User Guide

FaceView

Artificial Intelligence
Facial Recognition APIs

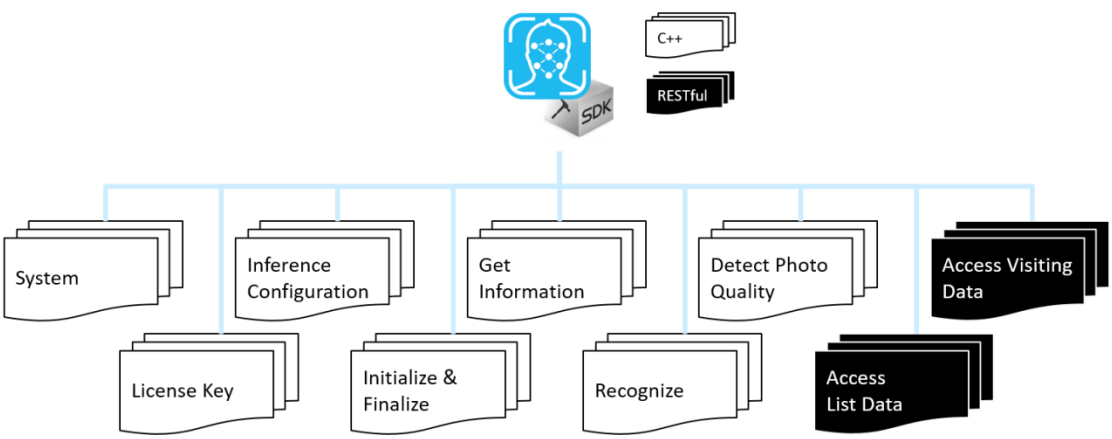
Revision History

Date	Version	Author	Reviewer	Remark
2020-05-22	1.0.3	Gary70.Lin	Alan.Kao	
2020-10-28	1.0.4	Felicity.Lin	Jasonjh.Huang	Add RESTful APIs
2020-11-19	1.0.5	Felicity.Lin	Gary70.Lin	Function block update
2020-12-08	1.0.6	Alan.Kao	Alan.Kao	Add Sample and Error code

Table of Contents

1. Introduction

Powered by CyberLink's FaceMe®, an industry-leading facial recognition engine, Advantech's FaceView application provides precise and scalable real-time facial recognition for various AIoT applications in the retail, hospitality, and public safety fields. Advantech also provides an easy-to-integrate SDK for rapid integration with existing systems using APIs. FaceView SDK offers 7 categories of C++ APIs and 2 categories of RESTful APIs, and will be introduced one by one in the following sections.



2. System

HRESULT FV_GetVersionInto ()

HRESULT FV_GetVersionInto (std::string *info);

【Descriptions】
To get FaceView library version.

【Return】
Return an HRESULT. Please refer to enum Err_code in Error Code Section or in common_def.h file.

【Sample】

```
std::string ver = "";
FV_GetVersionInto(&ver);
```

void FV_CreateInstance ()

void FV_CreateInstance ();

【Descriptions】
To create a FaceView SDK instance.

【Sample】

```
std::string ver = "";
FV_CreateInstance (&ver);
```

3. License Key

HRESULT FV_HWKey ()

HRESULT FV_HWKey ();

【Note】
Useful only for HW bundle version which license key input is unnecessary.

【Descriptions】
To adopt hardware bundle certification.
Before activating hardware bundle license, it's used to get the embedded license key which is workable only for Advantech devices.

【Return】
Return an HRESULT. Please refer to enum Err_code in Error Code Section or in common_def.h file.

HRESULT FV_SWKey ()

HRESULT FV_SWKey (const unsigned char *keydata, uint16_t len);

【Note】
Useful only for SW distribution version which license key input is necessary.

【Descriptions】
Before license certification, use this function to input and keep license key.

【Input】
License key and its length.

【Return】
Return an HRESULT. Please refer to enum Err_code in Error Code Section or in common_def.h file.

【Sample】

```
int Demo_UtilityDlg::SDK_SetLicense()
{
	if (m_Sdk_Ptr == NULL)
		m_Sdk_Ptr = FV_GetInstance();

	int res = 0;

#ifdef _HWK_
	res = FV_HWKey();
#else
	res = FV_SWKey((const unsigned char*)key,strlen(key));
#endif

	return res;
}
```

HRESULT FV_KeyStart ()

HRESULT FV_KeyStart ();

【Descriptions】

To activate license by the kept license key. The key can be the embedded license key for hardware bundle or input additionally.

【Return】

Return an HRESULT. Please refer to enum Err_code_license in Error Code Section or in common_def.h file.

【Sample】

```
int Demo_UtilityDlg::SDK_LicenseStart()
{
	int val = 0;

	if (m_Sdk_Ptr == NULL)
		m_Sdk_Ptr = FV_GetInstance();

	return FV_KeyStart();
```

```
}
```

HRESULT FV_KeyDeactive ()

```
HRESULT FV_KeyDeactive ();
```

【Descriptions】

To deactivate the used license key and then be able to use another license key on the same device.

【Return】

Return an HRESULT. Please refer to enum Err_code_license in Error Code Section or in common_def.h file.

【Sample】

```
int Demo_UtilityDlg::SDK_LicenseDeactive()
{
	if (m_Sdk_Ptr == NULL)
	{
		AfxMessageBox(L"Please initial SDK");
		return -1;
	}
	return FV_KeyDeactive();
}
```

4. Inference Configuration

HRESULT FV_SysConfig ()

```
HRESULT FV_SysConfig (const char* bundle_path);
```

【Descriptions】

To declare the location path of your application for library authentication.

【Return】

Return an HRESULT. Please refer to enum Err_code in Error Code Section or in common_def.h file.

【Sample】

```
int res;

CString path;
GetModuleFileName(NULL, path.GetBufferSetLength(MAX_PATH + 1), MAX_PATH);
path.ReleaseBuffer();
int pos = path.ReverseFind('\\');
path = path.Left(pos);

string moduledir_path = CT2A(path);

res = FV_SysConfig(moduledir_path.data());
```

HRESULT FV_UseDefaultParams ()

HRESULT FV_UseDefaultParams ();

【Descriptions】

To load default parameters of FaceView inference engine without step by step setting.

【Return】

Return an HRESULT. Please refer to enum Err_code in Error Code Section or in common_def.h file.

HRESULT FV_ThresholdLevel ()

HRESULT FV_ThresholdLevel (int32_t level);

【Descriptions】

To set precision level (FAR: False Accept Rate) into FaveView inference engine.

enum precision

```
{
    lvlE6 = 0,
    lvlE5,     // lvlE5 : error rate = 0.00001
    lvlE4,     // lvlE4 : error rate = 0.0001
    lvlE3,     // lvlE3 : error rate = 0.001
    lvlE2     // lvlE2 : error rate = 0.01
};
```

【Return】

Return an HRESULT. Please refer to enum Err_code in Error Code Section or in common_def.h file.

HRESULT FV_DetectionOutputOrder ()

HRESULT FV_DetectionOutputOrder (uint32_t order);

【Descriptions】

To set output order of recognition results.

```
enum output_order
{
    fv_NO_ORDER = 0,          // Detection output order by location
    fv_ORDER_CONFIDENCE,      // Detection output order by confidence
    fv_ORDER_FACE_WIDTH       // Detection output order by face width
};
```

【Return】

Return an HRESULT. Please refer to enum Err_code in Error Code Section or in common_def.h file.

HRESULT FV_MinFace ()

HRESULT FV_MinFace (uint32_t width);

【Descriptions】

To set minimum face width to be identified. The suggested arrange is between

40~240 in pixels.

【Return】

Return an HRESULT. Please refer to enum Err_code in Error Code Section or in common_def.h file.

【Sample】

Ret = FV_MinFace(40);

HRESULT FV_MaxFace ()

HRESULT FV_MaxFace (uint32_t width);

【Descriptions】

To set maximum face width to be identified. The maximum width must be bigger than the minimum one.

【Return】

Return an HRESULT. Please refer to enum Err_code in Error Code Section or in common_def.h file.

【Sample】

Ret = FV_MaxFace(400);

HRESULT FV_ExtractOption ()

HRESULT FV_ExtractOption (int32_t option);

【Descriptions】

To set required features to be extracted.

```
enum fv_FEATURE_OPTIONS
    {
        FEATURE_OPTION_NONE = 0,
        FEATURE_OPTION_BOUNDING_BOX = (1LL << 1),          //00000010
        FEATURE_OPTION_FEATURE_LANDMARK = (1LL << 2),      //00000100
        FEATURE_OPTION_FEATURE = (1LL << 3),               //00001000
        FEATURE_OPTION_EMOTION = (1LL << 4),               //00010000
```

```
        FEATURE_OPTION_AGE = (1LL << 5),                    //00100000
        FEATURE_OPTION_GENDER = (1LL << 6),                 //01000000
        FEATURE_OPTION_POSE = (1LL << 7),                   //10000000
        FEATURE_OPTION_ALL = INT32_MAX,                     //11111111
    };
```

【Return】

Return an HRESULT. Please refer to enum Err_code in Error Code Section or in common_def.h file.

HRESULT FV_RecognizeConfig ()

HRESULT FV_RecognizeConfig (int32_t prefer, int32_t detect_level, uint16_t detect_threads, int32_t extract_level, uint16_t extract_threads);

【Descriptions】

To set preference mode, detection level, extraction level and number of threads/VPUs. The number of threads/VPUs will depend on your system spec.

```
enum fv_detection_lvl
{
        fv_d_LEVEL_DEFAULT = 0,   // Use default detection level
        fv_d_LEVEL_HIGH,          // Use high detection level
        fv_d_LEVEL_ULTRA_HIGH,    // Use ultra-high detection level
};

enum fv_extraction_lvl
{
        fv_e_LEVEL_DEFAULT = 0,   // Use default extraction level
        fv_e_LEVEL_STANDARD,      // Use standard extraction level
        fv_e_LEVEL_HIGH,          // Use high extraction level
        fv_LEVEL_ULTRA_HIGH,      // Use ultra-high extraction level
        fv_LEVEL_HIGH_ASIAN,      // Use high for asian extraction level
        fv_e_VERY_HIGH            // Use very high extraction level
};

enum Preference
```

```
{
        PREFER_NONE = 0,
        PREFER_HARDWARE_DETECTION = (1 << 1),
        PREFER_FAST_DETECTION = (1 << 2),
        PREFER_HARDWARE_EXTRACTION = (1 << 3),
        PREFER_FAST_EXTRACTION = (1 << 4),
        PREFER_INTEL_MOVIDIUS_VPU_DETECTION = (1 << 5),
        PREFER_INTEL_MOVIDIUS_VPU_EXTRACTION = (1 << 6)
};
```

【Return】

Return an HRESULT. Please refer to enum Err_code in Error Code Section or in common_def.h file.

5. Initialize and Finalize

HRESULT FV_InitialSDK ()

HRESULT FV_InitialSDK ();

【Descriptions】
To initialize FaceView SDK after activating license and setting configuration.

【Return】
Return an HRESULT. Please refer to enum Err_code in Error Code Section or in common_def.h file.

HRESULT FV_InitEngine ()

HRESULT FV_InitEngine ();

【Descriptions】
To initialize recognition inference engine after initializing SDK.

【Return】
Return an HRESULT. Please refer to enum Err_code in Error Code Section or in common_def.h file.

HRESULT FV_FinalizeSDK ()

HRESULT FV_FinalizeSDK ();

【Descriptions】
To release FaceView SDK resource and close it.

【Return】
Return an HRESULT. Please refer to enum Err_code in Error Code Section or in

common_def.h file.

6. Get Information

HRESULT FV_GetMinFace ()

HRESULT FV_GetMinFace (uint32_t *value);

【Descriptions】
To get current setting of minimum face width.

【Return】
Return an HRESULT. Please refer to enum Err_code in Error Code Section or in common_def.h file.

【Sample】
```
unit32_t minFace;
FV_GetMinFace(&minFace);
```

HRESULT FV_GetMaxFace ()

HRESULT FV_GetMaxFace (uint32_t *value);

【Descriptions】
To get current setting of maximum face width.

【Return】
Return an HRESULT. Please refer to enum Err_code in Error Code Section or in common_def.h file.

【Sample】
```
unit32_t maxFace;
FV_GetMaxFace(&maxFace);
```

HRESULT FV_GetOutputOrder ()

HRESULT FV_GetOutputOrder (uint32_t *value);

【Descriptions】

To get current setting of recognition output order.

【Return】

Return an HRESULT. Please refer to enum Err_code in Error Code Section or in common_def.h file.

【Sample】

```
unit32_t OutputOrder;
FV_GetOutputOrder(&OutputOrder);
```

HRESULT FV_GetThesholdValue ()

```
HRESULT FV_GetThesholdValue (float *value);
```

【Descriptions】

To get the threshold value of FAR.

【Return】

Return an HRESULT. Please refer to enum Err_code in Error Code Section or in common_def.h file.

【Sample】

```
float ThesholdValue;
FV_GetThesholdValue(&ThesholdValue);
```

7. Recognize

HRESULT FV_ExtractNumberOfFace ()

HRESULT FV_ExtractNumberOfFace (cv::Mat image, uint32_t *number);

【Descriptions】

To get number of faces in the input image. The recognition process will take this API to identify how many faces in the given image.

【Input】

One video frame in in terms s of cv::Mat category.

【Output】

uint32_t *number

To get the number of faces within the given image. Later on it can be used as input of FV_GetFaceExtractedInfo() API to determine how many faces and related features will be further extracted.

【Return】

Return an HRESULT. Please refer to enum Err_code in Error Code Section or in common_def.h file.

【Sample】

```
uint32_t number = 0;
Mat image = frame.clone();
int ret = FV_ExtractNumberOfFace(image, &number);
```

HRESULT FV_GetFaceExtractedInfo ()

HRESULT FV_GetFaceExtractedInfo (uint32_t number, std::vector<UserFaceItem> *pFaceRecognizedInfos);

【Descriptions】

To get all identified face feature data.

【Input】

The required number of extracted faces to output their features. The maximum will depend on the return value of FV_ExtractNumberOfFace.

【Output】

Class UserFaceItem can also be found in fv_face.h and it defines the required properties for a recognized face.

```
class UserFaceItem
{
    public:

        FV_ADV::fv_FaceInfo faceInfo;
        FV_ADV::fv_FaceAttribute faceAttribute;
        FV_ADV::fv_FaceFeature faceFeature;
        FV_ADV::fv_FaceLandmark face_landmark;
        std::string name;
};
```

- *faceInfo* defines the auxiliary information, like a face bounding box.
- *faceAttribute* defines the recognized face information: age, gender, emotion and pose (angles).
- *faceFeature* defines the specific facial data to be used in face comparison.
- *face_landmark* defines the landmark data representing eyes, nose and mouth positions.

【Return】

Return an HRESULT. Please refer to enum Err_code in Error Code Section or in common_def.h file.

HRESULT FV_CompareFaces ()

HRESULT FV_CompareFaces (const fv_FaceFeature *compare_faceFeatureA, const fv_FaceFeature *compare_faceFeatureB, float *confidence);

【Descriptions】

To compare two faces and then obtain a confidence value. A higher confidence value means more similar between both of them.

【Input】

Two faces to be compared and each one is further represented as the fv_FaceFeature property of UserFaceItem class.

【Output】

A confidence value which directly reflects the similarity between both compared faces.

【Return】

Return an HRESULT. Please refer to enum Err_code in Error Code Section or in common_def.h file.

8. Detect Photo Quality

HRESULT FV_DetectQuality ()

HRESULT FV_DetectQuality (cv::Mat image, const fv_QualityDetectConfig *config, fv_QualityDetectResult *result);

【Descriptions】

To evaluate image quality according to the setting of required items.

【Return】

Return an HRESULT. Please refer to enum Err_code in Error Code Section or in common_def.h file.

【Sample】

```
std::vector<FV_ADV::fv_QualityDetectResult> res;
        FV_ADV::fv_QualityDetectConfig conf;

        FV_INIT_STRUCT(&conf, fv_QualityDetectConfig);
        conf.detectType = QUALITY_ISSUE_OPTION_ALL;
        conf.blurDetectMode = BLUR_CAMERA_MODE;
        conf.checkMode = QUALITY_CHECK_MODE_ALL_FAILURE;

        conf.overExposureThreshold = 180;
        conf.underExposureThreshold = 40;


        conf.faceCount = length;
        conf.faceInfos = &face_infos[0];
        conf.faceLandmarks = &face_lands[0];
        conf.poses = &poses[0];

        int ret = FV_DetectQuality(image, &conf, &res[0]);
```

【Structure Define】

```
#define FV_INIT_STRUCT(ptr, class_name) { std::memset((uint8_t *)ptr, 0,
sizeof(class_name)); (*ptr).sizeOfStructure = sizeof(class_name); }

struct fv_QualityDetectConfig
{
        uint32_t sizeOfStructure;        // The size of the structure.
        int32_t detectType;              // The detect type.
        int32_t checkMode;               // The mode of photo quality check.
        int32_t blurDetectMode;          // Different detect config for blur
detection
        int32_t overExposureThreshold;   // The threshold of over
exposure.
        int32_t underExposureThreshold;  // The threshold of under
exposure.
        uint32_t faceCount;              // Count of faces. When
faceCount > 0, faceInfos, faceLandmarks and poses cannot be nullptr.
        fv_FaceInfo* faceInfos;          // The face information for
analyzed.
        fv_FaceLandmark* faceLandmarks;  // The feature landmarks
for analzed.
        _Pose* poses;                    // The face poses for
analyzed.
};

struct fv_QualityDetectResult
{
        uint32_t sizeOfStructure;
//!< The size of the structure.
        int32_t issue;                                  //!< The
detect result.
        float blur;                                     //!< The
value of blurriness.
        float exposure;                                 //!< The
value of exposure.
        uint32_t faceSize;                              //!< The
face size. (pixel)
```

```
                    _Pose wrongAngle;                                   //!< The
wrong face angle.
                    int32_t occlusionReason;          //!< @see
EFR_OCCLUSION_FAIL_REASON.
          };
```

9. Access List Data (RESTful)

[Post] Add a user

URL:

localhost:2211/FV/DBService/AddRecord/ww

Post BODY format example:

```
{
"Bindid":"ABC-001", # your unique id
"alias":"Mary Lin", #register user name
"facedata":" facedata from FV_ADV::fv_FaceFeature in base64 string",
"facetype":4, #facetype (an integer)from FV_ADV::fv_FaceFeature
"subtype":0, #faceSubtype (an integer)from FV_ADV::fv_FaceFeature
"f_size":2064, #facesize (an integer)from FV_ADV::fv_FaceFeature
"picture":"jpg binary in base64 string",
"apvalue":0,
"apstring":"string" #your remark
}
```

Reply:

Success : Add XXX done

Fail : errorcode-1 (lack of Bindid/unique id)

Fail : errorcode-2 (lack of facedata)

[Get] Get amount of users

URL :

localhost:2211/FV/DBService/ReadRecord_amount/

Reply: integer (obtaining amount)

[Get]　Get userlist　(display with bindid)

URL:

localhost:2211/FV/DBService/ReadRecord/userlist/StartIndex/EndIndex

StarIndex : integer > 0,

it will reply a userlist that index >= startindex & index < EndIndex.

Request example

localhost:2211/FV/DBService/ReadRecord/userlist/1/300

Reply format example


```
[
  "AS-02",
  "AS-03",
  "AS-06",
  "AS-08",
  "AS-16",
  "AS-17"
]
```


[Get]　Get user's base information

URL:

localhost:2211/FV/DBService/ReadRecord_information/AS-16

（AS-16 is the query bindid/uniqueid）

Reply format example


```
{
  "bind_id": "AS-16",
  "alias": "piggy chu",
  "age": 45,
  "gender": 1,
  "apvalue": 0,
  "apstring": "this is a tester"
}
```

[Get]　Get user’s face information

URL:

localhost:2211/FV/DBService/ReadRecord_faces/AS-16

（AS-16 is the query bindid/uniqueid）

Reply format example:

No data:


```
{
  "Num": 0,
  "Data": null
}
```


Have data:


```
{
  "Num": 1,
  "Data": [
    [
      {
        "faceid": "36a0b4f0-b13b-11ea-a740-25d9e756866f",
        "facedata": "ccccccccdaeweweqshjq271bnslwqiw=wq27=",
        "facetype": 2,
        "facesubtype": 0,
        "fsize": 2064,
        "thumbnail": "2o312t3oajeerwioryw87=rerwioerr=="
      }
    ]
  ]
}
```

[Post] Update alias

URL:
localhost:2211/FV/DBService/UpdateRecord/alias/AS-16/newname
（AS-16 is the query bindid/uniqueid）

Reply:
Success: update ok
Fail: -5

[Post] Update gender

URL:
localhost:2211/FV/DBService/UpdateRecord/gender/AS-16/value
（AS-16 is the query bindid/uniqueid.）
value: 1, means male.
value: 2, means female.

Reply:
Success: update ok
Fail: -5

[Post] Update age

URL:
localhost:2211/FV/DBService/UpdateRecord/age/AS-16/value
（AS-16 is the query bindid/uniqueid; value is the new age.）

Reply:
Success: update ok
Fail: -5

[Post] Update apstring

URL:

localhost:2211/FV/DBService/UpdateRecord/apstring/AS-16/newstring

（AS-16 is the query bindid/uniqueid; newstring is the new information.）

Reply:

Success: update ok

Fail: -5

[Post] Update apvalue

URL:

localhost:2211/FV/DBService/UpdateRecord/apvalue/AS-16/apvalue

（AS-16 is the query bindid/uniqueid; apvalue is the new value.）

Reply:

Success: update ok

Fail: -5

[Post] Update facedata

URL: localhost:2211/FV/DBService/UpdateRecord/picture/498a2b00-b13b-11ea-a740-25d9e756866f
(498a2b00-b13b-11ea-a740-25d9e756866f is sample faceid.)

BODY format sample


```
{
  "facedata":"asqwqwa1113iu4u3l1ue=r23r=34i231odwqllq",
  "facetype": 2,
  "subtype":2,
  "f_size":4096,
  "picture":"oq3iu4hkqhk3=24?=weq34ug1gvszvczqiqiqhqwoeqoiweyg1k"
}
```


[DELETE] Delete a user

URL:
localhost:2211/FV/DBService/DeleteRecord/AS-17
（AS-17 is the query bindid/uniqueid）

Reply:

Success: delete done

Fail: -5

10. Access Visiting Data (RESTful)

Query

URL:
localhost:2468/FV/History/Query/StartTime/Endtime

Query the history in the time interval StartTime to EndTime.
StartTime and EndTime are unix timestamp.

For example:

<StartTime>
GMT+8：2020-10-26 10:35:30
unix timestamp：1603679730000000 (us)

GMT+8：2020-10-26 10:40:0
unix timestamp：1603680000000000 (us)

And then the query is:
localhost:2468/FV/History/Query/1603679730000000/1603680000000000

Example reply:


```
{
  "num": 6,
  "content": [
    {
      "alias_": "Lily Chen",
      "binduid_": "b1e40639-d1b3-42fe-b73d-a7690f069596",
      "type": "VIP",
      "in_out": "0"
    },
    {
      "alias_": "Lily Chen",
      "binduid_": "b1e40639-d1b3-42fe-b73d-a7690f069596",
      "type": "VIP",
```

"in_out": "1"
},
{
"alias_": "Lily Chen",
"binduid_": "b1e40639-d1b3-42fe-b73d-a7690f069596",
"type": "VIP",
"in_out": "0"
},
{
"alias_": "Lily Chen",
"binduid_": "b1e40639-d1b3-42fe-b73d-a7690f069596",
"type": "VIP",
"in_out": "1"
},
{
"alias_": "Nick da",
"binduid_": "35cfdd9b-1969-4586-9a0c-f1f8340eee11",
"type": "BlockList",
"in_out": "1"
},
{
"alias_": "Nick da",
"binduid_": "35cfdd9b-1969-4586-9a0c-f1f8340eee11",
"type": "BlockList",
"in_out": "0"
}
]
}

11. Error Code

List all error code for reference

Err_Code

```
{
        Err_OK = 0,                     // No error
        Err_fail,                       // Fail
        Err_LicenseInst_fail,           // License instance fail
        Err_RecognizerInst_fail,        // Recognizer instance fail
        Err_DataMagagerInst_fail,       // Data manager instance fail
        Err_QualityDetectInst_fail,     // Quality detect instance fail
        Err_UAInst_fail,                // UA instance fail
        Err_LicenseInit_fail,           // License initial fail
        Err_RecognizerInit_fail,        // Recognizer initial fail
        Err_DataMagagerInit_fail,       // Data manager initial fail
        Err_QualityDetectInit_fail,     // Quality detect initial fail
        Err_UAInit_fail,                // UA initial fail
        Err_SetMinFace_fail,            // Set minimum face size fail
        Err_SetMaxFace_fail,            // Set maximum face size fail
        Err_SetOrder_fail,              // Set output order fail
        Err_SetPrecision_fail,          // Set precision level fail
        Err_FaceInfo_cfg,               // Error in face information
        Err_FaceAttr_cfg,               // Error in face attribute
        Err_FaceFeature_cfg,            // Error in face feature
        Err_FaceLandmark_cfg,           // Error in face landmark

        Err_SDK_Fail = 98,
        Err_Already_init = 99,

        Err_NullPtr = 101
    };
```

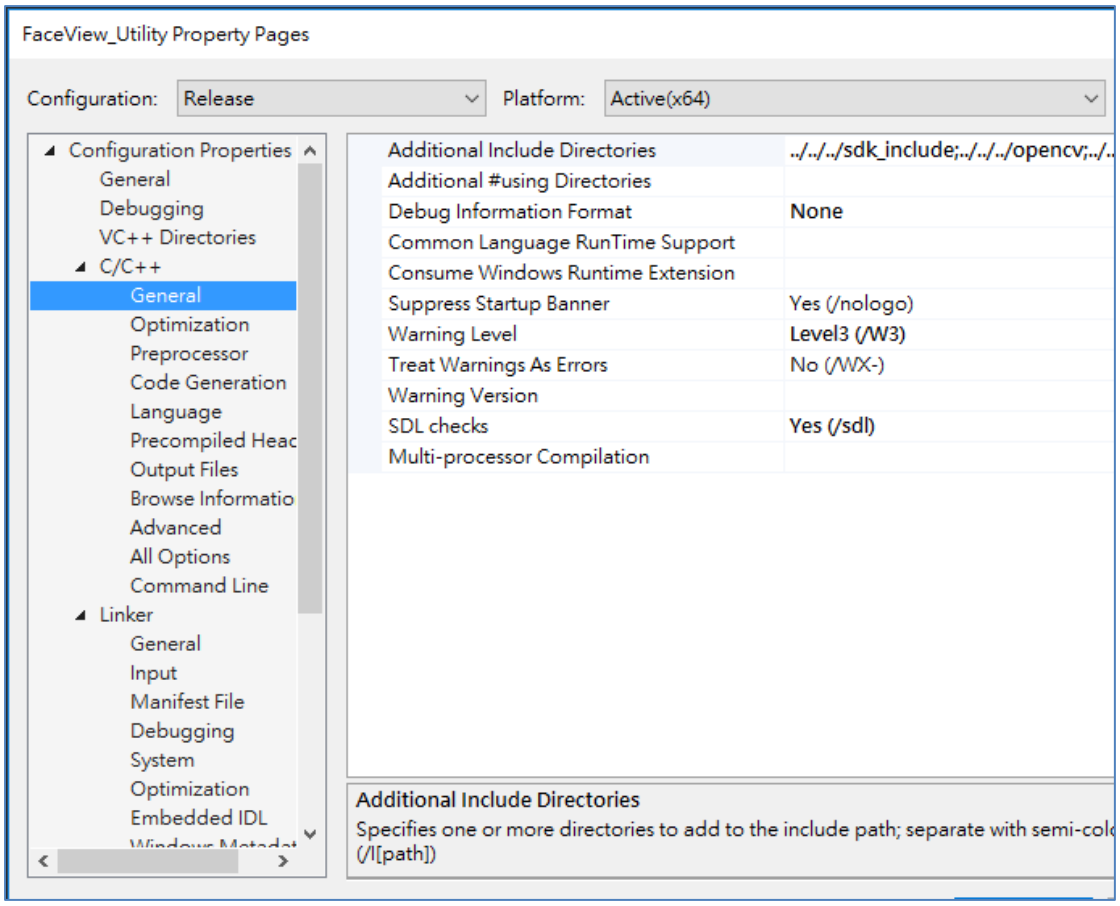
Err_code_license

```
{
        Err_L_Register = 50,
        Err_L_Deactive,         // Error in license deactivate
        Err_L_Renew             // Error in license re-new
    };
```

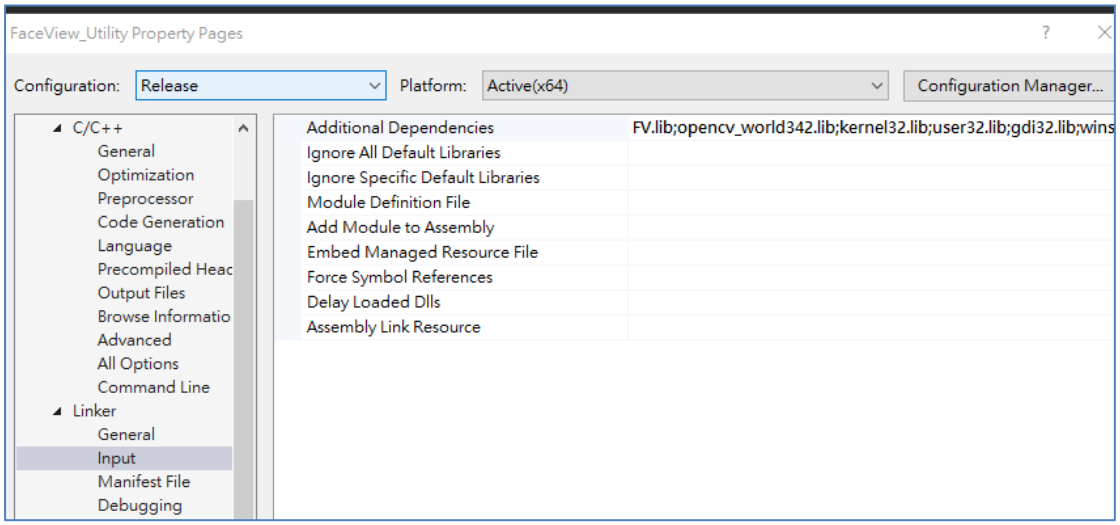
Reference

The header files and lib files of FaceView APIs can be found in the following path: folders of sdk_include and sdk_library.

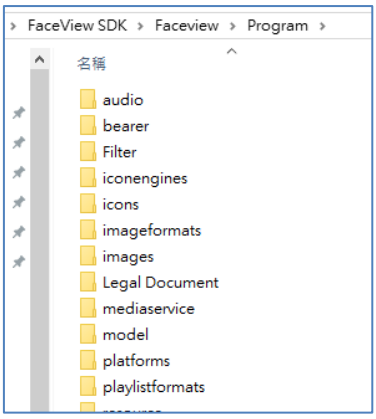
If your IDE is Microsoft Visual Studio, it's required to add two dependencies in your project. One is to include the path of sdk_include in Additional Include Directories.



The other is to include the path of sdk_library in Additional Dependencies.



Eventually, if you have an execution file developed by yourself through FaceView APIs, it's noted that put your execution file in the following **Program** folder to be executable.


FaceView SDK › Faceview › Program ›
名稱
audio
bearer
Filter
iconengines
icons
imageformats
images
Legal Document
mediaservice
model
platforms
playlistformats