1. Import AUTELSDK

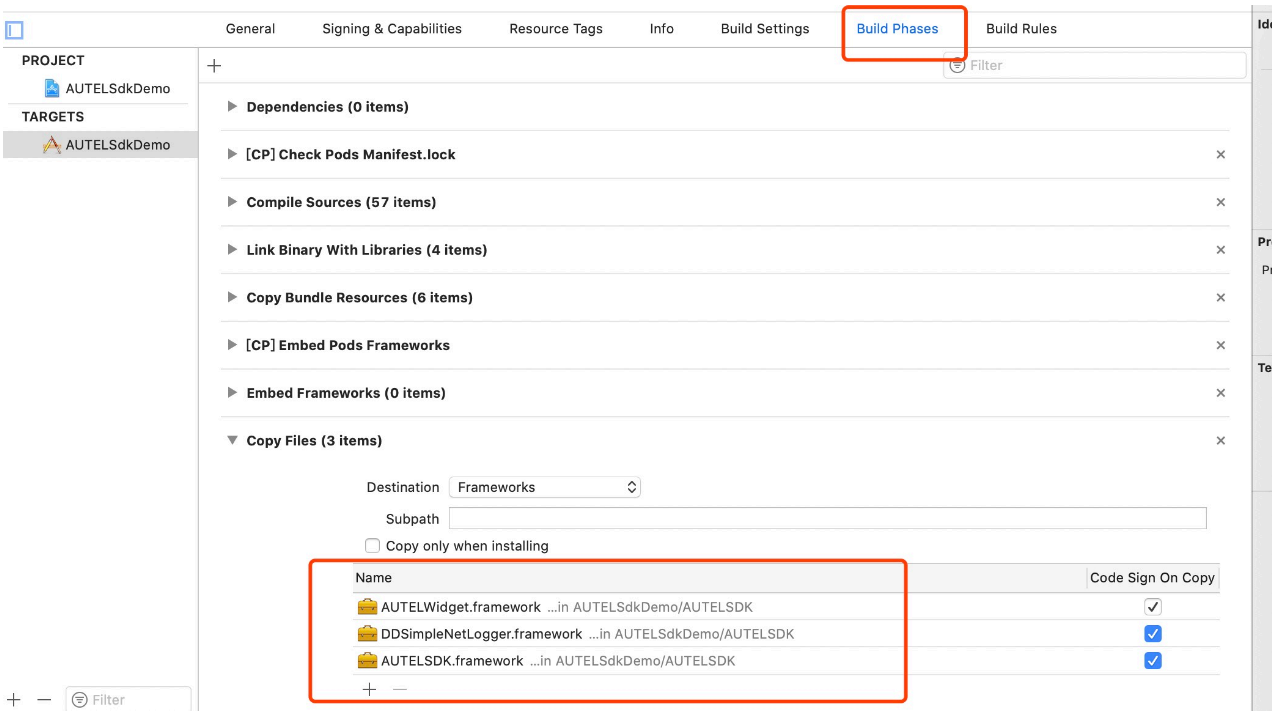
AUTELSDK supports iOS 11 and above.

AUTELSdkDemo was complied with XCode 12.4, support swift 5.3。

1. AUTELSDK includes 3 xcframework: AUTELSDK.xcframework, AUTELWidget.xcframework, DDSimpleNetLogger.xcframework. 2 strings: shader_fsh_myExpose.strings, shader_vsh_myExpose.strings.

Refer to AUTELSdkDemo project, and drag AUTELSDK folder to your project directly.

In AUTELSdkDemo project, build setting：target->Build Phases ->+ sign in the upper left corner -〉New Copy Files Phase, Under Copy Files setting, select Destination: Frameworks, and add DLL


General
Signing & Capabilities
Resource Tags
Info
Build Settings
Build Phases
Build Rules
PROJECT
AUTELSdkDemo
TARGETS
AUTELSdkDemo
Filter
Dependencies (0 items)
[CP] Check Pods Manifest.lock
Compile Sources (57 items)
Link Binary With Libraries (4 items)
Copy Bundle Resources (6 items)
[CP] Embed Pods Frameworks
Embed Frameworks (0 items)
Copy Files (3 items)

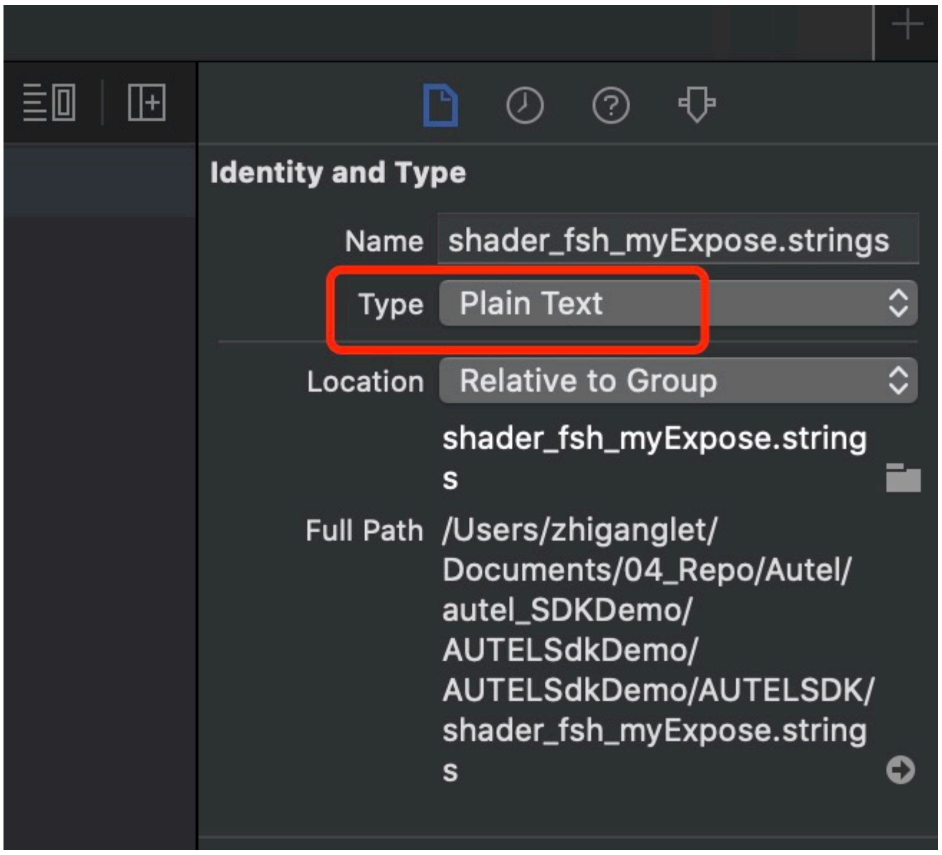
Destination: Frameworks

Subpath:

Copy only when installing ☐

Name	Code Sign On Copy
AUTELWidget.framework ...in AUTELSdkDemo/AUTELSDK	✓
DDSimpleNetLogger.framework ...in AUTELSdkDemo/AUTELSDK	✓
AUTELSDK.framework ...in AUTELSdkDemo/AUTELSDK	✓

2. For shader_fsh_myExpose.strings and shader_vsh_myExpose.strings, under the Identity and Type option, select "Plain Text" for type option as shown below.


Identity and Type

Name shader_fsh_myExpose.strings

Type Plain Text

Location Relative to Group

shader_fsh_myExpose.strings

Full Path /Users/zhiganglet/Documents/04_Repo/Autel/autel_SDKDemo/AUTELSdkDemo/AUTELSdkDemo/AUTELSDK/shader_fsh_myExpose.strings

2. Configure Info.plist

The following configuration needs to be added into Info.plist:

```
<key>CFBundleDisplayName</key>
<string>AUTELSdkDemo</string>
<key>UISupportedExternalAccessoryProtocols</key>
<array>
  <string>com.autelrobotics.control</string>
  <string>com.auteltech.control</string>
</array>
<key>NSAppTransportSecurity</key>
<dict>
  <key>NSAllowsArbitraryLoads</key>
  <true/>
</dict>
<key>NSLocationAlwaysAndWhenInUseUsageDescription</key>
<string>App needs your consent to access location</string>
<key>NSLocationAlwaysUsageDescription</key>
<string>App needs your consent to access location</string>
<key>NSLocationWhenInUseUsageDescription</key>
<string>App needs your consent to access location</string>
```

1、CFBundleDisplayName

Indicates the APP name, which must be configured and must be consistent with the APP name you used for APP key application.

2、UISupportedExternalAccessoryProtocols

The external accesory frame, only if it is configured, the APP can connect to Autel Remote Control through USB cable.

3、NSAppTransportSecurity

Apply for http permissions

4、NSLocationAlwaysAndWhenInUseUsageDescription

Appy for positioning permissions

三、Initialize SDK

Refer to **AUTELSdkDemo**，the DataLayer module in AUTELSdkDemo is used for initialize SDK and establish the connection of each module of the aircraft, and monitor the data reported by each module of the SDK, and it can be used in your project directly.

1. Call the SDKConnection.shard.connect() in didFinishLaunchingWithOptionsSDKConnection.shared.connect(), as follows

```
func application(_ application: UIApplication, didFinishLaunchingWithOptions
launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {

      SDKConnection.shared.connect()

      return true
  }
```

2. APP Registration is performed in the SDKConnection.shared.connect(), appKey and proxy are required for registration, and the SDK will authenticate the appKey, and the authentication result will be called back by the proxy. If the authentication is passed, the SDK will be initialized. For detail, please refer to the codings in demo.

```
func connect() -> Void {
      //appKey
      AUTELAppManager.registerApp("your appKey", with: self)
  }
```

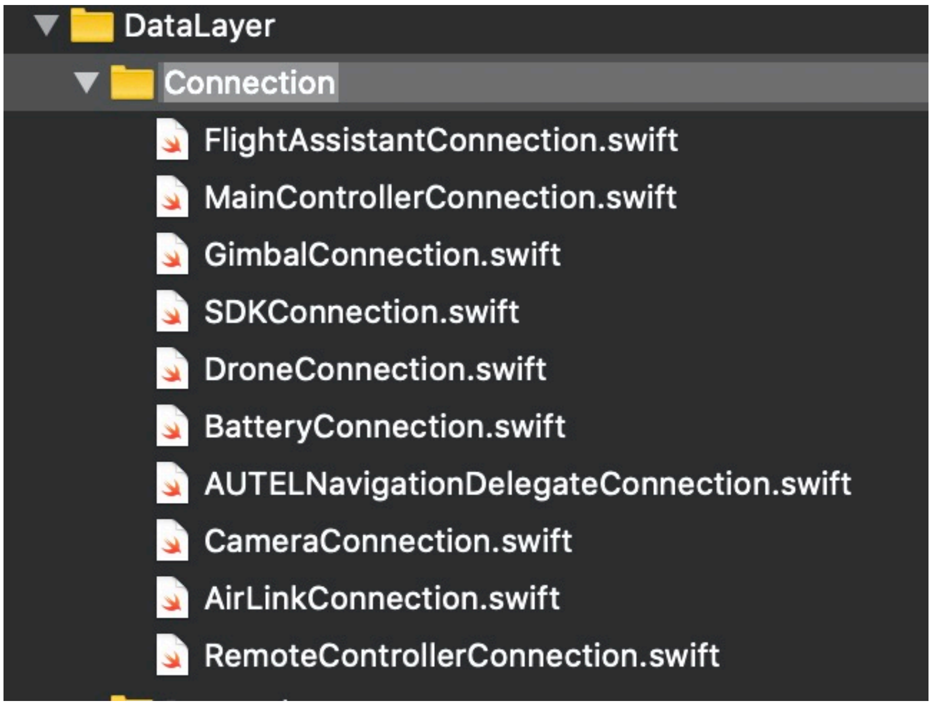
NOTE: appKey is the assoicated key applied by the devlopers on the Autel Developer Platform(http://pro.autelrobotics.com/)

3. If the registration is sucessful, connect mobile phone and RC with a USB data cabe. Open APP, establish the connection to trigger a callback of didConnectionStatusChanged, and establish the proxy relationship between the Connection layer and all the modules of SDK.

```
// MARK: - AUTELDeviceDelegate

    func device(_ device: AUTELDevice!, didConnectionStatusChanged status:
AUTELDeviceConnectionStatus) {
        guard let drone = device as? AUTELDrone else {
            self.connectedDrone = nil
            return
        }
        CameraConnection.shared.connect(drone.camera)
        GimbalConnection.shared.connect(drone.gimbal)
        BatteryConnection.shared.connect(drone.battery)
        MainControllerConnection.shared.connect(drone.mainController)
        RemoteControllerConnection.shared.connect(drone.remoteController)
        AirLinkConnection.shared.connect(drone.airLink)

AUTELNavigationDelegateConnection.shared.connect(drone.mainController.navig
ationManager)

FlightAssistantConnection.shared.connect(drone.mainController.flightAssista
nt)

        //callback
        queue.async {
            for (_, aProtocol) in self.protocols {
                aProtocol.device(device, didConnectionStatusChanged:
status)
            }
        }

    }
```

Connection layer implements the proxy method of each module at the fundation of SDK, and monitor the status updates of each module:

- FlightAssistantConnection： Vision connection layer
- MainControllerConnection： Flight control connection layer
- GimbalConnection： Gimbal connection layer
- BatteryConnection： Battery connection layer
- AUTELNavigationDelegateConnection： Mission connection layer
- CameraConnection： Camera connection layer
- AirLinkConnection： Image transmission connection layer
- RemoteControllerConnection： Remote Control connection layer


DataLayer
Connection
FlightAssistantConnection.swift
MainControllerConnection.swift
GimbalConnection.swift
SDKConnection.swift
DroneConnection.swift
BatteryConnection.swift
AUTELNavigationDelegateConnection.swift
CameraConnection.swift
AirLinkConnection.swift
RemoteControllerConnection.swift

If the application layer wants to monitor the data of a certain module, you can register to monitor the corresponding Connection layer and implement the coressponding prxoty method. For detail, refer to AircraftStatus Demo in AUTELSdkDemo:

```
fileprivate func registerProtocol() {
    DroneConnection.shared.addProtocol(self)
    MainControllerConnection.shared.addProtocol(self)
    RemoteControllerConnection.shared.addProtocol(self)
    BatteryConnection.shared.addProtocol(self)
    CameraConnection.shared.addProtocol(self)
    GimbalConnection.shared.addProtocol(self)
    FlightAssistantConnection.shared.addProtocol(self)

}
```

4. Using SDK

Bridge to the module mangement class corresponding to SDK through each Connection layer, and then call the interface of each module:

AUTELDrone

This type can be used to operate EVO 1 and EVO 2 aircraft

Access Handle of application layer: DroneConnection.shared.drone

AUTELDroneMainController

Through AUTELDroneMainController class, you can get the current status of aircraft main control, read and write the main control parameters, and also send commands to the main control.

Access Handle of application layer: MainControllerConnection.shared.mainController

AUTELBattery

It manages battery related information and can obtain the real-time status of the aircraft battery.

MainControllerConnection.shared.mainController

AUTELRemoteController

You can read and write remote control parameters, calibrate the remote control, and you can also get the status of the remote control, the real-time status of joystick, button and scroll wheel.

Access Handle of application layer: RemoteControllerConnection.shared.remoteController

AUTELDroneGimbal

You can get the real-time status of the gimbal, and send commands to the gimbal to control the gimbal

Access Handle of application layer: GimbalConnection.shared.gimbal

AUTELBaseCamera

You can get the current state of the camera, read and write camera parameters, and perform related operations on the camera as well.

Access Handle of application layer: CameraConnection.shared.camera

AUTELAirLink

You can get the connection status of the communication link

Access Handle of application layer:AirLinkConnection.shared.airlink

AUTELFlightAssistant

You can get the real-time status of the vision, send commands to the vision to control the vision as well.

Access Handle of application layer:
MainControllerConnection.shared.mainController?.flightLimitation

AUTELNavigation

With navigationManager, the aircraft perform automatic flight missions

Access Handle of application layer:
MainControllerConnection.shared.mainCtl?.navigationManager

5. Camera Module

Currently available camera types are XT701, XT705, XT706. After the APP is successfully connected to the aircraft, it will also establish a connection with the camera at the same time. After the connection, the camera status and changes will be reported in real time through the CameraConnection proxy method. You can monitor each status of the camera as needed.

```
protocol CameraProtocol: CheckProtocol {

    func camera(_ camera: AUTELBaseCamera!, didUpdateConnectState connectState: AUTELCameraConnectState) -> ()

    func camera(_ camera: AUTELBaseCamera!, didUpdateRecordRecoverState recoverState: AUTELCameraRecordRecoverState) -> ()

    func camera(_ camera: AUTELBaseCamera!, didGenerateNewMediaFile newMedia: AUTELMedia!) -> ()

    func camera(_ camera: AUTELBaseCamera!, didUpdateSDCardState sdCardState: AUTELCameraSDCardState!) -> ()

    func camera(_ camera: AUTELBaseCamera!, didUpdateMMCState mmcState: AUTELCameraSDCardState!) -> ()

    func camera(_ camera: AUTELBaseCamera!, didUpdateSystemState systemState: AUTELCameraSystemBaseState!) -> ()

    func camera(_ camera: AUTELBaseCamera!, didUpdateParameters parameters: AUTELCameraParameters!, change: [AnyHashable : Array<Any>]!) -> ()

    func camera(_ camera: AUTELBaseCamera!, didUpdateCurrentExposureValues exposureParameters: AUTELCameraExposureParameters!) -> ()

    func camera(_ camera: AUTELBaseCamera!, didUpdateHistogramTotalPixels totalPixels: Int, andPixelsPerLevel pixelsArray: [Any]!) -> ()

    func camera(_ camera: AUTELBaseCamera!, didUpdateTimeLapseCaptured captureCount: Int, andCountDown timeInterval: TimeInterval) -> ()

    func camera(_ camera: AUTELBaseCamera!, didUpdateAutoFocusState focusState: AUTELCameraAFLensFocusState, andLensFocusAreaRowIndex rowIndex: Int, andColIndex colIndex: Int) -> ()

    func camera(_ camera: AUTELBaseCamera!, didUpdateAeAfState state: Int) -> ()
}
```

AUTELCameraParameters: You can get the parameter currently supported by the camera. The example is as follows. Get the resolution range of the photo supported by the current camera. For details, refer to the demo:

```
class func requestPhotoSizeWithItem() -> [String] {
        guard let numbers =
CameraConnection.shared.camera?.parameters.supportedCameraPhotoAspectRatioRange
() else { return [] }
        var items: [String] = []
        for num in numbers.reversed() {
            let parameter = AUTELCameraPhotoAspectRatio(rawValue:
UInt8(truncating: num))?.displayName() ?? ""
            items.append(parameter)
        }
        return items
    }
```

The application layer calls the SDK camera command. For detail, please refer to the CameraCommandDeliver.swift file in the demo, such as setting the size of the picture

```

class func setPhotoSize(_ size: AUTELCameraPhotoAspectRatio, with result:
ResultHandler) {
    CameraConnection.shared.camera?.setPhotoRatio(size, withCompletion: {
result?($0) })
}
```

The query parameters currently supported are: AUTELCameraWorkMode, AUTELCameraVideoResolution, AUTELCameraVideoFileFormat、AUTELCameraPhotoAspectRatio, AUTELCameraExposureMode.

6. AUTELSdkDemo

For more details, please refer to AUTELSdkDemo, and the following features are available:

- Image transmission display
- Map display, including the aircraft home points and real-time aircraft point updates
- Flight status display
- Read and write in camera mode
- Read and write of camera resolution
- Waypoint flight demo for mission flight