IOS SIP SDK

Mizutech SIP SDK source-code for iOS

Contents

About

This is the documentation for the usage of the Mizutech iOS SIP SDK, which is a modified PJSIP SIP stack already configured with all the codecs and project settings for iOS. This can save you from all the hassle of configuring, extending and compiling PJSIP.

License terms

All source code provided by Mizutech must be treated as confidential material.
We (Mizutech SRL) provide the source code of the iOS SDK for your organization (as an individual, company or any organization), under the following terms:

- Title, ownership rights, and intellectual property rights in the Software shall remain with MizuTech and/or its suppliers.

- The agreement and the license granted hereunder will terminate automatically if you fail to comply with the limitations described herein. Upon termination, you must destroy all copies of the Software
- You can use the source code to develop your own iOS application.
- We reserve all rights regarding the source code. The iOS developer virtual machine, project settings and source code constitutes valuable copyrighted intellectual property and you will not become the owner of the software.
- We can provide the source for your organization only as a confidential material to fulfill your own app development requirements.
- You are allowed to use the source code only for your organization internal needs. Any kind of distribution of the source code is strictly forbidden.
- No right is granted to sell, resell, rent, distribute, make available, publish or otherwise transfer the source code. The source code must never leak out of your organization.
- The SDK is based on PJSIP, which is under GPL or proprietary license.
  By using the source code or the virtual machine provided by Mizutech, you must also agree with the PJSIP license terms.
- You shall not employ source code in any way that competes either directly or indirectly with us including but not limited to creation of derivative works that compete either directly or indirectly with our MizuPhone software.
- You shall not sell or offer as a service the derivative works for other companies, organizations or individuals
- The source code and the virtual machine is provided "as is" without any warranty of any kind

By opening or coping the source code files, you agree with all the above terms.
Otherwise, you must immediately delete all copies of the virtual machine and all the source code from all your computers.

Virtual Machine

This chapter is useful only if you received a developer virtual machine image. Otherwise, please skip this chapter.

We can provide a turnkey development environment as a VMware image with completely configured and working project settings for the iOS SDK; just download the VMware image, run it and you are ready to develop your iOS application.

The Steps to run the Mac OS developer VM image and start development are listed below.

Prerequisites

- A PC running Windows
- VMware Player or VMware Workstation
- Developer Mac image form Mizutech

If you prefer other virtual machine such as Virtual Box, VMWare Fusion or Parallels Desktop, then you can convert the VMware image to your preferred format and ignore the VMWare Workstation/Player related instructions below.

Download the virtual machine image first from Mizutech. If compressed, unzip it (usually compressed with zip).

Install VMware

Follow the steps below to install VMware Player or VMware Workstation:

1. Install the VMware player or workstation
2. Unzip the Developer Mac image downloaded form Mizutech
3. Patch VMware to be able to run Mac OS:
    a. Make sure VMware is closed
    b. Navigate to “vmware-unlocker” which can be found in the downloaded image
    c. Right click on “win-install.cmd” -> Run as administrator
    d. Wait a few minutes until the operation is finished (the command line widow will close)

Create Virtual machine

Follow the steps below:

1. Launch VMware and click *File* menu -> *New virtual machine.*
2. On the first page of the wizard select *Typical* and click *Next.*
3. On the *Guest Operating System Installation* page click *I will install the operation system later* and click *Next.*
4. On the *Select a Guest Operating System* page select *Apple Mac OS X* for the operating system and select *macOS 10.14* form the dropdown list. Click *Next*.
5. On the *Name the Virtual Machine* page give a name to your machine and select the location for this machine. Click *Next*.
6. On the *Specify disk capacity* page just click *Next*. We will replace this disk anyway in the following steps.
7. Click *Finish*.

Edit virtual machine settings

Follow the steps below:

1. Go to *VM* menu -> *Settings*
2. On *Hardware* tab:
    a. Select *Memory* and give it at least 4GB of system memory.
    b. Select *Processors* and set the number of processors and processor cores based on your hardware
    c. Select *USB Controller:* for *USB Compatibility* select *USB 2.0* and check *Show all USB input devices* checkbox
3. On *Hardware* tab select *Hard Disk*:
    a. Click *Remove*.
    b. Click *Add*.
    c. On *Hardware Type* page select *Hard Disk* and click *Next*.
    d. On *Select a Disk Type* page leave the recommended settings and click *Next*.
    e. On *Select a Disk* page select *Use an existing virtual disk* and click *Next*.
    f. On *Select an Existing Disk* page click *Browse* and select the virtual machine that you have downloaded from Mizutech and click *Finish*.
    g. If a popup appears asking “Convert existing virtual disk to newer format?”, just click “Keep existing Format”
4. Click *OK* on the bottom of the Settings page.
5. Navigate to your virtual machine files on your hard disk and find the file with extension “vmx”
    a. Open this file with Notepad
    b. Go to the end of the file and add the following text in a **new line**: smc.version = "0"
    c. Save the file and close it.
6. Adjust the Network and other settings after your needs.

We have finally finished setting up the Mac OS development virtual machine. Just start the virtual machine.

For the MAC machine the login (admin) username is “Mizutech” and the password is “mizutech”.
Start your virtual machine and login to start working with your iOS SIP SDK and create your SIP application or integrate SIP into your existing project.
We do not recommend installing any upgrades as this might result in unbootable virtual machine - unless if new OS or XCode version is enforced by Apple. We also upgrade the virtual machine image usually once a year.

Development

To be able to work with the iOS SIP SDK, you will need some familiarity with MacOS, Objective-C and iOS development knowledge.

Both the SDK and the example project should build without errors or in case if you encounter any error at build time then it should be easily solvable by checking your source code and trying to fix the issue based on the error message.

The library is compiled for both iOS arrmv7 and arm64 architectures. G729 and Opus codecs and other patches are also added to the library. The library is backward compatible including iOS 8. It can run on both armv7 and arm64 architecture devices.

Documentations:

- General guide and documentation can be found here, here, here, here and here.
- Details about the SDK and API usage can be found here.
- The PSSIP developer guide can be downloaded from here.
- iOS specific documentation can be found here.

You can inspect the MizuDemo example and extend it after your needs or create your app integrating the SDK in similar way.

Folders:

The most important folders are the followings:

- pjproject-svn: PJSIP library built for iOS (the ARM version by default)
- pjproject-svn_X64Simulator: PJSIP for x86/x64 (rename to pjproject-svn if you need this)

Here is the full list:

- MizuPhone_SDK.xcodeprj: the test project to be opened
- MizuPhoneNew: generated app icons and launch images
- Classes: source code, required
- Mizu: source and icons
- Other: pjsip API
- Resources: project resources (not required for the SDK)
- Sounds: sound resources
- G729: not needed but can be kept if later modifications are required
- build: auto generated (not relevant, might be missing)
- Info.plist: test project plist

Note: You might not receive all these folders or the folder and file names might be slightly different in your deliverable, depending on your license, order and version.

Building PJSIP

In case if you wish to modify or rebuild PJSIP, follow these instructions:

Build pjsip for device:

- Open terminal
- Navigate to pjsip directory: cd Documents/Apps/MizuPhone/pjproject-svn/
- Run following command to clean pjsip: *make distclean*
- For each architecture pjsip needs to be built separately: *armv7* and *arm64*

- For armv7 run following commands

```
export
DEVPATH=/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer

export ARCH='-arch armv7'
```

./configure-iphone --with-opus=/Users/mizutech/Documents/Apps/MizuPhone/pjproject-svn/third_party/opus-dev-lib/ --with-ssl=/usr/local/openssl-its

```
make dep
make clean
make
```

- After build finished, copy the entire pjproject-svn and all of it's contents to directory named: armv7-pjproject-svn

- Run command: *make distclean* to prepare pjsip for arm64 build, and run the below commands:

```
export
DEVPATH=/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer

export ARCH='-arch arm64'
```

./configure-iphone --with-opus=/Users/mizutech/Documents/Apps/MizuPhone/pjproject-svn/third_party/opus-dev-lib/ --with-ssl=/usr/local/openssl-ios64 --disable-libwebrtc

```
make dep
make clean
make
```

- After build finished, copy the entire pjproject-svn and all of it's contents to directory named: arm64-pjproject-svn

- Run command: *make distclean* again in pjproject-svn directory

- Now we just need to combine the libraries from the two architectures by running the following shell script: lipoNEW.sh located in Documents/Apps/MizuPhone/ directory

- After script finished, we have the ready to use libraries in pjproject-svn directory

Build pjsip for iOS simulator:

- Open terminal
- Navigate to pjsip directory: cd Documents/Apps/MizuPhone/pjproject-svn/
- Make a backup copy of pjproject-svn directory
- Run following command to clean pjsip: *make distclean*

- Run the following commands:

```
export
DEVPATH=/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneSimulator.platform/Developer
ARCH="-arch x86_64" CFLAGS="-O2 -m32 -mios-simulator-version-min=8.0" LDFLAGS="-O2 -m32 -mios-simulator-version-min=8.0" ./configure-iphone --disable-libwebrtc
```

1. make dep
2. make clean
3. make

Useful stuff:

Combine/merge armv7 and armv7s libraries:

http://stackoverflow.com/questions/12670902/pjsip-for-ios-6-under-xcode-4-5-armv7-armv7s

```
lipo -output libgsmcodec-arm-apple-darwin9.a -create libgsmcodec-arm-apple-darwin9-armv7.a libgsmcodec-arm-apple-darwin9-armv7s.a
```

Generate certificate for voip push notification:
openssl pkcs12 -in pushkit.p12 -out pushkit.pem -nodes –clcerts

Sample project

We created a simple example project for your convenience to showcase the SDK usage.

A shortcut to the MizuDemo iOS SDK project directory can be found on the desktop.

Go to project directory and open MizuDemo.xcodeproj. This is a basic example on the usage of the pjsip iOS SDK.

MizuDemo can be installed and launched from XCode on an iOS device or on iOS Simulator.

Launch on device:

1. Connect your iOS device to you PC using your USB-Lightning cable
2. In VMware click *VM* in the top menu bar *-> Removable devices -> Apple iPhone ->* click *Connect(Disconnect from host)*
3. In XCode select your device and click Run.

Launch on Simulator:

1. To run on Simulator we need to change a few things:
    a. Go to MizuDemo project directory and rename *pjproject-svn_X64Simulator* to pjproject-svn (rename original *pjproject-svn* to something else)
    b. In XCode click on MizuDemo project in the left pane, go to *Build Settings* and change *Valid Architectures* from ***armv7 arm64*** to **x86_64**
2. Select the Simulator in the top-left corner in XCode and click Run.

Do not forget to reverse the above-described changes to be able to run the project on a device or to be able to publish it.

Publish

Once you finished the development of your app, you can publish it on the Apple App Store.
Here are a few hints:

Project settings

- target: armv7 amr64
- XCode -> Preferences -> Accounts -> add your account
- Project -> Target -> Signing Capabilities (All; always All)
- Product -> Archive -> Publish

AppStore publishing

1. Create an Apple Developer account, if you do not already have one, and enroll in the iOS Developer program.
2. Create Distribution certificate, App ID and Distribution Provisioning profile on the developer.
3. Go to XCode Menu bar -> XCode -> Preferences -> Accounts and add you developer Apple ID here.
4. Build the iOS softphone with XCode: Menu bar -> Product -> Archive.
5. Upload and publish your app with App Store Connect

Certificates

You can find the certificates at developer.apple.com -> certificates, identifiers & profiles:

- iOS Distribution: this is the most important to sign the app
- Developer ID Application and Installer -> required for MacOS
- iOS Development: for testing to be able to install app to my own device
- Apple push services: for the VoIP server or push gateway (not for XCode or for app)
- VoIP Services: for better push (optional for voip apps)

A more in-depth MizuPhone specified publishing guide can be found here (see the “Publish” chapter).The steps listed in this document is very similar to any other iOS app.

Support

We provide the source code “as-is” with no additional support included.
We do not include iOS development services, iOS support services or iOS consultancy services.

We always test the delivered source code and it compiles/builds correctly on our test machines generating a correctly working softphone application. Please send a detailed problem description to support@mizu-voip.com if you find any issue. We cannot guarantee direct support, but we always consider all bug reports and we release new versions periodically with improvements and bug fixes.

Resources

- Mizutech home page
- iOS SDK home page
- Contact