SKYPE EVENT STORAGE

MAX KLINGER
HTTP://TEHSUCK.DE/SKYPE.HTML

Contents

1. Introduction

Skype (Linux, MacOS X and Skype for Windows before Version 4) uses a number of files ending in .dbb. The format is apparently derived from Kazaa's. The format is not really cryptic, it just lacks decent documentation.
If you are looking for information about the format used in Skype Version 4 and above read up on sqlite and have a look into main.db.

1.1. **Ressources.** There are already a couple of places where there is information about the format, that I am aware of:

- The largest body seems to be created by someone at lpcforencsics.
- There is a C - program by Dr. Nael Krawetz with some comments here
- Johannes Buchner also did some analysis and wrote a python extractor, c.f. this page

I hope to include all information that i consider correct in my analysis over time, so hopefully this will be the only ressource you need.

1.2. **Validity.** I took great care to test all my assumptions on the logs available to me, which at the moment is a body of just shy of 100,000 events with hardly anything but messages most of which fit within the 256 bytes limit (see below). This situation is less than ideal, since i have little information about events other than messages and related. On the other hand, I think my usage habits are not too far from representative and message events are the most important ones to me right now anyway.

To help you I will clearly mark untested assumptions and conjectures so you know where to tread lightly.

1.3. **Preliminaries.** A lot of things are shared among the many different types of events/data logged by Skype. These apply to all of these unless noted otherwise.

1.3.1. *Location.* The Skype files where the logging takes place are all located under one folder and then filed under subdirectories named after the username. This way more than one account can be used with the same computer login account. This main folder is located under:

Windows: C://Documents and Settings/[Username]/Application Data/...
Skype/[Skype Account]/
Linux: /.Skype/[Skype Account]/
MacOS X: /Library/Application Support/Skype/[Skype Account]

1.3.2. *Files.* Located in this directory you will typically find a selection of files ending in .dbb (among others). The filename consists of a category like chatmsg or user which denotes the class of events stored (although there is more than one type of event in this file typically) and a number denoting how long a record is in bytes - 8. For example **chatmsg256.dbb** contains several chat message type events with a total length of 264 bytes including all tags.

The nominal record sizes I have observed thus far run from 256 to 32768 Bytes in powers of two. [1]

Files are created as needed, e.g. you will likely only have one profile file since you only need to store your profile once. Items are stored in the smallest length format possible taken into account the tags attached. It is possible for this rule to be violated because chat message events that are altered later on are still stored in the same place even if shorter afterwards. For more details to the intricacies involved confer the section about chat messages. Events are always null padded at the end to achieve the desired size, i.e. the next power of two.

From what i have gathered thus far the mapping should be as follows

chat: ? contains the first text message event of every chat whyever

[1] I know there is an upper limit on the size at least programmatically since Skype will bail with an error if you try larger messages but I haven't been motivated to figure if this is 32768 Bytes.

chatmsg: stores mood messages, chat initiation events, normal chat messages and more.

call: Skype call events

callmember: Supports the feature of conference calls but is used for every call?

chatmember: Same as callmember but for chats

contactgroup: There are a number of predefined contact groups whose's status is stored here alongside user defined ones.

profile: Your own user profile

transfer: Metadata for any transfers you initiated or received.

user: Profiles of people you know, same format as profile

voicemail: Voicemail metadata.

1.4. **Data formats.** These formats only hold for the events themselves, the header 2.1 is different.

Strings: are NULL terminated, sometimes Double NULL terminated, sequences of utf-8 chars. I have a feeling there is a system to the termination but I haven't figured it out yet.

Numbers: are stored as seven bit continuated little endian integers, similar to the way utf-8 works. You check if your number fits into the next seven bits, if yes store it and be done, if not store 7 bits of it, set the highest bit to 1 and continue with the rest in the next byte. This allows to store integers of arbitrary length.

Times: are stored as unix time stamps, i.e. have a precision of one second and stored with the same prescription as other integers. This means that given that unix time is 4 bytes atm we end up with 5 bytes in the 7 bit scheme.

2. General Format

Every Event is started with the same 16 Byte header unless the event was deleted in which case you have only the first 4 Bytes non NULL[2]. The actual events start straight after the header.
The header format is simple

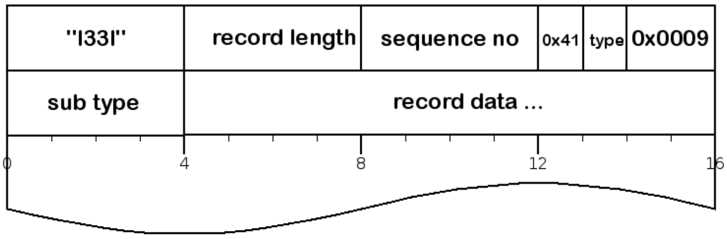
2.1. **Header Format.**

- A 4-Byte marker (0x6c33336c = "l33l") to label a new record
- A 4 Byte little-endian integer telling you how long the rest of the record is starting after this. [3]

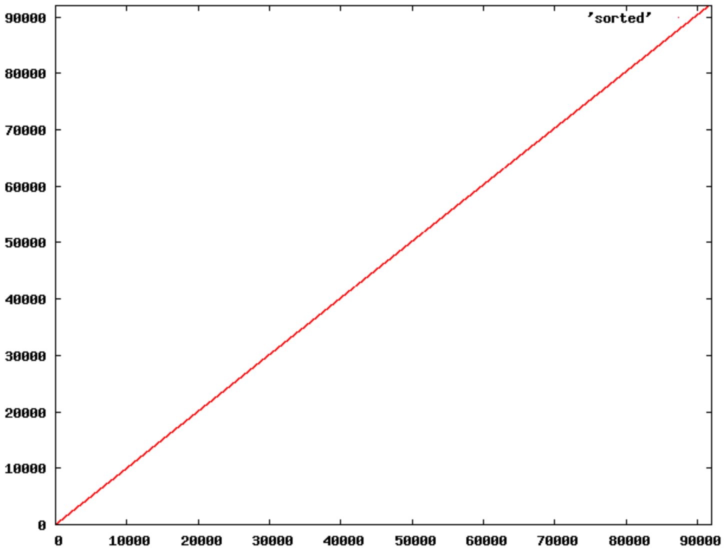
[2] This really is consistent because that also means the size field is zero and thus the rest of the record

[3] **Note:** The space this record occupies is still the next bigger power of two plus 8 Bytes. The difference is just NULL padding.

- A 4 Byte little-endian integer that constitutes a sequence number, unique across all types of events.
- A 1 Byte marker (0x41 = ”A”) to label the ID?
- A 1 Byte big-endian![4] int used as a tag with information about the type of event. I first thought this is unique but then I found several clashes. [5]
- A 2 Byte marker (Always 0x00 0x09) Probably demarks ”end of header”.
- A 2 Byte field (likely little endian) that i presume is something along the lines of a subtype. [6]



2.2. **Sequence number.** The sequence number is unique across all events and not a lot of them are lost. I plotted most of my events in the following Figure.



If you have better eyes than me you can see that the slope of the line is almost 1, but only almost; it deviates very slightly from $\phi = \pi/4$. This is due to a few omitted/deleted numbers. This can happen for various reasons I believe, the most

[4] I am not really certain of this yet but it makes so much more sense if it were big-endian

[5] I haven’t really made sense of that either. I believe, it would be unique again if you also consider the 2 bytes after the ”0009” but I haven’t proven that yet. And then why would anyone do that in any case? They are nowhere close to exhausting the 256 values they would theoretically have with one Byte.

[6] I am not really sure if this is part of the header but then the type isn’t sensible anymore

important of which is if you deleted a chat message. On my log less than roughly 0.4% were missing.
The uniqueness can be checked with

```
$ dumptags file | sort -u -n > unique
$ dumptags file | sort -n > nonunique
$ diff unique nonunique
```

Dumptags is part of the tools I have written along with this analysis and available from the website stated in the title.

2.3. **Tags.** I haven't made an effort yet to figure out every tag but the current list of my guesses is this

0x4102-0x4105: Contact Group related (difficult topic see there)
0x4106: Also Contact groups, albeit somewhat more complicated even, can denote an empty user defined group or a predefined one under unclear conditions.
0x4107: Manually created contact group that is not empty, maybe also something call related.
0x4108: Call
0x4109: Call
0x410a: Multi user chat initialization message
0x410b: Single user chat initialization message
0x410c: Multi user chat message or voice Message
0x410c: Single user chat message
0x410e: Mood messages most of the time (exception were unsent messages to a person that didn't authorize me)
0x410f: Multiuser Chat related or file transfer notification
0x4110: can mean a message that was changed later or one that is deleted (tricky topic see chats)
0x4111: seems to always mean deleted messages
0x4113: seems to be user profile
0x4114: seems to be user profile
0x4115: seems to be user profile
0x4118: some type of chat message
0x4119: some type of chat message
0x411a: Your User Profile and the introductory chat message by the Mood Message Feed.

3. Chat Messages

coming soon...