Aeroscope SDK Linux

User Guide V1.0

2018.05

Contents

Introduction

Purpose

AEROSCOPE^{TM} SDK is a software development kit that provides application programming interfaces (APIs) to help users build a server with an Aeroscope unit.

This manual provides a simple guide for SDK users, allowing users to quickly understand the SDK and its main processes and APIs.

Intended Audience

This document is intended for users who are building their own server with an Aeroscope unit.

Legends

i Important
! Warning
X Error

References

File Name	Description
UavMonitorSdkLinux_Reference.chm	*Aeroscope SDK Reference Manual* (English Version): Provides a detailed reference for all APIs. Included in the SDK package's Help directory.

Abbreviations and Terms

Term	Description
Aeroscope	A UAV detection system to identify, track, and monitor airborne drones.
API	Application Programming Interface
SDK	Software Development Kit
SSL	Secure Sockets Layer, a network security protocol
UAV	Unmanned Aerial Vehicle

Aeroscope SDK Package

The Aeroscope SDK package is provided as a ZIP file.
The directory structure after extraction is as follows:

```
UavMonitorSdkLinux_V1.n.nn.nnn
 ├── Help
 │   └── html
 ├── lib
 ├── UavMonitorSdk
 └── UavMonitorSdkExample
```

The detailed directory and file contents are as follows:

```
UavMonitorSdkLinux_V1.n.nn.nnn
 ├── Help
 │   │  UavMonitorSdkLinux_V1.n.nn.nnn_Reference.chm
 │   │  UavMonitorSdkLinux_V1.n.nn.nnn 参考手册 .chm
 │   │
 │   └── html
 │         annotated.htm
 │         … (More files)
 │         index.htm
 │         …(More files)
 ├── lib
 │     libcrypto.so
 │     libssl.so
 │     libUavMonitorSdk.so
 │     openssl.lic
 │
 ├── UavMonitorSdk
 │     build_env_ver.txt
 │     UavMonitorSdkApi.h
 │     UavMonitorSdkMsg.h
 │     ZytPal.h
 │
 └── UavMonitorSdkExample
       main.cpp
       UavMonitorSdkExample.pro
       UavMonitorSdkExample.pro.user
```

Descriptions for the main folders in the directory are as follows:

Folder	Description
Help	This folder contains Help files, including the reference manual.
lib	This folder contains the SDK's shared library files (*.so), which need to be copied to the corresponding "lib" directory of the Linux system.
UavMonitorSdk	This folder contains the SDK's API-defined header files.
UavMonitorSdkExample	This folder contains an SDK application example.

Installation

Environmental Requirements

Refer to the "build_env_ver.txt" file in the "lib" folder of the SDK package for details about the Linux SDK’s production, installation, and operation environments.
The following shows an example of a "build_env_ver.txt" file:

Linux ubuntu 3.13.0-32-generic #57-Ubuntu SMP Tue Jul 15 03:51:08 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux
gcc (Ubuntu 4.8.4-2ubuntu1~14.04.3) 4.8.4
Qt Creator 3.0.1 based on Qt 5.2.1
OpenSSL 1.1.0f 25 May 2017

SDK Package Extraction and Runtime Library Deployment

Extract the SDK package and copy the runtime files to the system's run directory.

For example:
Copy the SDK's shared library files (*.so) to the "/usr/lib" directory.

Running Examples

The application example source code is in the "UavMonitorSdkExample" directory of the SDK package.
Open "UavMonitorSdkExample.pro" with QT Creator, compile, and run.

Main Processes and APIs

Main Processes

The following APIs are used for the processes described in the table:

API	Description
InitSdk	Initializes the SDK.
InitUavMonitorListenerCer	Configures the digital certificate and CA certificate for the receiving server of the SDK.
StartUavMonitorListener	Specifies the server TCP port and the application's data processing callback function, then starts the receiving server.
StopUavMonitorListener	Turns off the receiving server.
ExitSdk	Exits the SDK and cleans up the environment.

You can refer to the Aeroscope SDK Reference Manual for details of example source code and further descriptions of the APIs.

APIs for Sending and Receiving Data

To send data to a specified unit, you can use the SendToDevice API.

There are two ways to receive data:
• Callback function
• Polling

If a callback function is set when calling StartUavMonitorListener, a callback function is used.

Otherwise, if StartUavMonitorListener passes a null pointer to the callback function parameter, polling is used. To receive data from all units through polling, use the ReceiveFromDevice API.

APIs for Managing Aeroscope's Digital Certificates

Configure the Aeroscope's digital certificate to manage whether the unit is allowed to access the receiving server.
The following APIs can be used to manage the digital certificates of Aeroscope:

API	Description
AddUavMonitorCer	Add an Aeroscope certificate to allow this unit to access the receiving server.
DelUavMonitorCer	Delete an Aeroscope certificate, preventing this unit from accessing the receiving server.
SetUavMonitorCer	Modify an Aeroscope certificate, which means that the unit must use the new certificate the next time to access the receiving server.

Data Analysis

MsgId

```
enum EIoMessageId
{
            eimiIdNone,         ///< 0
            eimiIdUavInfoInd,   ///< 1, Device -> UAV Information Ind -> Server
            eimiIdDeviceStatusInd,          ///< 2, Device -> Device Status Ind -> Server
            eimiIdReadDeviceSetting,        ///< 3, Server -> ServerToDeviceReq -> Device ->
ServerToDeviceAck -> Server
            eimiIdWriteDeviceSetting,       ///< 4, Server -> ServerToDeviceReq -> Device ->
ServerToDeviceAck -> Server
            eimiIdDeviceStatusLogInd,       ///< 5, Device -> Device Status Log Ind -> Server
            eimiIdScanFreqResultInd,        ///< 6, Device -> ScanFreq Result Status Ind ->
Server
            eimiIdDeviceBistInd,            ///< 7, Device -> Device Bist Ind -> Server
            eimiIdDevMacSetting,            ///< 8, Server -> ServerToDeviceReq -> Device ->
ServerToDeviceAck -> Server
            eimiIdDevVersion,   ///< 9, Server -> ServerToDeviceReq -> Device ->
ServerToDeviceAck -> Server
   eimiId2DeviceHeartBeatInd,               ///< 10, Server -> ServerToDeviceReq -> Device
            eimiIdEnterUpgradeMode,         ///< 11, Server -> ServerToDeviceReq -> Device ->
ServerToDeviceAck -> Server
            eimiIdReceiveFirmwareDataTransferMode,          ///< 12, Server ->
ServerToDeviceReq -> Device -> ServerToDeviceAck -> Server
            eimiIdFirmwareDataTransfer,  ///< 13, Server -> ServerToDeviceReq -> Device ->
ServerToDeviceAck -> Server
            eimiIdFirmwareTransferDone, ///< 14, Server -> ServerToDeviceReq -> Device ->
ServerToDeviceAck -> Server
            eimiIdDevReboot,   ///< 15, Server -> ServerToDeviceReq -> Device ->
ServerToDeviceAck -> Server
            eimiIdFirmwareUpgradeControl,           ///< 16, Server -> ServerToDeviceReq ->
Device -> ServerToDeviceAck -> Server
            eimiIdFirmwareUpgradeStatusInd,         ///< 17, Device -> Device Firmware
Upgrade Status Ind -> Server
            eimiIdClientCertValidInd,       ///< 18, Device -> Device Cert Valid Ind -> Server
            eimiIdClientSnInvalidInd,       ///< 19, Device -> Device SN Invalid Ind -> Server
            eimiIdClientNotMatchInd,        ///< 20, Device -> Device SN Not Match Ind -> Server
   eimiIdGetVersionOrLog,      ///< 21, Server -> ServerToDeviceReq -> Device ->
ServerToDeviceAck -> Server
   eimiIdDeviceFirwareConsUpgradeInd,              ///< 22, Device -> Device Firmware
consistency upgrade Ind -> Server
            eimiIdNum           ///< maximum
};
```

MsgType

```
enum EIoMessageType
{
        eimtTypeNone,       ///< 0
        /// Request from Server to Device
        eimtTypeServerToDeviceReq, ///< 1, Server -> Req -> Device
        eimtTypeServerToDeviceAck, ///< 2, Server <- Ack <- Device
        eimtTypeServerToDeviceTimeout,      ///< 3, Server <- ServerToDeviceReq
Timeout <- SDK
        eimtTypeServerToDeviceError,        ///< 4, Server <- ServerToDeviceReq Error
<- SDK

        /// Request from Device to Server
        eimtTypeDeviceToServerReq, ///< 5, Device -> Req -> Server
        eimtTypeDeviceToServerAck, ///< 6, Device <- Ack <- Server
        eimtTypeDeviceToServerError,        ///< 7, SDK <- Error <- Server

        /// Indication from Device to Server
        eimtTypeDeviceToServerInd,  ///< 8, Device -> Ind -> Server
        // Indication from Server to Device
        eimtTypeServerToDeviceInd,  ///< 9, Server -> Ind -> Device

        /// Device Management from SDK to Server
        eimtTypeDeviceArrival,      ///< 10, SDK -> Device Arrival -> Server
        eimtTypeDeviceRemoval,      ///< 11, SDK -> Device Removal -> Server
        eimtTypeDeviceFailure,      ///< 12, SDK -> Device Failure -> Server
        eimtTypeDeviceRecovery,     ///< 13, SDK -> Device Recovery -> Server

        /// SDK Management from Server to Device
        eimtTypeSdkError,   ///< 14, SDK -> Error -> Server
        eimtTypeSdkIoControlReq,    ///< 15, Server -> IO Control Req -> SDK
        eimtTypeSdkIoControlAck,    ///< 16, Server <- IO Control Ack <- SDK
        eimtTypeNum         ///< maximum
};
```

Transmitting Packet Data Structures

Not yet available.