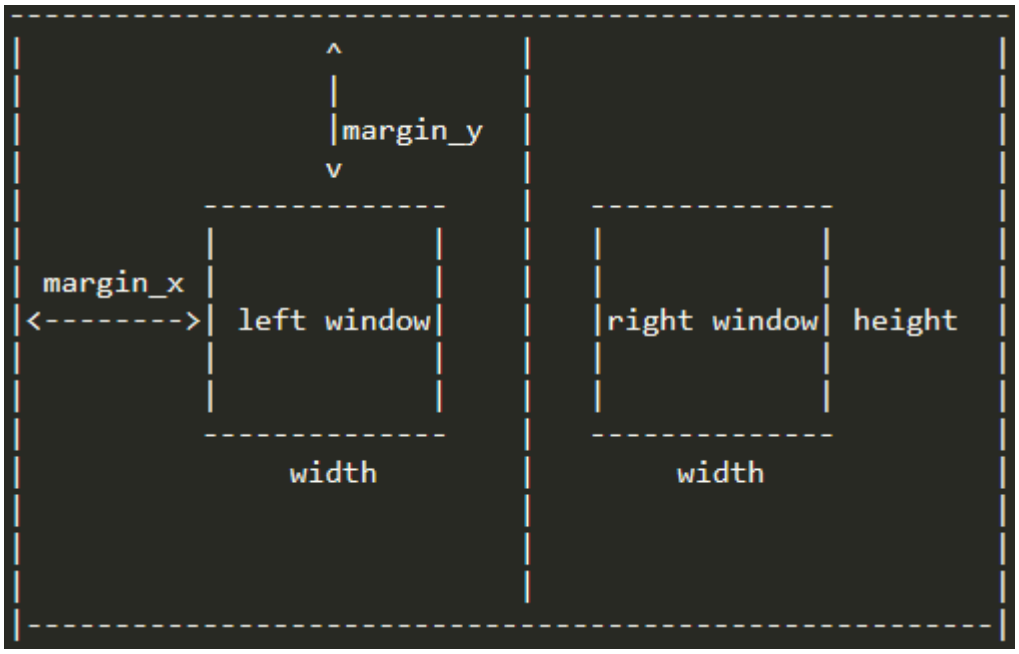
3.18.1 Function Definition

```
int IrisEng_set_preview2(const char* mode,
                         IMAGE_PIXEL_FORMAT format,
                         int width, int height,
                         int margin_x, int margin_y,
```

```
                                PreviewCB callback);
```

Parameter list:

Parameters	Explanation
mode	Select the region where the image will be output "whole" -- Complete image region Output the whole image area and scale it by the specified width/height "custom"-- Customize the region Output a stitched image of the central regions of the left and right eyes
format	IMAGE_PIXEL_RGB565 = 0, IMAGE_PIXEL_RGB888 = 1, IMAGE_PIXEL_ARGB8888 = 2, IMAGE_PIXEL_RAW = 3, // directly use the raw image data captured by the module, such as JPEG. In general, this format is not used
width	When used for "custom", truncate the width of the area for the left/right eye. For "whole", the actual width of the output image after scaling
height	For "custom", truncate the height of the area for the left/right eye. For "whole", the actual height of the output image after scaling
margin_x	Used for "custom" to take the horizontal distance between the left edge of the cut area and the leftmost of the overall image for the left eye (or the right edge of the cut area for the right eye and the rightmost of the overall image). -1 indicates center
margin_y	Used for "custom", the vertical distance between the top edge of the truncated region and the top edge of the overall image. -1 indicates center
callback	For a preview of callbacks, see 3.19



3.18.2 Sample code

slightly

3.19 CallBack

Preview the callback function

3.19.1 Function Definition

```
typedef void (* PreviewCB)(const unsigned char* data, int len,
                           int width, int height, int flag);
```

Parameter list:

Parameters	Explanation
data	Image data pointer
len	Image data length
width	Image width
height	Image height
flag	Image marker -1 -- Global image 0 -- Left eye 1 -- Right eye

3.19.2 Instructions

Do not perform long tasks in this callback, and do not call registered/identified stop or start interfaces.

3.20 IrisEng_get_device_info

Get iris device information

3.20.1 Function Definition

```
int IrisEng_get_device_info(IrisDeviceInfo *dev_info);
```

Where IrisDeviceInfo is defined as follows:

```
typedef struct {
    char fw_ver[32];                // firmware version
    char fw_date[32];               // firmware date
    char manufacturer[32];          // equipment manufacturer
    char logo[32];                  //LOGO
    char sn[32];                    // device serial number
    unsigned int max_user_num;      // maximum number of users allowed
on the device
} IrisDeviceInfo;
```

3.20.2 Sample code

slightly

3.21 IrisEng_get_device_info_2

Get iris device information

3.21.1 Function Definition

```
int IrisEng_get_device_info_2(char* dev_info, int len)
```

The dev_info contains basically the same content as the above IrisDeviceInfo, but the format is json format, which is convenient for the expansion or adjustment of subsequent information

3.21.2 Sample code

slightly

3.22 IrisEng_get_algor_version

Get the current algorithm version and release date

3.22.1 Function Definition

```
int IrisEng_get_algor_version(char *version, char* date)
```

Parameter list:

Parameters	Explanation
version	Algorithm Version Three paragraphs, such as: "2.3.1"
date	Date of algorithm release For example: "2022-03-01"

3.22.2 Sample code

slightly