

Playlist Manager
(aka Playlist-Manager-SMP)

regorxxx

September 3, 2021

Contents

I	Features	5
1	Managing Playlist files and Auto-playlists	5
1.1	Auto-playlists	5
1.2	How paths are written: absolute and relative paths	5
1.3	Playlist loading	6
1.4	Autosaving and Autoloading	6
1.4.1	Autosaving	6
1.4.2	Autoloading	7
1.5	Playlist binding	7
1.5.1	Deleting and restoring files	7
1.6	Playlist metadata	7
1.6.1	Lock state	7
1.6.2	[Playlist] Tags	8
1.6.3	Track tags	8
1.6.4	Category	9
1.6.5	UUID	9
2	Exporting playlists and files	10
2.1	Copy Playlist file	10
2.2	Export and copy tracks to	10
2.3	Export and convert tracks to	10
2.4	Export or import Auto-playlist files	10
3	Additional tools	11
3.1	Selected playlist contextual menu	11

3.1.1	Force relative paths	11
3.2	List menu	11
3.2.1	Dead items	11
3.2.2	External items	11
3.2.3	Duplicated items	11
3.2.4	Mixed absolute and relative paths	12
4	Shortcuts	13
II	UI	14
5	Features	14
6	List view	15
6.1	Category filtering -permanent-	15
6.2	Tag filtering -temporal-	15
6.3	Sorting	15
6.4	Tooltip	16
7	Customization	17
7.1	Custom color	17
7.2	Others	17
III	Other scripts integration	18
IV	Playlist formats	19
8	.m3u & .m3u8	19
8.1	Extended M3U	20

9 .pls	21
10 .fpl	22
11 Auto-Playlists	24
 V FAQ	 25
 VI Tips	 26
12 Sharing	26
13 Multiple views	26
14 Tag automation	26
15 Pools	27
16 Portable 'plug&play' installation	27

Part I

Features

1 Managing Playlist files and Auto-playlists

Using the Playlist Manager, playlists are virtual items also linked to a physical file in the preferred format (.m3u8, .m3u, .pls or .fpl).

Auto-playlists, due to its nature (its content change according to the library they reside in), are saved into json format in a file named accordingly to the folder tracked by the Playlist Manager¹.

.fpl playlists, similarly to Auto-playlists, are read only files... due to its closed source nature they are locked for further editing by default and metadata² is saved into the json file described previously.

In resume, writable formats are those that may be freely edited (.m3u8, .m3u, .pls) and readable formats are those that may be tracked by the manager with at least a minimum set of features³ and the capability to load their tracks within foobar (.m3u8, .m3u, .pls, .fpl or .json).

1.1 Auto-playlists

Contains all functionality on Auto-playlist Manager by marc2003 plus more.

- Create, rename, delete Auto-playlists.
- Edit query, sort pattern and sort forcing.
- Adds tooltip info, UI features, filters, etc.
- Number of tracks output is updated at foobar startup. (and/or 'at manual refresh')
- Queries and sort patterns are checked for validity before using them, instead of crashing.
- Import playlists from Auto-playlist Manager by marc2003[2.4].

1.2 How paths are written: absolute and relative paths

In writable playlists formats, tracks may be written as absolute or relative paths (considering the root the folder where the playlist resides in). For ex:

¹For ex. if the panel is set to track 'H: My Music

Playlists', then the playlist json file (at foobar profile folder) will be named 'playlistManager_Playlists.json'.

²Category, tags, lock status, etc.

³Metadata editing and conversion to a writable format.

Absolute path:

D:\Music\Big Retro Hits 90s\007. Chateau Pop - Maniac.mp3

Relative path on current root (D:\):

.\Music\Big Retro Hits 90s\007. Chateau Pop - Maniac.mp3

Relative path one level up from root (D:\Playlists\):

..\Music\Big Retro Hits 90s\007. Chateau Pop - Maniac.mp3

Relative paths may be enabled changing the related config on header menu⁴. Note enabling this feature is not enough condition per se, since the tracks must reside in the same drive disk to have relative paths working. Whenever relative paths can not be set, absolute paths are used as fallback.

1.3 Playlist loading

Foobar2000 loads playlists pretty fast thanks to using a binary playlist format (.fpl) instead of looking for the physical track files. The binary format stores all the relevant metadata needed to then display the tracks within foobar2000.

On the contrary, loading any of the writable format playlists in native Foobar2000 is really slow. The physical files are loaded one by one and then their metadata retrieved... that process is done asynchronously and can easily take minutes as soon as a playlist has more than a hundred of tracks⁵.

The manager uses those writable formats to create clones of the playlists within foobar but they are loaded as fast as the native binary format (.fpl) by finding matches on library for every track. Since playlists are supposed to be pointing to items already on Foobar2000's library on most cases, caching the paths of every item on library greatly speeds up the process⁶.

1.4 Autosaving and Autoloading

1.4.1 Autosaving

The first refers to the capability of duplicating any change made within a foobar loaded playlists (usually those on the playlist tabs) to their physical file counterpart. Obviously, only those playlists with a "clone" in the Playlist Manager panel will be tracked for changes⁷. This may be configured or completely disabled (to only reflect changes manually).

Since the only way to assign a playlist file to a Foobar2000 playlist is forcing both to have the same name, a new problem appears... what about duplicates?

⁴Save paths relative to folder...\Yes, relative to playlists folder

⁵Note subsequent loading of the same playlist is much faster since those items have been already cached.

⁶This erratic behaviour has been already reported at Foobar2000 support forums but so far there has been zero interest on implementing proper non-binary playlist loading in native Foobar2000.

⁷In other words, it may be possible to have both tracked and non tracked playlists within foobar.

By default the Playlist Manager will not allow you to have 2 playlists with the same name if there is already a playlist file named equal to them. i.e. no duplicates allowed. Note there is not a UUID associated to every playlists to work with, so in fact the only thing that may be used as UUIDs are the names. There are multiple configurable UUIDs that can be set instead of the plain name that can be used to allow some kind of 'duplicates' or to differentiate tracked from non tracked playlists.

1.4.2 Autoloading

The former refers to the capability of automatically tracking any playlist file within the tracked folder. i.e. any change made to that folder⁸ is reflected on real time on Foobar200. This may be configured or completely disabled (to only reflect changes manually or on startup).

1.5 Playlist binding

Bind playlist to files: - Tracks playlists for changes and update bound files. - Auto-saving (configurable). - Deleting the file also ask to delete the bound playlist. - Renaming the files also renames the bound playlist. - Show bound playlist (becomes active playlist).

1.5.1 Deleting and restoring files

RecycleBin: deleting and restoring. - Uses timestamps to uniquely identify files: no collisions with other files within the RecycleBin. A backup of the previous playlist json file is created every time the panel is loaded. Old backups are sent to recycle bin. Properties descriptions change according to things set on the panel, not just the values. i.e. if you change the sort method, then the description reflects the associated states dynamically.

1.6 Playlist metadata

1.6.1 Lock state

Playlist files may be locked or not. Locked files are considered read only for all purposes and therefore never rewritten⁹. Other features:

- Automatically locking of native foobar playlists files (.fpl)¹⁰.
- When locked, playlists can not be updated nor edited but can be deleted.

⁸Like adding or removing playlist files.

⁹This may be used to not update a playlist file on purpose while manipulating it within Foobar2000 until all is done or to avoid changing the playlist format.

¹⁰Configurable at properties panel only.

- Filename can be changed, but not playlist name (inside the file). This allows to set different playlist and file names if required.
- List view allows to show only or filter locked playlist[6].

1.6.2 [Playlist] Tags

Playlists may have multiple tags, i.e. keywords for informative purposes. There are some special purpose tags which are associated to some actions performed by the manager automatically as soon as it loads a playlist with such keywords.

- Playlists may be tagged with 'bAutoLoad', 'bAutoLock' or a custom set of tags (for arbitrary purposes).
- Auto-Functions: automatically applies some actions whenever a playlist is loaded on the panel according to the tags present on it.
- 'bAutoLoad' makes the playlist to be loaded within foobar automatically (on the UI). Meant to be used on remote servers with online controllers.
- 'bAutoLock' locks the playlist as soon as it's loaded on the panel.
- List view allows to show only or filter specific tags[6]. Note tag filtering gets reset on start-up¹¹.

1.6.3 Track tags

Playlists may have multiple track tags, i.e. tag presets which are meant to be applied to the tracks within the playlist. Note this has nothing to do with the [Playlist] Tags which are just keywords.

Track Auto-tagging is a feature which must be explicitly enabled on the header menu[4] to work; adding track tags to a playlist will not apply it until it's enabled.

- Can be configured separately for standard playlists, Auto-playlists, locked playlists and individual playlists.
- Standard playlists may be used to easily tag your library just by sending them to the right playlist (which don't need to be loaded at all).
- Auto-playlists Auto-tagging allows to automatically (and periodically) apply some tagging logic to the current library following some condition.
- Multiple playlists may be used as pools, using a final Autoplaylist which checks for an specific added tag (by other playlists -aka pools-).
- Allows multiple inputs:

¹¹In opposition to category filtering, and that's why tags are not meant to be used as a way to categorize playlists but only for informative purposes.

- Foobar2000 Title Format expressions (or %tags%)¹².
- JavaScript functions (defined at 'helpers\helpers_xxx_utils.js').
- Value (string or number).

1.6.4 Category

1.6.5 UUID

UUIDs: added to the name, so they are separated from non tracked playlist by name when loaded in foobar. Some also allow some level of names duplication.

- Invisible Unicode chars plus (*)¹³
- (a-f)¹⁴
- (*)¹⁵
- Or none¹⁶.

¹²https://wiki.hydrogenaud.io/index.php?title=Foobar2000:Titleformat_Reference

¹³Allows the highest degree of name duplication. For ex. multiple playlists with name 'Summer' would have different invisible UUID's... Experimental feature.

¹⁴Allows some degree of name duplication. The UUID will be visible along the name, thus being less useful than the Unicode version... but more stable.

¹⁵Duplication is not allowed but serves as a indicator of playlist being tracked by the manager.

¹⁶The only way to know if the playlist is tracked by the manager is by looking at the manager panel and checking the loaded or now playing indicators[5].

2 Exporting playlists and files

2.1 Copy Playlist file

2.2 Export and copy tracks to

2.3 Export and convert tracks to

2.4 Export or import Auto-playlist files

3 Additional tools

3.1 Selected playlist contextual menu

3.1.1 Force relative paths

3.2 List menu

The following set of tools can be found on the list menu [4].

3.2.1 Dead items

The manager can check all the playlists currently tracked for dead items on them, whether the tracks are on current Foobar2000's library or not. Dead items are considered files which don't exist at their path, no matter if it's a relative or absolute path.

The tool will span a popup reporting a list of playlists with dead items (but will not list the items their-selves). To find such items or replace them with items from current library use a dedicated tool like the one found at Playlist-Tools-SMP¹⁷.

3.2.2 External items

The manager can check all the playlists currently tracked for external items on them, i.e. tracks not present on Foobar2000's library but which exists at their path. Note external items are technically not dead items, since they do exist outside Foobar2000 database.

3.2.3 Duplicated items

The manager can check all the playlists currently tracked for duplicated items on them, i.e. two tracks with the same path. Note there is a limit though, if absolute and relative paths are intentionally mixed and 2 paths point to the same physical file, they will not be considered duplicated. For ex¹⁸:

```
[...]  
#EXTINF:141,Jeffery Mykals - Gyal Bad  
..\music\Jeffery Mykals - Gyal Bad.mp3 *  
#EXTINF:141,Jeffery Mykals - Gyal Bad  
D:\music\Jeffery Mykals - Gyal Bad.mp3  
[...]  
#EXTINF:141,Jeffery Mykals - Gyal Bad  
..\music\Jeffery Mykals - Gyal Bad.mp3 *
```

¹⁷Look for 'Playlist Revive'.

¹⁸Only the tracks with an *will be considered duplicates.

The tool will span a popup reporting a list of playlists with duplicated items (but will not list the items their-selves). To find such items or remove them use a dedicated tool like the one found at Playlist-Tools-SMP¹⁹

3.2.4 Mixed absolute and relative paths

The manager can check all the playlists currently tracked to ensure there are no playlist files with both absolute and relative paths at the same time. Such files are probably an error, whether intentional or not, and should be fixed in most cases. For ex:

```
[...]
#EXTINF:141,Jeffery Mykals - Gyal Bad
..\music\Jeffery Mykals - Gyal Bad.mp3
#EXTINF:271,Jeffery Mykals - Chico & Voicemail
D:\music\Jeffery Mykals - Chico & Voicemail.mp3
```

The tool will span a popup reporting a list of playlists with such 'problem'. If all items exist (no dead items), as soon as the playlist is rewritten by the manager it will be automatically fixed²⁰.

¹⁹Look for 'Duplicates and tag filtering'. Foobar2000's main menu 'Edit/Remove duplicates', after loading the playlist, can also be used.

²⁰It will use the current path configuration, thus converting all paths to absolute or relative paths. Rewrite will trigger, after loading the playlist within Foobar2000, on auto-update or via manual update on selected playlist menu [4].

4 Shortcuts

All features can be used using the following mouse gestures:

- **Left Click**: Selected playlist contextual menu.
- **Right Click (*list*)**: List menu; new playlist and other playlist tools.
- **Right Click (*header*)**: Manager configuration.
- **Double Click (*list*)**: Load selected playlist / Make bound playlist active.
- **Double Click (*header*)**: Categories cycling.
- **Ctrl + Left Click**: Load selected playlist / Make bound playlist. active
- **Shift + Left Click**: Send current selection to selected playlist.
- **Ctrl + Shift + Left Click**: Recycle selected playlist.

Part II

UI

5 Features

- UI re-sizable on the fly. i.e. it will adjust layout to panel size.
- Selection indicators.
- Now playing playlist indicator: <
- Loaded playlist indicator: —
- Empty / not empty playlist indicators. To be used as fallback when size is not shown.
- Font Size (configurable).
- Separators between different names/categories (configurable).
- Colors for different playlists types, status, text, background and selection (configurable).

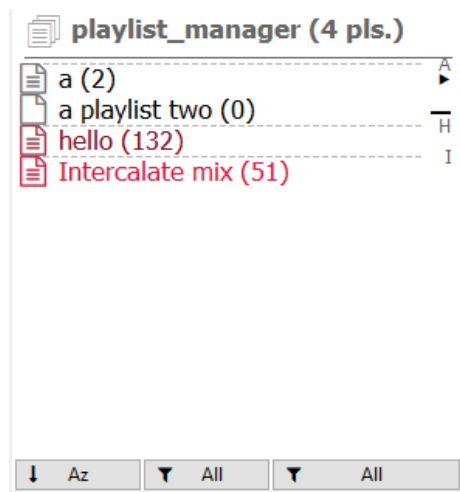


Figure 1: Now playing ('a'), loaded ('a playlist two') and two non loaded playlist.

6 List view

The main panel view features a simple listing of all currently tracked playlists (whether they are physical or virtual -json- files). This view can be filtered according to these parameters:

- Show all Only Auto-playlists Only standard playlists
- Show all Not locked Locked
- By selectable categories (virtual folders)
- By selectable tags

The current view is always maintained on subsequent foobar startups unless changed. Note tag filtering is always reset since it's meant for informative purposes²¹, not for playlist categorization.

6.1 Category filtering -permanent-

Playlists may be filtered by category (like virtual folders), and multiple individual selections are allowed via menu (for ex. to display all but one specific category). When lists are being filtered by category, an indicator is shown in the header text. The selected filtering is maintained on subsequent startups.

An additional way to filter by category is cycling the current category being shown (one by one) [4]. This works in conjunction with playlists being allowed to only have one category at the same time, so no playlist will be shown more than one time.

6.2 Tag filtering -temporal-

Playlists may also be filtered temporarily by tags; it gets reset on subsequent startups. Therefore tags should only be used for informative purposes but not for categorization. Note a playlist may have multiple tags at the same time.

6.3 Sorting

List view may be sorted by name, category or size. By default playlists are sorted by name, using natural sorting (Az). Ordering may be changed using the appropriate buttons. The selected sorting mode is maintained on subsequent startups.

Additionally, name / category separators may be shown on the UI to easily identify where the next char begins (123, A, B, ...).

²¹In fact is also used for some complex playlist automatic actions.

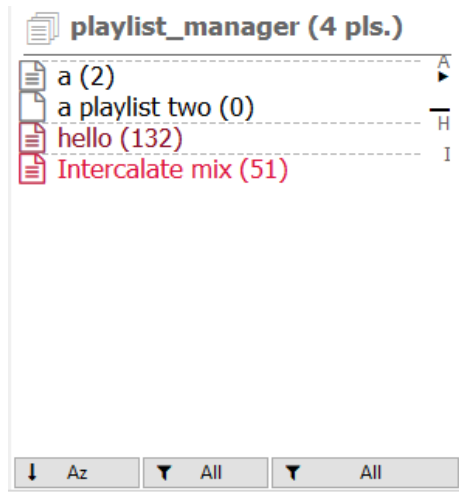


Figure 2: 'A', 'H' and 'I' separators are shown for names.

6.4 Tooltip

Tooltips show different playlist info:

- Header:

- * Absolute path of currently tracked playlist folder.
- * Filters used on current view.
- * Categories shown.
- * Playlist and Auto-playlists shown on current view.
- * Shortcuts info (configurable).

- Playlists:

- * Playlist type: Playlist / Auto-Playlist
- * Name plus UUID.
- * Playlist size (tracks). Configurable for Auto-playlists (output by query).
- * File name with extension.
- * Status (lock).
- * Category / Tag(s).
- * Track Tag(s).
- * Query. Sort Pattern. (only Auto-playlists)
- * Shortcuts info (configurable).

7 Customization

7.1 Custom color

- Background: panel background.
- Standard text: for standard playlists and header.
- Auto-playlists: different color for Auto-playlists.
- Locked playlists: different color for non editable (locked) playlists.
- Current selection: currently selected playlist.

7.2 Others

- Tooltip: Shortcuts info [4] can be shown or hidden.
- Separators: Name or Category separators may be shown when sorting by those values.
- Playlist size: Track count may be shown on parenthesis along playlist names. An additional configuration allows to refresh Auto-playlists on startup.
- Font size: applies to all text within the panel.

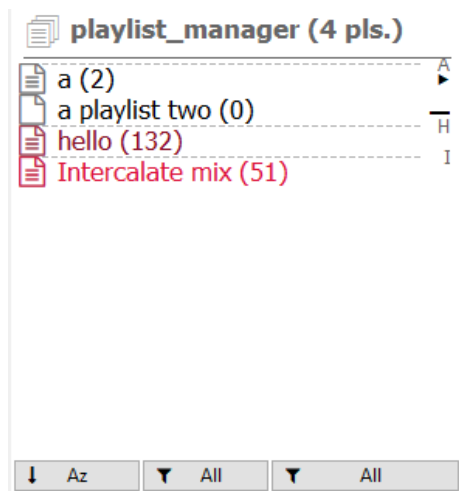


Figure 3: Size and separators are shown for playlists.

Part III

Other scripts integration

Other scripts integration:

- Playlist-Tools-SMP:

- * **Random Pools:** Pools may use tracks from playlists files tracked by the manager, not requiring to have playlists loaded within foobar. i.e. Random Pools component-like playlist creation, using not only queries as sources, but also other playlists or playlists files.
- * **Playlist Revive:** Finds and replaces dead items on loaded playlists or selection. Meant to be used along dead items checks on playlist files [3.2.1]. First check all playlist files, then load those with dead items and use Playlist Revive.
- * **Duplicates and tag filtering:** The manager allows to report playlist with duplicated items, but it's limited to entries with same path. This tool expands Foobar2000 native functionality of removing duplicates, allowing to find duplicates by tags (for ex. any track with same Title - Artist).
- * **Import track list:** Takes a plain text list of tracks (for ex. Title - Artist) and finds matches on library to create a playlist. Meant to be used for playlist importing when the track list does not follow an standard format or there are no paths provided²². Instead of sharing a list of files, list of tracks may be used which work universally no matter your configuration. Non found items are simply discarded.

²²Technically that is not a playlist. But note playlists with relative paths may easily be considered a track list as long as you discard the '.\' part. In other words, a plain-text list can be retrieved from playlists in many cases.

Part IV

Playlist formats

- Writable formats: .m3u, .m3u8 and .pls.
- Readable only formats²³: .fpl and Auto-Playlists.

In general, writable formats work with more features than their readable only counterparts²⁴. This is specially true for .fpl playlists, which are pretty limited on all aspects²⁵. Therefore, the use of .m3u8 playlists is preferred over other formats (.fpl or .pls) whenever it is possible.

8 .m3u & .m3u8

M3U (MP3 URL) is a file format for a multimedia playlist. Although originally designed for audio files, such as Flac, it is commonly used to point media players to audio and video sources, including online sources, without distinction. There is no formal specification for the M3U format, it is a de facto standard.

An M3U file is a plain text file that specifies the locations of one or more media files. The file is saved with the 'm3u' filename extension if the text is encoded in the local system's default non-Unicode encoding (e.g., a Windows code-page), or with the 'm3u8' extension if the text is UTF-8 encoded. Within the manager, for all purposes, files generated by it are equivalent since '.m3u' files are also UTF-8 encoded²⁶.

Each entry carries one specification. The specification can be any one of the following:

- An absolute local path-name; e.g., C:\My Music\Listen.mp3
- A local path-name relative to the M3U file location; e.g., Listen.mp3
- A URL; e.g., http://www.example.com:8000/Listen.mp3

Position	Description	Entries
Header:	Required if using Extended M3U.	#EXTM3U,#EXTENC[,...]
Track(s)	Track entries, arbitrary number allowed.	[#EXTINF,]file path or url
...		

Table 1: M3U structure.

²³For all purposes, locked playlists may be considered into this category, no matter their extension.

²⁴There are some more exceptions to this rule. For example, .m3u8 playlists have more features than .pls ones.

²⁵Neither exporting [2], nor additional tools [3] work with them. And obviously, since they are a readable only format, no new tracks may be added to them using the manager.

²⁶Although this 'breaks the standard', it's indicated with the appropriate extended M3U directive so it should be totally safe. Also note the 2015 proposal for HTTP Live Streaming follows the same convention.

8.1 Extended M3U

The M3U file can also include comments, prefaced by the '#' character. In extended M3U, '#' also introduces extended M3U directives which are terminated by a colon ':' if they support parameters.

Directive	Description	Example
#EXTM3U	File header, first line	#EXTM3U
#EXTENC:	Text encoding, second line	#EXTENC:UTF-8
#PLAYLIST:	Playlist display title	#PLAYLIST:My Playlist
#EXTINF:	Track information: seconds and title	#EXTINF:256,Chateau Pop - Maniac

Table 2: Standard M3U extensions.

Since arbitrary comments can be prefaced by '#' and custom directives are allowed, the manager includes its own set of directives to support additional playlist metadata [1.6]:

Directive	Description	Example
#UUID	Playlist UUID [1.6.5]	#UUID: (*)
#LOCKED:	Lock state	#LOCKED:false
#CATEGORY:	Playlist category	#CATEGORY:Summer
#TAGS:	Playlist tags (sep by ';')	#TAGS:Celtic;Chill
#TRACKTAGS:	Tags to apply tracks (json) [1.6.3]	#TRACKTAGS:["Mood": "Chill"]
#PLAYLISTSIZE:	Playlist size (# of tracks)	#PLAYLISTSIZE:2

Table 3: Additional M3U extensions for playlist metadata support.

The use of Extender M3U along the additional custom directives allows the manager to make use of all features without restrictions. For ex. playlists may have UUIDs independently of their playlist name, categories may be used to filter the list, etc.

An example of a full .m3u8 playlist with relative paths²⁷ can be found below:

```
#EXTM3U
#EXTENC:UTF-8
#PLAYLIST:My playlist
#UUID: (*)
#LOCKED:false
#CATEGORY:Summer
#TAGS:Celtic;Chill
#TRACKTAGS:["Mood": "Chill"]
#PLAYLISTSIZE:2
#EXTINF:256,Chateau Pop - Maniac
.\Music\Big Retro Hits 90s\007. Chateau Pop - Maniac.mp3
#EXTINF:259,Jaki Graham - Round And Around
.\Music\Big Retro Hits 90s\004. Jaki Graham - Round And Around.mp3
```

²⁷Playlist file would be at current folder and tracks within 'music' subfolder.

9 .pls

PLS is a file format for a multimedia playlist. Used with audio and video sources, including online sources, without distinction.

PLS is a more expressive playlist format than the basic M3U playlist, as it can store information on the song title and length (supported in extended M3U only)²⁸.

The format is case-sensitive and essentially that of an INI file structured as follows:

Position	Description	Entries
Header:	Always required. 'As is'	[playlist]
Track(s)	File entries, arbitrary number allowed. X equals entry number	FileX[,TitleX,LengthX]
...		
Footer	Always required.	NumberOfEntries,Version

Table 4: PLS structure.

Key-value pairs are separated by '=':

Entry	Description	Example
[playlist]	Always required	[playlist]
FileX	Location of media file/stream.	File1=http://stream2.streamq.net:8020/
TitleX	Track title (optional)	Title1=My radio station
LengthX	Seconds (-1 equals indefinite) (optional)	Length1=-1
NumberOfEntries	Playlist size (# of tracks). Required	NumberOfEntries=3
Version	only a value of 2 is valid. Required	Version=2

Table 5: PLS entries.

Since the pls format follows a pretty strict format, additional metadata like categories or UUID's can not be used with them (playlist name is the same than the filename). Switch to another format to make use of those features.

An example of a full .pls playlist with relative paths²⁹ can be found below:

```
[playlist]
File1=.\foobar2000\Big Retro Hits 90s\004. Jaki Graham - Round And Around.mp3
Title1=Round And Around
Length1=259

File2=.\foobar2000\Big Retro Hits 90s\007. Chateau Pop - Maniac.mp3
Title2=Maniac
Length2=256

NumberOfEntries=2
Version=2
```

²⁸This is only true for basic M3U playlists usually found on the net. Within the manager context, M3U playlists are always richer in metadata and features since they make use of extended M3U and the additional custom directives.

²⁹Playlist file would be at current folder and tracks within 'music' subfolder.

10 .fpl

FPL is a file format for a multimedia playlist by Foobar2000. Used with audio and video sources, including online sources, without distinction.

It's a closed source format whose structure has not been shared publicly, although it's known it uses a binary format to store the metadata of the tracks included to greatly speed up its loading time within the program³⁰.

Although it looks as an improvement over plain text playlist formats, the 'closed source & binary' format requirements no longer holds true to offer short loading times. Plain text playlist formats may be used perfectly fine, as long as the files are matched with those on the library, without speed penalties³¹. This is the behavior followed by manager and it has been already reported to Foobar2000 developers³² to properly implement playlist loading if desired. Loading speed penalties only happen when some items are not on library (whether they are external items or dead items) [3].

To allow additional metadata for .fpl playlists, considering the files are non-editable, an external json³³ file is used. The same applies to Auto-Playlists. The following keys-values pairs are used:

Entry	Description	Example
id	UUID	"id": " (*)"
name	Playlist name	"name": "example"
nameId	Playlist display name	"nameId": "example (*)"
extension	Playlist file extension	"extension": ".fpl"
path	Playlist file path	"path": ".\profile\playlist_manager\example.fpl"
size	Playlist size (# of tracks)	"size": 2
fileSize	Playlist file path	"fileSize": 20739
isLocked	Lock status	"isLocked": true
isAutoPlaylist	Is an Auto-Playlist?	"isAutoPlaylist": false
query	Auto-Playlist query	"query": ""
sort	Auto-Playlist sort(optional)	"sort": ""
bSortForced	Auto-Playlist sort forced?	"bSortForced": false
category	Playlist category	"category": "Summer"
tags	Playlist tags (sep by ';')	"tags": ["bAutoLoad", "bAutoLock"]
trackTags	Tags to apply tracks (json)	"trackTags": [{"Mood": "Chill"}]

Table 6: FPL (and Auto-Playlist) json format.

Note all Auto-playlist related metadata is empty, the path points to the physical .fpl file and its file-size is cached³⁴. Apart from those differences, it can be easily checked that all playlist metadata is present (the same it was in M3U format). In fact, all playlists are converted -for internal use- to this format within the code.

³⁰Instead of loading the tracks and retrieving their metadata from the physical files.

³¹Which is one of the most common use-case of playlists within Foobar2000.

³²Check Why m3u8 loading is so slow on hydrogenaud.io-

³³File structure is not specified since JSON is a well known standard. Check <https://en.wikipedia.org/wiki/JSON> for more info.

³⁴For Auto-playlists, it would be the contrary. No physical file is associated and the query and sorting is used instead... but -essentially- they use the same format.

An example of a full .fpl playlist associated json file can be found below. In real files, every playlist would be concatenated to the same file, thus having multiple {playlists objects}, separated by comma (','), between the brackets [{...}, {...}]:

```
[
  {
    "id": " (*)",
    "name": "Example",
    "nameId": "Example (*)",
    "extension": ".fpl",
    "path": ".\\profile\\playlist_manager\\Intercalate mix.fpl",
    "size": 51,
    "fileSize": 20739,
    "isLocked": true,
    "isAutoPlaylist": false,
    "query": "",
    "sort": "",
    "bSortForced": false,
    "category": "Summer",
    "tags": ["bAutoLoad", "bAutoLock"],
    "trackTags": ["Mood": "Chill"]
  }
]
```

11 Auto-Playlists

Structure and format is exactly the same than .fpl use-case, so check it for reference [10]. Specifics for Auto-Playlists are listed below:

Entry	Description	Example
extension	Playlist file extension	"extension": ""
path	Playlist file path	"path": ""
fileSize	Playlist file path	"fileSize": 0
isAutoPlaylist	Is an Auto-Playlist?	"isAutoPlaylist": true
query	Auto-Playlist query	"query": "ALL"
sort	Auto-Playlist sort (optional)	"sort": "SORT DESCENDING BY TITLE"
bSortForced	Auto-Playlist sort forced?	"bSortForced": true

Table 7: Auto-Playlist json format changes.

Note extension and path are empty, since there is no physical file associated. File-size is therefore equal to zero. 'isAutoPlaylist' boolean is true and the query related value must be filled. Sorting is optional.

An example of a full Auto-playlist associated json file can be found below. In real files, Auto-Playlists are mixed with .fpl playlists... the only distinction being the 'isAutoPlaylist' boolean value:

```
[
  {
    "id": "",
    "name": "Entire Library",
    "nameId": "Entire Library",
    "extension": "",
    "path": "",
    "size": 132,
    "fileSize": 0,
    "isLocked": false,
    "isAutoPlaylist": true,
    "query": "ALL",
    "sort": "SORT DESCENDING BY TITLE",
    "bSortForced": true,
    "category": "Summer",
    "tags": [],
    "trackTags": []
  }
]
```


Part V

FAQ

- **Writing playlists to files fails due to permissions problems?** Use something like this:

```
attrib -r D:\YOUR_PLAYLIST_PATH\* /D /S
```

- **How to use native foobar playlists (.fpl) without changing their format?** .fpl playlists are locked by default, so they will never be auto-saved (and thus reformatted) without user intervention. Just save all your foobar playlists on the tracked folder and load them when needed using the manager. Whenever you make a change on any of them, re-save it manually using main menu (File/Save playlist...). This way the native format is maintained, while some neat features are still available for use (not cluttering the UI with all playlists on tabs, categories, tags, etc.).
- **To create/track a folder in the same folder Foobar2000 resides in, relative paths may be used (.\playlist_manager\server\):** Note this will allow the manager to work properly on portable/network installations where the drive letter or absolute path changes.
- **Native playlists are too limited in features?** The use of .m3u8 playlists is preferred since it allows the full use of all features. This is by design, and nothing can be done unless the format becomes open source or Spider Monkey Panel supports directly editing/saving them. Nothing is lost using .m3u8 playlists, since they load as fast as native playlists when using the manager.
- **Playlist metadata is lost on format switch:** Since only .m3u8 supports the full set of metadata and features, converting those playlists to .pls necessarily implies discarding of not supported metadata. Converting those playlists back to .m3u8 format will not restore it once is lost!

Part VI

Tips

12 Sharing

- As noted on [1], **Autoplaylist's json file is saved using the tracked playlist folder name**. Instead of using an arbitrary UUID to avoid collisions between multiple panels, this can be used to **easily share playlists between different foobar instances**. Just create SymLinks³⁵ or use some cloud syncing tool (like Dropbox) to easily share the same playlists when tracking the same folder on different foobar instances or panels. Note only the folder name is used, so it would work even on shared network folders.

13 Multiple views

- Following the same principle, it's possible to have **multiple panels tracking the same folder in the same foobar instance**. It may be used to have different filters enabled at the same time. For ex. one view for Auto-playlists only and another for .m3u8 playlists.
- **Categories may be used as virtual folders, even if all playlists are in the same physical folder**. Note every playlist can only have one category at the same time, so cycling the categories allows to easily see different 'virtual sub-folders'. Double clicking on the header allows to easily do that. Alternatively, the multiple panel tracking the same folder trick can be used to show 2 views of the same physical folder but with different categories filters, again working as sub-folders.

14 Tag automation

- **Tracks are never tagged twice using the Track tags feature**, so there is no need to check if any of them has been already tagged or not before.
- **Using Auto-playlists along the Track tags feature, items on library can be auto-tagged on startup** according to some conditions without any user input. For ex. tagging all tracks with 'Rock' and 'Acoustic' as genre and BPM lower than 90 with an specific mood or occasion tag. Just set it and forget, it will be done on every new track added on library as soon as it matches the conditions.
- **Using standard playlist along the Track tags feature is an easy way to manually tag tracks on batches** while listening to them. For ex. to add tags like 'Instrumental' or 'Acoustic', 2 playlists may be used for auto-tagging with these conditions and just listen to the

³⁵Symbolic links are virtual links created by the OS which may be used to have multiple virtual files pointing to the same physical file. It's similar to a shortcut, although the extension doesn't change in this case... and the file properties are those of the original one.

music; as soon as one track must be tagged it takes a second to send the current track to any of the 2 playlist (Shift + L. Click) to tag it.

- **Use category filtering to have a virtual folder of 'tagging playlists' which would only be shown when needed**, thus not convoluting the UI the rest of the time. This has some improvements and limits compared to the use of custom buttons and Mass-tagger presets.

15 Pools

- **Native Foobar2000 limits the sourcing of Auto-playlists to the library, not allowing to create playlists from playlists...** although this can be simulated copying the original query and using in the new playlist, it becomes easily convoluted as soon as you do it 2 or 3 times. **As an alternative, Track tags feature may be used.** If you set any playlist to **automatically tag its tracks with an specific tag**, for ex. playlist = 'Summer', you can **then create another Autoplaylist with a query for that tag 'PLAYLIST IS Summer'**.
- The same trick can be used to **merge multiple playlist sources into one** ('PLAYLIST IS Summer OR PLAYLIST IS Chill OR PLAYLIST IS BEST'), **effectively using other playlists as pools**. Note 'sources' are not limited to Auto-playlists and that's the real power of this solution, both standard playlists and Auto-playlists which automatically tag tracks this way may be used.
- Alternatively, **Playlist-Tools-SMP [III] allows to directly use playlists as sources for pools without requiring the use of intermediate tags.**
- Used along Playlist-Tools-SMP [III], the emulation of pools can be greatly enhanced with the Remove Duplicates or X random selection features to create playlists in a matter of seconds from your pre-selected set of tracks.
- More complex workflows may be done by mixing these tips along the Pools feature of Playlist-Tools-SMP [III], using the already created 'pools' playlists within the playlist manager as sources.

16 Portable 'plug&play' installation

- **Real portable installations (i.e. on a external drive, network installations, etc.) may need to track playlist folders using their relative paths** instead of absolute paths to work properly... otherwise they will not found the tracked folder as soon as the drive letter changes. e.g. `.\profile\playlist_manager\server\`
- **Relative paths for files always are checked against foobar path**; this is true only at places like the properties panel, etc.³⁶ i.e. `'.\profile\playlist_manager\server\'` equals to `'D:\foobar2000\profile\playlist_manager\server\'`.

³⁶When checking tracks, their root is considered to be the playlist path.

- **Relative paths on playlists should be preferred...** this is specially a must when the music is stored along the disk Foobar2000 resides in. Otherwise the files would not be found as soon as the drive letter changes (see previous tip). This may greatly affect the speed of the loading playlist process or even make it fail.
- Use the '**Check playlists consistency...**' (right button menu) to ensure all is properly set (library, configuration, playlists, ...) **on portable installations**; also handy to ensure all playlists items are found and within the library.
- Once the panel is set properly, it just takes a matter of seconds to copy the entire manager 'as is' to other installations: the config files (FOOBAR_PROFILE_PATH\js_data*.json), properties panel (can be saved as json) and playlists folder (along its files) can be transferred without changes.