# MORFIN AUTH WINDOWS CSHARP SDK

Version: 1.0.0.0

**Supported Devices**

MFS500, MELO31, MARC10

MANTRA™

## Contents

MANTRA™

## CONTROL SHEET

| | |
|---|---|
| DOCUMENT TITLE | MORFIN AUTH WINDOWS CSHARP SDK Version : 1.0.0.0 |
| RELEASE DATE | 26/06/2023 |
| PROJECT TEAM | Rajesh Koriya, Devyang Sathavara , Henry Roy |
| PREPARED BY | Rajesh Koriya |
| REVIEWED BY | Mahesh Patel |
| APPROVED BY | Mahesh Patel |
| SECURITY CLASSIFICATION | Restricted |

## DOCUMENT HISTORY

| **Document Version** | **Release Date** | **Release Notes** | **Author** |
|---|---|---|---|
| 1.0.0.0 | 26/06/2023 | Initial release document | Rajesh Koriya |

## 1. Overview

The document provides the functional and implementation information to work with MELO31, MARC10, MFS500 and you can also match finger (1:1) using ISO or ANSI template.

## 2. Prerequisite

SDK required below dependency.

- Window 10 and above
- .Net Framework 4.8 and above.
- VC redist 2013 & 2015
- Morfin Driver Setup

# 3. Functions

## 3.1. Initialize Device

Used for Initialized devices with manufacture data.

**CSharp:**

```
public int Init(string strProductName, int intProductNameLen, refFINGER_DEVICE_INFO
StDeviceInf);
```

**Structure:**

```
public struct MORFin_DEVICE_INFO
  {
[System.Runtime.InteropServices.MarshalAsAttribute(System.Runtime.InteropServices.Un
managedType.ByValTStr, SizeConst = 13)]
     public string SerialNo;

[System.Runtime.InteropServices.MarshalAsAttribute(System.Runtime.InteropServices.Un
managedType.ByValTStr, SizeConst = 13)]
     public string Firmware;

[System.Runtime.InteropServices.MarshalAsAttribute(System.Runtime.InteropServices.Un
managedType.ByValTStr, SizeConst = 7)]
     public string Make;

[System.Runtime.InteropServices.MarshalAsAttribute(System.Runtime.InteropServices.Un
managedType.ByValTStr, SizeConst = 12)]
     public string Model;

     public int Width;

     public int Height;

     public int DPI;

[System.Runtime.InteropServices.MarshalAsAttribute(System.Runtime.InteropServices.Un
managedType.ByValTStr, SizeConst =20)]
public string PCBSerialNo;

  }
```

**Parameters:**

string strProductName:
    [in] User passed product name.
int intProductNameLen:
    [in] User passed product name length.
ref FINGER_DEVICE_INFO StDeviceInf:
    [ref] Device initiation success on return device information.

**Return Values:**

0 = Success

0! = Failed

## 3.2. Device Connected Status

Used for checking device connected or not.

**CSharp:**

public int IsConnected(string strProductName);

**Parameters:**

string strProductName:

[in] User passed product name.

**Return Value:**

0 = Connected

0! = Failed

## 3.3. Get SDK Version

**Return SDK version.**

**CSharp:**

public int GetSDKVersion(out string Ver);

**Parameters:**

string Ver:

[out] Return SDK version.

**Return Value:**

0 = Success

0! = Failed

## 3.4. Start Capture

Used for asynchronously capture. Device initialization required.

**CSharp:**

public int StartCapture(int Timeout = 10000, int MinimumQuality = 40)

**Preview Call back:**

void OnPreview(CaptureData ObjCaptureData);

**Capture Complete Call back:**

void OnCaptureCompleted(CaptureData ObjCaptureData);

**Finger Position Call back :**

void OnFingerPosition(CaptureData ObjCaptureData);

**Parameters:**

int MinimumQuality:

[in] Quality range 1 to 100.

int Timeout:

[in] Timeout value set in milliseconds. If timeout set 0 then capture will be stopsafter finger detected with desired quality.

**CaptureData ObjCaptureData:**

Bitmap AutoCaptureBitmap.

[out] Preview bitmap Image.

int ErrorCode.

[out] Error Code while capturing.

Int Quality;

[out] Image Quality while capturing.

int Nfiq

[out] NFIQ Score after capturing success.

**Return Value:**

0 = Capture started

0! = Capture start failed

## 3.5. Start Auto Capture

Used for synchronously capture. Device initialization required.

**CSharp:**

public int AutoCapture(out int Qlt, out int Nfiq, int TimeOut = 10000, int MinimumQuality = 40);

**Preview Call back:**

void OnPreview(CaptureData ObjCaptureData);

**Finger Position Call back :**

void OnFingerPosition(CaptureData ObjCaptureData);

**Parameters:**

int MinimumQuality:

[in] Quality range 1 to 100.

int Qlt:

[out] Get quality on success.

int Nfiq:

[out] Get Nfiq on success.

CaptureData ObjCaptureData:

Bitmap AutoCaptureBitmap.

[out] preview bitmap Image.

int ErrorCode.

[out] Error Code while capturing.

Int Quality;

[out] Image Quality while capturing.

**Return Value:**

0  = Capture started

0! = Capture start failed

## 3.6. Stop Capture

Used for forcefully capture stop while capturing.

**CSharp:**

public int StopCapture();

**Return Value:**

0 = Success

0! = failed

## 3.7. Get Capture Finger Image

Used for getting finger image like “BMP”, “PNG”, “JPEG2000” , “WSQ” , ”RAW”, ” FIR_V2005”, ” FIR_V2011”, ” FIR_WSQ_V2005”, ” FIR_WSQ_V2011”, ” FIR_JPEG2000_V2005” and ” FIR_JPEG2000_V2011”. Capture success required.

**CSharp:**

public int GetImage(out byte[] bytesTemplate, IMAGE_FORMAT Format, int CompressionRatio);

**Enum:**

```
public enum IMAGE_FORMAT {
            BMP = 0,
            JPEG2000 = 2,  WSQ = 3,
            RAW = 4,
            FIR_V2005 = 5,
            FIR_V2011 = 6,
            FIR_WSQ_V2005 = 7,
            FIR_WSQ_V2011 = 8,
            FIR_JPEG2000_V2005 = 9,
       FIR_JPEG2000_V2011 = 10,
                                    } ;
```

**Parameters:**

byte[] bytesTemplate:
    [out] Last captured finger print image in bytes.
IMAGE_FORMAT format:
    [in] Passed image format.
Int CompressionRatio :
    [in]Pass (1-15) range for JPEG2000 and (1-10) for WSQ image.

**Return Value:**

0 = Success
0! = failed

## 3.8. Get Capture Finger Template

Used for getting finger print template as per mention enum . Capture success required.

**CSharp:**

```
public int GetTemplate(out byte[] bytesTemplate, TEMPLATE_FORMAT Format, int CompressionRatio);
```

**Enum:**

```
public enum TEMPLATE_FORMAT {
            FMR_V2005 = 0,
            FMR_V2011 = 1,
            ANSI_V378 = 2,
        };
```

**Parameters:**

byte[] bytesTemplate:
    [out] Last captured finger template in bytes.
TEMPLATE_FORMAT format:
    [in] Passed template format
Int CompressionRatio :
    [in]Pass (1-15) range for JPEG2000 and (1-10) for WSQ template.

**Return Value:**

0 = Success
0 ! = failed

## 3.9. Match Finger Template (1:1)

Used for matching fingerprint (1:1).

**Csharp:**
public int MatchTemplate(byte[] probTemplate, int probTemplateLen, byte[] galleryTemplate, int galleryTemplateLen, out int MatchScore, TEMPLATE_FORMATFormat);

**Parameters:**

byte[] probTemplate:
    [in] Passed probe template
int probTemplateLen:
    [in] Passed probe template length
byte[] galleryTemplate:
    [in] Passed gallery template
int galleryTemplateLen:
    [in] Passed gallery template length
int MatchScore:
    [out] Return matching score. (0- 1000 range)
TEMPLATE_FORMAT format:
    [in] Passed template format

**Note: This Functionality working with FMR_V2005, FMR_V2011 and ANSI_V378 Template only.**

**Return Value:**

0 = Success
0! = failed

## 3.10. Uninitialized Device

Used for uninitialized device and free all allocated memory.

**CSharp:**

public int Uninit();

**Return Value:**

0 = Success
0! = failed

## 3.11. Get Error Description

Used for getting description respect to error code.

**CSharp:**

public string GetErrDescription(int err);

**Parameters:**

int err:
    [in] Pass Error Code Generated by SDK.

## 3.12. Get Device List

Used for getting device list on success.

**CSharp:**

public int GetConnectedDevices(DEVICE_LIST[] DeviceList, out int DeviceCnt);

**Structure:**

public struct DEVICE_LIST{

[System.Runtime.InteropServices.MarshalAsAttribute(System.Runtime.InteropServices.UnmanagedType.ByValTStr, SizeConst = 12)]
public string Model;
}

**Parameters:**

DEVICE_LIST[] DeviceList:
[out] Structure with Connected Device List.int
DeviceCnt :
[out] Connected Device Count.

**Return Value:**

0 = Success
0 ! = failed

## 3.13. Get Supported Device List

Used for getting supported device list in SDK on success.

**CSharp:**

public int GetSupportedDevices (DEVICE_LIST[] DeviceList, out int DeviceCnt);

**Structure:**

public struct DEVICE_LIST
{

[System.Runtime.InteropServices.MarshalAsAttribute(System.Runtime.InteropServices.UnmanagedType.ByValTStr, SizeConst = 12)]
public string Model;
}

**Parameters:**

DEVICE_LIST[] DeviceList:
[out] Structure with Connected Device List.int
DeviceCnt :
[out] Connected Device Count.

**Return Value:**

0 = Success
0! = failed

## 3.14. Enable Logs

Used for tracing application log

**CSharp:**

```
public int EnableLogs (MorFin_AUTH_LOG_LEVEL eLogLevel, string pFolderPath);
```

**Enum:**

```
public enum MorFin_AUTH_LOG_LEVEL
{
   MorFin_AUTH_LOG_LEVEL_OFF = 0,
   MorFin_AUTH_LOG_LEVEL_ERROR = 1
}
```

**Parameters:**

MorFin_AUTH_LOG_LEVEL eLogLevel:

[in] pass enum log level.

string pFilePath:

[in] Logs will be saved on folder path, if folder path is blank then logs will be savedon application path (default).

**Return Value:**

0 = Success

0! = failed

## 3.15. Auto Device Detect Event

Used for device detection call back registration.

**CSharp:**

```
void OnDeviceDetection(string DeviceName DEVICE_DETECTION_EVENT dvcStatus)
```

**Enum:**

```
public enum MorFin_AUTH_LOG_LEVEL
{
   EVENT_DISCONNECTED = 0,
   EVENT_CONNECTED = 1
}
```

**Parameters:**

string DeviceName:

[out] Get device name while device detection.

DEVICE_DETECTION_EVENT  dvcStatus:

[out]: Status of the device - Connected/Disconnected from enum.