# HƯỚNG DẪN CHUNG

## SDK

- Các thư viện trong folder: “SDK/hn212\_sdk\_v1.x.zip”

## Sample code sử dụng SDK

- Windows 10, Visual studio 2022, .netcore 6

- Tải mã nguồn tại folder: “Code mẫu/demo-src.zip”, giải nén.

- Copy các thư viện trong sdk vào ./lib thư mục vừa giải nén

- Mở project bằng vs2022, các thư viện bổ sung được update qua nuget tự động.

## Ứng dụng demo

- Ứng dụng demo build sẵn trong folder: “Demo/demo-bin.zip”, tải về giải nén và chạy file: IdCard.Hanel.exe

## **Một số lỗi và cách xử lý**

- 1 số dòng chip Intel Pentium báo lỗi load tennis.dll. Khắc phục: Vào folder chứa tennis.dll, đổi tennis.dll => tennis\_bkp.dll, đổi tennis\_pentium.dll => tennis.dll.

- Đối với Windows 10N hoặc KN (Bản không có Media), cần Add feature bổ sung: 1. Chạy cmd.exe quyền admin; 2. Chạy lệnh: DISM /Online /Get-Capabilities để list các package có thể cài, tìm MediaFeaturePack; 3. Nếu chưa cài đặt thì cài theo lệnh: DISM /Online /Add-Capability /CapabilityName:Media.MediaFeaturePack~~~~0.0.1.0

# HƯỚNG DẪN SỬ DỤNG SDK

- Để dễ dàng phát triển, vui lòng sử dụng mã nguồn tại “Code mẫu/demo-src.zip”, và chỉnh sửa trực tiếp trên project có sẵn.

- Hầu hết các tác vụ của SDK là Async, kết quả xử lý được thông báo qua event OnStatusChanged và OnVideoFrame.

## Khai báo object và khởi tạo

Bắt đầu sử dụng:

VnHn212Reader \_reader = new VnHn212Reader; //Tạo object

\_reader.OnStatusChanged += Reader\_OnStatusChanged; //Handle nhận event

var config = new VnHn212Config();

\_reader.StartMonitor(config);

config: Sử dụng để tùy chỉnh tham số, null để sử dụng mặc định

Kết thúc sử dụng:

\_reader.StopMonitor();

## Xử lý các sự kiện

private void OnEvent(object sender, StatusEventArgs e)

{

switch (e.EventName)

{

case EVENT\_NAMES.READER:

ProcessReaderEvent(e as StatusEventReaderArgs);

break;

case EVENT\_NAMES.CARD:

ProcessCardEvent(e as StatusEventCardArgs);

break;

case EVENT\_NAMES.READ:

ProcessReadEvent(e as StatusEventReadArgs);

break;

case EVENT\_NAMES.READER\_CAMERA:

ProcessCameraEvent(e as StatusEventCameraArgs);

break;

…

}

}

SDK cung cấp 5 trạng thái event:

- READER: Các sự kiện thông báo trạng thái đầu đọc (Cắm vào, rút ra, đầu đọc lỗi)

- CARD: Các sự kiện liên quan thẻ: Có thẻ, không có thẻ

- READ: Các sự kiện liên quan quá trình đọc thẻ: START/ PACE/ READDG…

- READER\_CAMERA: Các sự kiện liên quan đến trạng thái camera chụp ảnh khuôn mặt (Cắm vào, rút ra)

- FACECAPTURE: Các sự kiện liên quan quá trình chụp ảnh khuôn mặt

### Sự kiện đầu đọc

private void ProcessReaderEvent(StatusEventReaderArgs? ev)

Trạng thái đầu đọc trong ev.NewState:

ev.NewState == READER\_STATUS.ADDED: Đầu đọc được cắm vào máy tính

ev.NewState == READER\_STATUS.REMOVED: Đầu đọc mất kết nối

### Sự kiện thẻ

private void ProcessCardEvent(StatusEventCardArgs? ev)

Trạng thái thẻ trong ev.NewState:

ev.NewState == CARD\_STATUS.PRESENT: Thẻ được cắm vào đầu đọc

ev.NewState == CARD\_STATUS.EMPTY: Thẻ bị rút ra khỏi đầu đọc

### Sự kiện xử lý đọc thẻ

private void ProcessReadEvent(StatusEventReadArgs? ev)

Các bước đọc thẻ trong ev.Step:

ev.Step == READ\_CARD\_STEPS.SCANCARD: Bắt đầu OCR mặt thẻ

ev.Step == READ\_CARD\_STEPS.START: Bắt đầu đọc thẻ CHIP

ev.Step == READ\_CARD\_STEPS.CONNECT\_CARD: Kết nối thẻ CHIP

ev.Step == READ\_CARD\_STEPS.PACE: Xác thực PACE

ev.Step == READ\_CARD\_STEPS.READ\_DGS: Đọc dữ liệu các phân vùng

ev.Step == READ\_CARD\_STEPS.VERIFY\_SOD: Kiểm tra tính toàn vẹn

ev.Step == READ\_CARD\_STEPS.AACA\_AUTHEN: Xác thực CHIP

ev.Step == READ\_CARD\_STEPS.FINISH: Kết thúc quá trình đọc thẻ

Trạng thái các bước được kiểm tra thông qua ev.Status:

ev.Status == READ\_CARD\_STATUS.START: Bắt đầu thực hiện bước hiện tại

ev.Status == READ\_CARD\_STATUS.SUCCESS: Bước thực hiện thành công

ev.Status == READ\_CARD\_STATUS.FAILURE: Bước thực hiện thành công

ev.Status == READ\_CARD\_STATUS.IGNORE: Bước hiện tại bị bỏ qua

Dữ liệu thẻ trong ev.Step == READ\_CARD\_STEPS.FINISH và trạng thái thành công (ev.Status == READ\_CARD\_STATUS.SUCCESS) được lưu trong \_reader.CardData bao gồm:

\_reader.CardData.Dg2File.FaceImage: Ảnh khuôn mặt lưu trong CHIP

\_reader.CardData.Dg13File: Thông tin các nhân

\_reader.CardData.Dg1File.Mrz: Thông tin MRZ

\_reader.CardData.VerifySOD: Thông tin kiểm tra tính toàn vẹn

\_reader.CardData.AaCaAuthen: Thông tin xác thực CHIP

### Sự kiện trạng thái Camera chụp ảnh

private void ProcessCameraEvent(StatusEventCameraArgs? ev)

Trạng thái camera chụp ảnh trong ev.NewState:

ev.NewState == CAMERA\_STATUS.PRESENT: Camera được kết nối

ev.NewState == CAMERA\_STATUS.EMPTY: Camera mất kết nối

### Sự kiện chụp ảnh khuôn mặt

Để đăng ký nhận các Frame ảnh chụp camera:

\_reader.OnVideoFrame += reader\_OnVideoFrame;

private void OnEvent(object sender, StatusEventArgs e)

{

CaptureEventArgs? vd = e as CaptureEventArgs;

if (vd == null)

return;

this.FaceType = vd.FaceType; // Ảnh khuôn mặt (Nếu có)

this.FaceImage = vd.FaceData; // Frame ảnh từ Camera

switch (vd.Status)

{

case FACE\_CAPTURE\_STATAUS.CAPTURING:

// Đang capture

break;

case FACE\_CAPTURE\_STATAUS.FAILURE:

// Nhận diện khuôn mặt thất bại

break;

case FACE\_CAPTURE\_STATAUS.SUCCESS:

// Nhận diện khuôn mặt thành công

break;

case FACE\_CAPTURE\_STATAUS.IGNORE:

// Bỏ qua

break;

…

}

}

## So sánh khuôn mặt

int score = \_reader.CompareFace(\_ChipFaceBts, \_CamFaceBts);

Trong đó:

\_ChipFaceBts: Dữ liệu ảnh khuôn mặt lưu trong CHIP

\_CamFaceBts: Dữ liệu ảnh khuôn mặt chụp từ Camera

score: Số điểm so sánh, từ 00%-100%