Serato Library Specification

Contents

Overview

This document specifies the Serato library storage format. The format is compatible across the four applications:

- Serato DJ
- Serato DJ Intro
- Scratch Live
- ITCH (which has been superseded by Serato DJ)

The library is shared across these applications, however care must be taken with regards to compatibility when used with legacy versions of Scratch Live, i.e. prior to version 2.0.

On the file system, Serato library is stored as a collection of flat binary files, the format of which will be described in this document.

Please note that the Serato library grew somewhat organically over a period of 10 years and may seem unusual. Ensuring forward and backward compatibility had been a major contributor to this. This required keeping old concepts while introducing new ones, and has ultimately resulted in some level of redundancy in the way the library is stored on disk.

Important concepts

Database

The database is a collection of references to all audio files imported into the library. The database inherently does not have a way of structurally (hierarchically) organising the tracks it refers to. It is a flat list of tracks, a concept originating all the way from Scratch Live 1.0. Structuring and categorising tracks is actually provided by the crate system. Likewise, tracks that don't belong to any crate are only kept in the database file.
However, the database does have another very important role and that is caching of audio file metadata. This is necessary so that the application can load quickly with a great amount of track information available on start up. It's important to note that is only a cache. The underlying metadata is sourced either directly from the audio file, or if that's not possible external metadata. External metadata is described in a separate document

Crates

Serato's track library has a concept of *Crates*. Creates are like directories, providing a hierarchical structure to the track library. Crates are simply a collection of references to audio files on the file system, or other crates (called sub-crates) for the purpose of categorisation. However they do not necessarily have to mirror the directory structure on the file system. Effectively, a crate can contain files from any number of different locations, and can be arranged in any way

Distributed structure

The Serato library can exist in pieces when multiple logical drives are attached to a system. Each of these partial libraries is responsible for storing information for tracks located on that particular drive. This allows for sharing and portability of partial track libraries between different systems. A DJ can have an external hard drive and use it across multiple systems. It is also portable between a Mac OS X and a Windows system.

Each system that a Serato application runs on also has a primary library, which is responsible for tracks imported off the primary drive. On Widows this is the “C:” drive. On Mac this is the primary volume, usually “Machintosh HD”. In addition to track location records, the primary library also stores other information related to that particular system, such as playback history.

Location

All the related library files are bundled inside a directory named:

```
_Serato_
```

It’s important to note that alongside the track library records, this directory also stores various other information such as saved MIDI mappings, log files, external metadata…etc. New files may be added in future versions too. So care must be taken not overwrite non-library information unless that is actually desired.

On the primary drive, the location of this folder is platform dependent and is also stored per user account. On Windows, this would be:

```
C:\Users\[username]\Documents\Music\_Serato_
```

On Mac OS X it is:

```
/Users/[username]/Music/_Serato_
```

On secondary drives, the _Serato_ folder is located at the drive’s root level:

```
D:\_Serato_
```

On Mac:

```
/Volumes/MyDrive/_Serato_
```

The application saves a number of files in the _Serato_ folder. Not all of them relate to the track library. These are the files or directories that do relate to the track library:

Name	Type	Description
Subcrates	Directory	Stores all the crates/sub-crates related to the files on this drive
database V2	File	The database file containing a reference to all imported files on this drive, and their cached metadata
neworder.pref	File	Stores the order that crates and sub-crates should be displayed in.

Common format specification

The files in _Serato_ folder that store track library data have some common format characteristics. The files are a simple sequence of short snippets of data called tags. A tag may contain simple data or a collection of child tags.

Primitive types

Before defining the structure of the tag, let's look at the primitive types that can be stored in the payload. All the types are stored in big endian byte order.

Type	Size in bytes	Description
boolean	1	A true/false value (encoded as a 1-byte integer)
uint16	2	A 2-byte unsigned integer
uint32	4	A 4-byte unsigned integer
string	Inferred from the tag length	A UTF-**16** encoded string. As with other types, the byte order is assumed to be big endian. A standard two-byte pair explicitly specifying such order is not present in the data.
blob	Inferred from the tag length	Tag specific binary data.

Tag structure

The common structure of a tag is simple.

1. The tag is identified by the first four bytes with an ASCII string.
2. This is followed by the Tag Length, which specifies the total size of the tag (including the identifier) in bytes.
3. Then it is followed by the actual data. The content is deduced from the identifier

Type	Identifier (ASCII)				Tag Length (Unt32 value)				Data
Offset	0	1	2	3	4	5	6	7	8…

Common tags

The definitions for known tags are explained in the appropriate section for each specific file. This is because each file has a different set of allowed tags. However it is worth describing some prototypical tags.

The first byte of the tag identifier is always the tag **type identifier**. Here is a table of allowed tag types, with the associated one-byte identifier

Tag type	Type Identifier (ASCII)	Data/payload type	Description
Version	v	string	Contains a string specifying the file identifier and the format verison. For more information consult the 'vers' tag specification
Boolean	b	boolean	Contains a boolean value
UShort	s	uint16	Contains an int16 value
ULong	u	uint32	Contains an int32 value
Text	t	string	Contains text as a string. The size of the string is inferred from the tag length, i.e. (TagLength – 8)
Path	p	string	Contains a special form of the Text type. Although it has the same structure, it is to be used exclusively to store a file system path. The path hierarchy levels must be encoded using the forward-slash (/) character. Thus on Windows the C:\Users\DrEvil path is to be encoded as C:/Users/DrEvil
Blob	a	blob	Contains tag specific binary data
Object	o	tag-specific	A compound tag, which is defined on case-by-case basis. This can either be a sequence of primitive types or a sequence of child tags.

vrsn tag

All library files that utilize the tagging system share this tag. It must appear at the very start of the file, and defines the type of the file and the version.

The vrsn tag is a Text tag with the payload of the following format

```
{major_version}.{minor_version}/{file_identifier}
```

- {major_version} – the major version of the file format
- {minor_version} – the minor version of the file format
- {file_identifier} – a string identifying the file format. This is specified in each of the file format sections.

For example for a database file, the vrsn tag may look like this:

```
2.0/Serato Scratch LIVE Database
```

Database file

Stores a cache of tracks and its common metadata for fast load on start up. Also contains the tracks that don't belong to any particular crate. The data is cached upon import of a track, but also updated whenever a user launches the "Scan ID3 tags" command, or whenever a song is loaded onto a virtual deck.

Windows location:

```
C:\Users\[username]\Documents\Music\_Serato_\database V2
```

Mac location:

```
/Users/[username]/Music/_Serato_/database V2
```

The first tag in the database file must be the vrsn tag, the version tag. This identifies the file, but also tells us what version of the format is used. The following is the identifier to be used in the vrsn tag.

```
Serato Scratch LIVE Database
```

Version 2.0 Database tags

otrk - Track object tag

Description: This tag identifies a track in the track library, along with its location and a cache of simple metadata.
Type: Object tag
Data: This tag contains an array of child tags from the following table.

pfil	tgrp	ulbl
pvid	trmx	utme
ttyp	tlbl	uadd
tsng	tcmp	ufsb
tart	ttyr	sbav
talb	tcrt	bhrt
tgen	tadd	bmis
tlen	tcor	bply
tsiz	tvfx	blop
tbit	tkey	bitu
tsmp	tiid	biro
tbpm	utkn	bovc
tcom	udsc	bcrt

The pfil tag associates the track with a file on disk, and the remaining tags are a cache of file's metadata. For information on each of these tags refer to the next section (Other tags)

ortk

Description: This tag is an action tag, and specifies that Serato software should action a track delete track the next time it is able to, for example on start-up. You can remove the appropriate track tag yourself when no Serato software is running, but otherwise it is safer to let Serato software do it through this action tag.
Type: Object tag
Data: This tag contains one child pfil tag, which specifies the path of the file to be removed

Other tags

These are the descriptions of other simple tags, which cannot exist on their own. They must be part of one of the other top-level tags.

Tag id	Description
Path tags (contains an identifier, size and a string value)	
pfil	Contains the file system path of an audio file
pdir	Contains the file system path of a directory
pvid	Contains the file system path of a video file which is associated to another file with audio content.
Text tags (contains an identifier, size and a string value)	
ttyp	File type identifier. Has to be one of: mp3 wave oggvorbis aiff

	quicktime streaming flac unknown
tsng	Name of the track, as read from the file metadata
tart	Name of the artist, as read from the file metadata
talb	Name of the album, as read from the file metadata
tgen	The genre, as read from the file metadata
tlen	The track length, as read from the file metadata
tsiz	The track file size, as read from the file metadata
tbit	The file bitrate, as read from the metadata
tsmp	The file sampling rate, as read from the metadata
tbpm	The file BPM, as read from the metadata
tcom	The track comment text, as read from the metadata
tgrp	The track grouping tag, as read from the metadata
trmx	The name of the remixer as read from the metadata
tlbl	The track record label, as read from the metadata
tcmp	The track composer, as read from the metadata
ttyr	The track year, as read from the metadata
tcrt	The name of the crate this track belongs to
tadd	The date this track was added to the Serato library
tcor	A tag that flags the associated track object as corrupt. The presence of this tag indicates that Serato previously tried and was successful at reading the track. The payload is the human readable text message explaining it what way it is corrupted.
tvfx	List of associated video effects. This tag should not be created externally
tkey	The track key, as read from the metadata
tiid	The iTunes track identifier if it was imported from iTunes
ULong tags (contains an identifier, size and a uint32 value)	
utkn	The track number, as read from the metadata
udsc	The track disc number, as read from the metadata
ulbl	The track colour label, as read from the file metadata. This is a 32-bit integer ARGB value. Note that the alpha channel is not actually used.
utme	A timestamp indicating when last time track metadata was modified, in “Unix time” (Seconds since January 1st, 1970)
uadd	A timestamp indicating when the track was added, in “Unix time“ (Seconds since January 1st, 1970)
ufsb	The track file size in bytes, as read from the file metadata
UShort tags (contains an identifier, size and a uint16 value)	
sbav	[Deprecated] A version for when this file was last analysed by Serato software
Boolean tags (contains an identifier, size and a boolean value)	
bhrt	Indicates whether the metadata was read from the file
bmis	A flag indicating that the file is missing
bply	Indicates whether the file was recently played.
blop	Indicates whether the track should be repeated when the end is reached. This is read from the file metadata

bitu	Indicates whether this track is coming from the iTunes library
biro	Indicates whether this track was a “read only” file the last time Serato software attempted to write to the file
bovc	Indicates whether the Serato software has completed the analysis process of this track
bcrt	A flag indicating if the file is corrupt or not.
bwlb	Indicates whether the file is a Whitelabel file
bwll	Indicates whether the file is an access controlled Whitelabel file
buns	Indicates whether the file is unsupported (or not). True – unsupported False or no tag – supported
bbgl	Indicates whether the beat grid is locked

Crates/sub-crates

Defines the crate hierarchy in the Serato track library, and their track associations. Each crate is defined using one file in the following directory:

Windows:

```
C:\Users\[username]\Documents\Music\_Serato_\ Subcrates\
```

Mac:

```
/Users/[username]/Music/_Serato_/Subcrates/
```

The file name is what defines the name of the crate. This is the only place a crate name is defined.
The extension of the crate file should be:

```
.crate
```

The hierarchy of the sub-crates is achieved using the following file naming scheme. The hierarchy level is delimited using the %% characters.

```
level1[%%level2 ...]
```

So, suppose you have a “Top40” sub-crate, inside an “80s” sub-crate, inside a “Pop” crate. The file for this will be named:

```
Pop%%80s%%Top40.crate
```

Identifier:

```
Serato ScratchLive Crate
```

The first tag in a create file must be the vrsn tag. This identifies the file, but also tells us what version of the format is used.

Version 1.0 Crate tags

otrk

Description: This tag specifies that a particular track belongs to this crate.
Type: Object tag
Data: This tag contains just one child ptrk tag that specifies the path of the track

Tag id	Type	Description
ptrk	Path	The path of the track on filesystem

ortk

Description: This tag is an action tag, and specifies that Serato software should action a track delete track the next time it is able to, for example on start-up. You can remove the appropriate track tag yourself when no Serato software is running, but otherwise it is safer to let Serato software do it through this action tag.
Type: Object tag
Data: This tag contains one child ptrk tag, which specifies the path of the file to be removed. The ptrk tag is defined above, in the otrk section

osrt

Description: This tag specifies how the column information will be sorted.
Type: Object tag
Data: This contains the following tags, in that order:

Tag id	Type	Description
tvcn	Text	The name of the column to sort. The names are specified in Column Names section
brev	Boolean	Indicates the sort direction: True – reverse False – forward

ovct

Description: This tag enables visibility of a particular column
Type: Object tag

Data: This contains the following tags, in that order:

Tag id	Type	Description
tvcn	Text	The name of the column to make visible. The names are specified in Column Names section
tvcw	Text	Specifies the column width, formatted as an integer inside a UTF-16 string

orvc

Description: This tag is an action tag which specifies that Serato software should remove visibility of a particular column
Type: Object tag
Data: This contains just the tvcn tag, which is specified in the tag above (ovct)

Column names

This is the list of available column names. They match up to a subset of the available metadata.

added	comment	key	samplerate
album	composer	label	size
artist	filename	length	track
bitrate	genre	location	video track
bpm	grouping	remixer	year

Crate order

The crate ordering is specified through the following file in the _Serato_ folder:

```
neworder.pref
```

It is a UTF8 encoded text file, and has a simple structure.
The data is enclosed in a [begin record] [end record] segment, for example

```
[begin record]
…data…
[end record]
```

The data is a list of line end terminated entries of crate names with the following format:

```
[crate]{name_of_the_crate}
```

For example, suppose this is the crate order:

- crateC
- crateB
- crateA
    - subcrateX

The crate order file will look as follows:

```
[begin record]
[crate]crateC
[crate]crateB
[crate]crateA
[crate]crateA%%subcrateX
[end record]
```

History

history.database contains references to session files which contains all information about tracks played in each session. All history files follow the “Common format specification”.

Windows location:

```
C:\Users\[username]\Documents\Music\_Serato_\History\
```

Mac location:

```
/Users/[username]/Music/_Serato_/History
```

History and session files all start with the vrsn tag, to declare that version 1.0 tags are used in these files. Session files are stored under the Sessions folder.

Version 1.0 History Tags

The following tags belong to history.database.

ocol

Description: This tag enables visibility of a particular column object under History

Type: Object tag

Data: This contains the following tags, in that order:

Tag id	Type	Description
ucok	UInt32	Column key
ucow	UInt32	Column width

Column keys are specified in the section “Entry Data Keys” below.

orco

Description: This tag identifies the column object to be removed

Type: Object tag

Data: This contains the following tag

Tag id	Type	Description
ucok	UInt32	Column key

oses

Description: This tag identifies the session object

Type: Object tag

Data: This contains the following tag

Tag id	Type	Description
adat	Blob	Entry data in binary blob

orse

Description: This tag identifies the session object to be removed

Type: Object tag

Data: This contains the following tag

Tag id	Type	Description
uses	UInt32	Session unique ID

The following tags belong to .session files

oent

Description: This tag identifies the entry object

Type: Object tag

Data: This contains the following tag

Tag id	Type	Description
adat	Blob	Entry data in binary blob

The tags used in the binary blob are specified in the section “Entry Data Keys” below.

oren

Description: This tag identifies the entry object to be removed

Type: Object tag

Data: This contains the following tag

Tag id	Type	Description
uent	UInt32	Entry unique ID

Entry Data Keys

The following are the data keys used in the binary blob in session files. They are also used as column keys in history.database.

Index	Description	Type
0	Unknown Entry Data	binary
1.	Entry Data UniqueID	UInt32
2.	Song Full Path	tchar(wide character)
3.	Song Folder	tchar(wide character)
4.	Song Short File Name	tchar(wide character)
5.	Song Unique ID	UInt32
6.	Song Name	tchar(wide character)
7.	Song Artist	tchar(wide character)
8.	Song Album	tchar(wide character)
9.	Song Genre	tchar(wide character)
10.	Song Length	tchar(wide character)
11.	Song File Size,	tchar(wide character)
12.	Song File Size in Bytes	UInt32
13.	Song Bit rate	tchar(wide character)
14.	Song Sample Rate	tchar(wide character)
15.	Song BPM Integer	UInt32
16.	Song BPM	double

17.	Song Comments	tchar(wide character)
18.	Song Comment Language	char
19.	Song Grouping	tchar(wide character)
20.	Song Remixer	tchar(wide character)
21.	Song Record Label	tchar(wide character)
22.	Song Composer	tchar(wide character)
23.	Song Year	tchar(wide character)
24.	Song Rating	tchar(wide character)
25.	Song Date Added	time_t
26.	Song Track Number	UInt32
27.	Song Number of Plays	UInt32
28.	Song Start Play time	time_t
29.	Song End Play time	time_t
30.	Song Play time	time_t
31.	Song Played On Deck	UInt32
32.	Song Percent Played	float
33.	Song Whitelabel Track	bool
34.	Song Order	UInt32
35.	Song Associated Video Full Path	tchar(wide character)
36.	Song Associated Video Relative Path	tchar(wide character)
37.	Song Associated Video Short Path	tchar(wide character)
38.	Song Video Effects Array	tchar(wide character)
39.	Song Played In Offline Player	bool
40.	Review Period	tchar(wide character)
41.	Review Session Name	tchar(wide character)
42.	Review Session Collapse State	bool
43.	Review Session Start Time	time_t
44.	Review Session End Time	time_t
45.	Review Session Length	time_t
46.	Review Session Location	tchar(wide character)
47.	Review Session Comment	tchar(wide character)
48.	Song Review Session ID	UInt32

49.	Song Notes	tchar(wide character)
50.	Song Passed Review Threshold	bool
51.	Song Key	tchar(wide character)
52.	Song Recently Played	bool
53.	Review Entry Last Edit	time_t
54.	Review Session Last Edit	time_t
55.	Review Session SMS ID	tchar(wide character)
56.	Review Session Last Submit	time_t
57.	Review Session Application	tchar(wide character)
58.	Review Session Version	UInt32
59.	Review Session UTC Offset	time_t
60.	Review Session Needs Clearing	bool
61.	Review Session Max Entry ID	UInt32
62.	Review Session GUID	tchar(wide character)
63.	Review Session Primary Device	tchar(wide character)
64.	Song Comment Descriptor	tchar(wide character)
65.	Song Status	UInt32