FeenPhone Non-Windows Developer Protocol Guide

By
MichaelWDean

FeenPhone source code is on GitHub, HERE.

.

Licensed BipCot NoGov License, version 1. More info HERE.

.

OVERVIEW / DEFINITION section by Michael W. Dean. PACKETS and AUDIO PARAMETERS sections by Derrick Slopey:

.

OVERVIEW / DEFINITION OF FeenPhone:

FeenPhone is free NoGov (*BipCot NoGov License, version 1*) decentralized P2P true-duplex mono VoIP software designed with highest quality audio in mind. FeenPhone uses Opus codec over TCP. Default Opus 24KHz (32kbps).

.

FeenPhone is made for people who wish to produce broadcast-quality conversational spoken audio with people who are not in the same location.

.

FeenPhone uses Opus codec and NAudio, plus a bunch of original code. The only component that is imperative to use in porting the functionalities in other OSes is the Opus Codec and the packet protocol.

.

Unlike all other VoIPs, FeenPhone contains no echo cancellation, no noise reduction and no audio compression. This helps make the audio quality higher than with other VoIPs. But this also means that FeenPhone must be used in a quiet room with a decent mic, a windscreen, the mic close to the talker's mouth, and most importantly, with headphones. The headphones should be closed-ear, not open-ear, to prevent echo.

.

FeenPhone could be used without headphones, but only in a push-to-talk system where only one person speaks at a time. This would be a possible application for FeenPhone for mobile devices. The microphones on iPhones, Android phones and Windows phones are fairly good, the limitations of the phone network and / or the limitations of various VoIPs commonly used on these devices are the reason they don't sound great. FeenPhone could sound great on any of these devices if set up to be push-to-talk in one direction at a time. (Read the last paragraph of the "Uses for FeenPhone" section on the front page of this website for more on push-to-talk uses for FeenPhone.)

.

If you add echo cancellation, noise reduction or audio compression to FeenPhone, it will cease to be FeenPhone. If that's your plan, you should probably use different VoIP code that already has those features anyway.

.

FeenPhone will work with any microphone, but is designed for, and sounds best with, a good cardioid dynamic USB mic, with a windscreen, on a stand, near the mouth of the user, in a quiet sound-conditioned room. The recommended specific mic is the $50 Audio-Technica AT2005USB. The makers of FeenPhone tested many microphones, and made FeenPhone with this mic in mind.

.

FeenPhone will work with gamer headsets, but those don't sound great. FeenPhone will work with USB condenser mics, but they're too sensitive, and will pick up background noise. FeenPhone will work with electret condenser mics on iPhones, iPads, Android devices, Windows Phones and Windows pads. This could sound fairly good, if the talker

on each end is a few inches away from the device, and in a quiet room. As stated above, users on both ends would have to use headphones (*not* ear buds), or only use FeenPhone in one direction, in a push-to-talk capacity.
.
Low buffer sizes for input and output are important for low latency, so audio drivers which work closest to the hardware are preferred. FeenPhone is client-server, and currently the server is responsible for redistributing the audio from each client to the others. The server will have the lowest audio latency, but requires the most bandwidth and processing power.
.
FeenPhone could be built for many of different platforms, but on some of them it may make more sense to just rebuild from scratch, using the same design goal (no noise cancellation, made to use for high-quality spoken audio in a quiet room using headphones and a good mic), and the same codec, codec settings, and the FeenPhone-specific packet protocol. A lot of the work on FeenPhone itself was concerned with interfacing with the audio system and handling system-level audio device stuff, this won't likely translate well to other systems.
.
The networking engine may work well on Mono, but the audio system will not.
.
It will be a lot easier to simply start by making *Clients* for Linux, Android and Apple than to make full Server / Clients like our Windows version.
.
As of version beta v0.1.5492.36528 (released 1/18/15), there is no encryption in FeenPhone. Encryption is a planned feature to add later. Encryption was not included in the working proof-of-concept beta version because FeenPhone was originally created for producing media with the intention of sharing that media to the public, and we had a lot of features to add. Adding encryption to FeenPhone will make FeenPhone the best-sounding secure VoIP in existence. This is because encryption, done right, will not degrade the audio quality of FeenPhone. And FeenPhone is *already* the best-sounding VoIP in existence.
.

PACKETS:

Currently the packet writers are defined in FeenPhone\Packets\Packet.cs

Readers can probably be inferred from them. The Packet readers/writers could use some cleanup and that is intendec to be done before the next release.

Basic packet structure:

> *[byte packetId][ushort payloadLength][byte[] payload]*

Encoding:

> *All numbers are little-endian unless otherwise specified.*
>
> *A ushort is a little-endian two-byte number, for example 258 is encoded as [0x01][0x02]*
>
> *A bool is a single byte 0 or 1*
>
> *Text is Ascii encoded, and is not prefixed with a length unless otherwise specified. Most text was put at the end of the message so that its length can be calculated from the payload length.*

Packet Ids:

- Chat = 2,
- LoginStatus = 5,
- LoginRequest = 6,
- UserList = 12,
- UserLogout = 14,
- UserLogin = 15,
- Audio = 16,
- PingReq = 22,
- PingResp = 23,

Structures:

UserData

[ushort length][AsciiEncodedText]The user data structure contains a unsigned 16bit short little-endian length of the ASCII encoded text data followed by the text data which is a tab delimited list of the follow string values:UserID: a guid string in the format shown in the example.
Admin: “1” or “0” indicating whether the user is an administrator
Username
NicknameText data example:”0989547f-9986-4319-a0e2-eeeab91a9137\t0\tderrick\tderrick”

Packets:

Chat (Client, Server):

[0x02][ushort payloadLength][UserData][AsciiEncodedText]Note: The acsii encoded text is not prefixed with a length, but will be equal to the payloadLength-UserDataLength. The UserData length is specified in the first two bytes of the UserData structure.

LoginStatus (Server):

[0x05][ushort payloadLength][bool loginStatus][AsciiEncodedText]

AsciiEncodedText:

An optional message from the server. If omitted, payloadLength will be 1

Client should prompt user for login credentials if loginStatus==0

LoginRequest (Client):

[0x06] [ushort payloadLength][AsciiEncodedText]

AsciEncodedText

Tab delimited username then password. Like: “derrick\tpass”

Server will reply with LoginStatus packet.

UserList (Server):

[0x0C][ushort payloadLength][byte count][UserData….]

Count

The number of user structures contained in the packet.

UserInfoStructures

The UserInfo datas for each connected user.

UserLogout (Server):

[0x0D][ushort payloadLength][][UserData]

UserData:

The User structure for the user which logged out or disconnected

UserLogin (Server):

[0x0E][ushort payloadLength][][UserData]

UserData

The User structure for the user which logged in

Audio(Client, Server):

[0x10][ushort payloadLength][byte codecID][bool containsGuid][optional byte(16) userID][AudioData]

codecID

This is the defined codec under which the audio data is encoded. See the Audio Codecs section for more information.

containsGuid

True if the UserID is specified, used by server for broadcasting other user's data. It is always false when the client is sending its own audio to the server, or if the server is sending its own audio to clients.

UserID:

If containsGuid is true, this is a 16-byte representation of the guid UserId which the audio was received from.

AudioData:

The encoded audio stream

The Server is responsible for echoing any audio received from clients to all the other clients. There is no mixing done on the server, each client receives a discrete audio stream from each user. It is preferred that the server add the received audio to its own play buffer before sending the packets to the clients to minimize local latency (at the expense of remote latency).

PingReq (Client, Server):

[0x16][0x02][ushort timecode]

Timecode

A 16-bit timestamp generated by the requestor. The source if the timestamp does not matter because the remote device simply echoes this code back to the requestor. In the .Net implementation the total milliseconds since program start is used for this value.

Upon receipt, the PingResp message should be sent back containing the same timecode which was sent.

PingResp (Client, Server):

[0x17][0x02][ushort echo]

echo

The 16-bit value that was sent in the request.

AUDIO PARAMETERS:

These are the most basic parameters for encoding the audio in supported formats. Please refer to the documentation for each codec for additional implementation information. The left prefixes each codec name with the CodecID as

specified by the packet protocol for the AudioData packet.

Currently Supported CodecsIDs:

43: OpusCodec24kHzVoip32768
Channels: 1Opus segment frames: 960bitRate: 32768outputSampleRate: 24000codingMode: Voip = 2048
77: OpusCodecAudio48kHz65536
Channels: 1Opus segment frames: 960bitRate: 65536outputSampleRate: 48000codingMode: Audio = 2049
140: G722ChatCodec
Channels: 1bitRate: 64000outputSampleRate: 16000Flags: None
106: Uncompressed8KHzMonoPcmChatCodec
Channels: 2bitRate: 128000outputSampleRate: 8000Sample format: 16bit

The non-Opus codecs / audio formats (G.722 and PCM) and non-TCP protocols (the options for UDP and for Telnet) in FeenPhone are optional and are not needed in a derived version. The client that does not support a codecID should gracefully decline to play the audio, and perhaps inform the user that the codec is unsupported.

.

UPDATE 2/25/15: While G.722 and PCM are still in the code of the program, we've removed the ability to access them from the Interface. They don't sound nearly as good as Opus. And G.722 and PCM will not allow FeenPhone to interface with hardware devices that use those formats, because FeenPhone's unique and better packet protocol will not work with that hardware.

.

Many people will say that UDP should be used instead of TCP. We found TCP works better with Opus under our packet protocol.

=-

Please contact us HERE or post a comment below if you're working on a non-Windows version so people can coordinate. And we're available for testing your version over the Internet, and will offer feedback free. If you work with us on it and it rocks, we may help you promote it when you're done. Thank you!