Playlist Manager

(aka Playlist-Manager-SMP)

regorxxx

September 8, 2021

Contents

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 [V] in a file named accordingly to the folder tracked by the Playlist Manager[1].

.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 metadata[2] 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 features[3] 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[4.1].

[1] For ex. if the panel is set to track 'H:\My Music\Playlists', then the playlist json file (at foobar profile folder) will be at '.\js_data\playlistManager_Playlists.json'.
[2] Category, tags, lock status, etc.
[3] Metadata editing and conversion to a writable format.

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:

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 menu[4]. 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 foobar200.

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 tracks[5].

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 process[6].

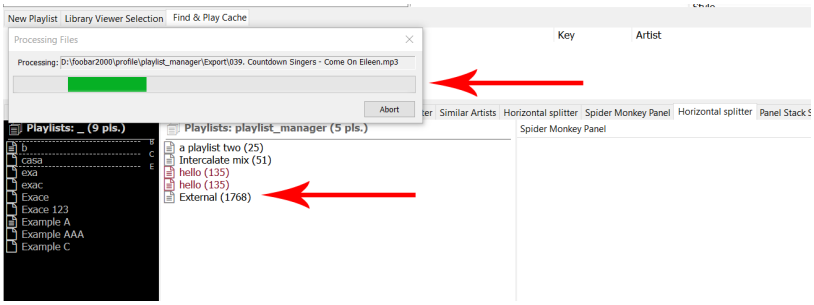

Figure 1: Async loading of a playlist file with tracks not present on library.

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

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

[6]This 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.

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 changes[7]. 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?

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 folder[8] 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

Binding is the action of associating a playlist within Foobar2000 and a physical file for syncing purposes. This is done by name, so a playlist named 'ABC' in the manager will be a mirror of a playlist named 'ABC' in Foobar2000 UI. Additional ways to handle playlist names can be set using UUIDs [11.5].

[7] In other words, it may be possible to have both tracked and non tracked playlists within foobar.
[8] Like adding or removing playlist files.

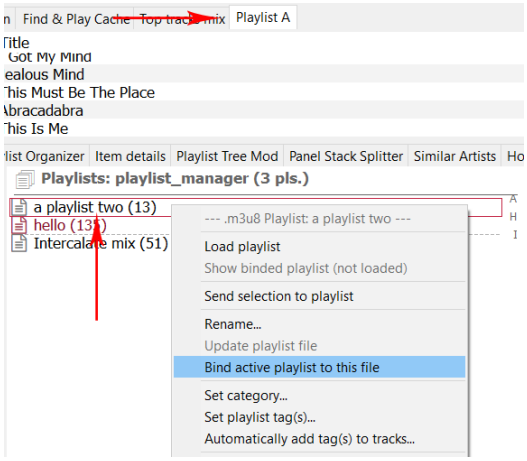

Figure 2: Bind selected playlist to active playlist.

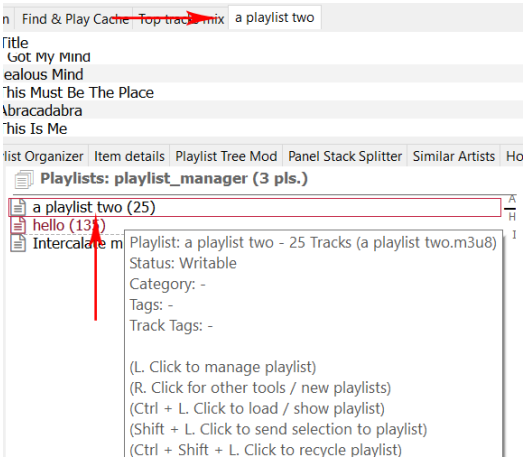

Figure 3: The active playlists is renamed and the playlist file updated with its contents.

Bound playlists are auto-saved if such feature is enabled, i.e. any change made to the Foobar2000 playlist will be automatically reflected in the physical file. Other restrictions may apply though[9]. Other manual actions include:

- Reload: reloads the playlist within Foobar2000, overwriting any non saved change.
- Update / Force update: saves any change to the physical file[10].
- Delete: deleting the file also asks to delete the bound playlist within Foobar2000.
- Rename: Renaming the file also renames the bound playlist[11].
- Show bound playlist: bound playlist within Foobar2000 becomes active playlist[12].

1.6 Deleting and restoring files

Playlist files deleted within the manager context are not permanently deleted but sent to the Recycle Bin. Timestamps are used to uniquely identify files; this is done to ensure no collisions with other files within the Recycle Bin. Manually deleting a playlist using the playlist contextual menu [7] allows restoring at a later point using the list contextual menu[13].

[9] Playlist may be locked for changes, a non writable format may be used, etc.
[10] The manual counterpart of auto-saving.
[11] This is a requisite, since the link is the name!
[12] Like the 'Show now playing' action, but instead it simply shows the selected playlist.
[13] Alternatively, the file may be found on the Recycle Bin... so it could be restored manually after stripping the timestamp.

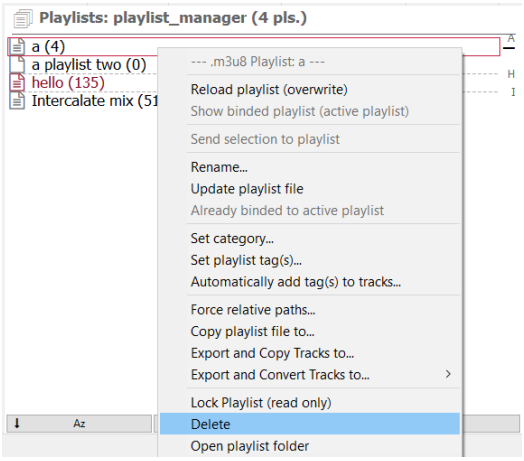

Figure 4: Delete selected playlist.

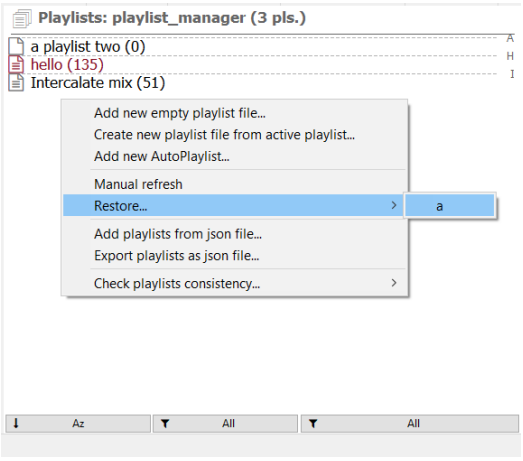

Figure 5: Restore deleted playlists.

Additionally, a backup of the Auto-playlist/fpl json database is created every time the panel is loaded and previous backups are sent to recycle bin[14].

[14]In other words, there is always 2 versions of the file. The current and the previous [start-up] one.

1.7 Locking files

Playlist may be locked to disable editing, overwriting the physical file or any change apart from unlocking or renaming it[15]. Locked playlists are also skipped on auto-saving.

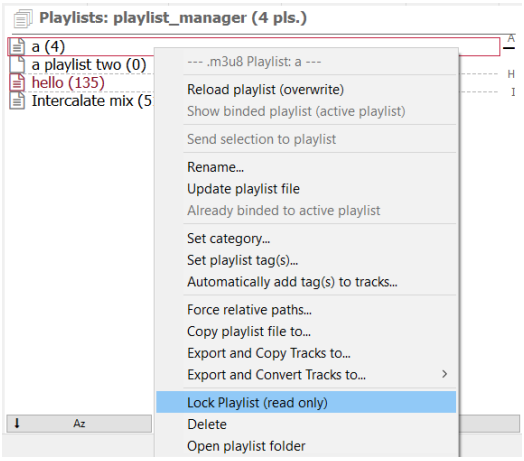

Figure 6: Lock selected playlist.

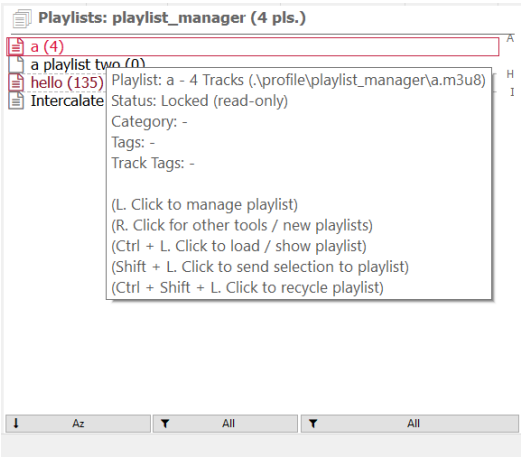

Figure 7: Locked playlist tooltip.

Note a loaded playlist within foobar may be edited even if its associated physical file is locked. This is done on purpose [16], to allow playlist edits while having the physical file locked and untouched. At any point the playlist may be unlocked and changes be saved or it may be directly forced to update the changes. Alternatively they may be discarded simply reloading the playlist file.

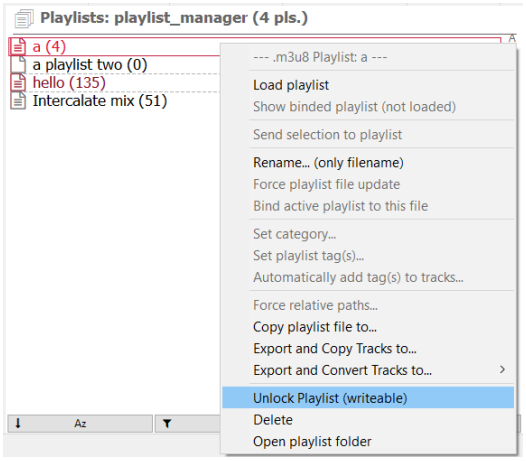

Figure 8: Unlock selected playlist.

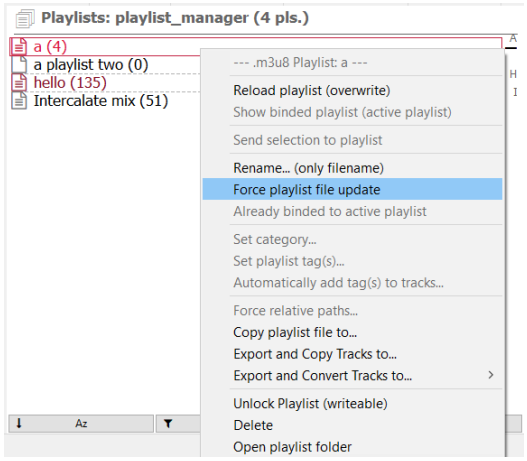

Figure 9: Force saving to locked playlist.

Native foobar playlists files (.fpl) are locked by default[17].

[15] Only the physical file is renamed in such case.

[16] Contrary to what other Foobar2000 playlist managers do, which lock/unlock the playlists within the program (sincere there is no physical files).

[17] Configurable at properties panel only.

2 Automatic playlist actions

Some reserved playlists tags names are used for special purposes by the manager to automatically perform some actions as soon as it loads a playlist with such keywords.

- '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.

The feature must be explicitly enabled on the header menu [7] to work; i.e. a playlist with such tags will not automatically perform any action until done so.

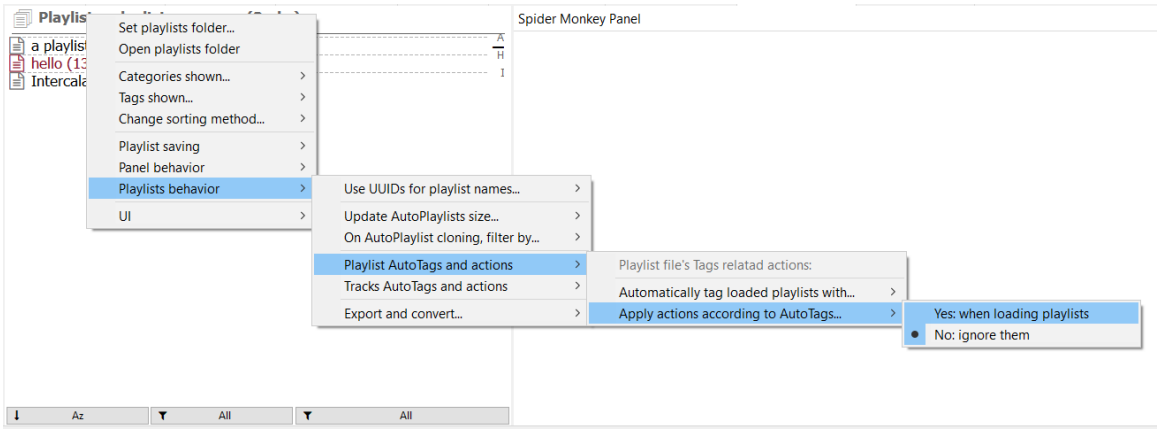

Figure 10: Enabling Automatic playlist actions on the header menu.

Additionally, tags may be added to playlists automatically as soon as they are loaded. This may be used to enforce an specific tag on all playlists tracked by the manager, no matter what it's set on the playlist file. Furthermore, used along the automatic playlist actions, it may be used to force loading or locking of all playlist tracked[18].

[18] It's disabled by default, so this allows both: to selectively apply actions to 'tagged' playlists or apply them to all.

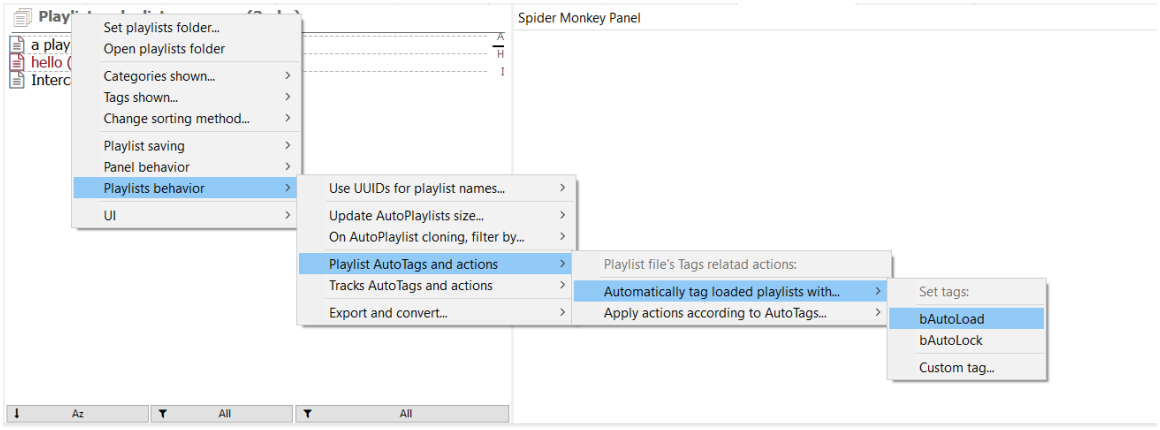

Figure 11: Setting tags to be added automatically to tracked playlists.

Obviously, nothing stops you to use it to simply tag playlists with a custom tag that has nothing to do with automatic actions. In any case, the tooltip shows the playlist tags (whether they are for informative purpose or used for actions):

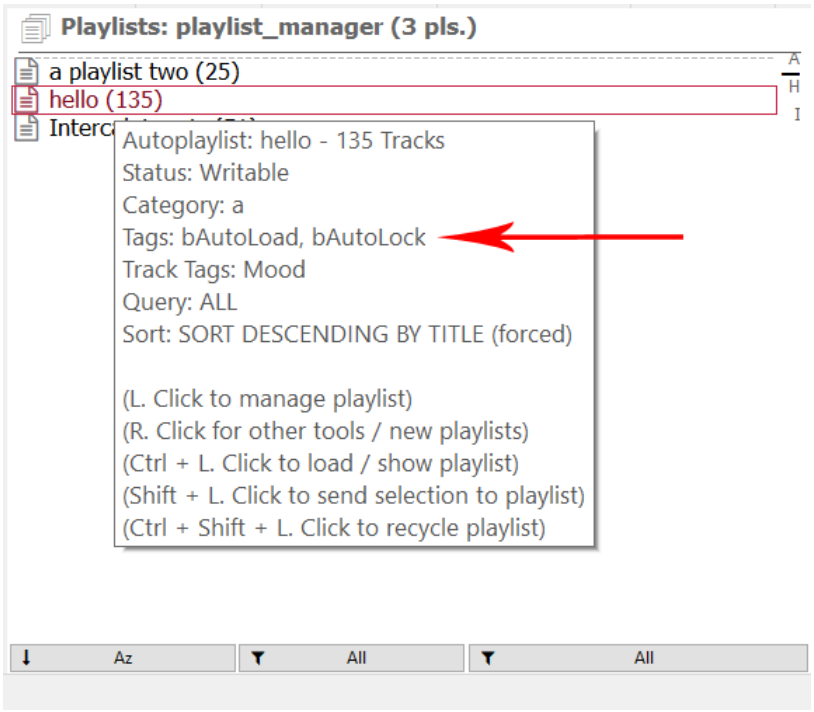

Figure 12: [Playlist] tags on tooltip.

3 Automatic track tagging

Playlists may be used to automatically tag tracks on demand (the moment you add a track) or on startup [11.3]. The conditions to tag tracks on a playlist are set using the contextual menu for the selected playlist [7] and must follow json format [V]. For example:

Example	Description of how tracks would be tagged
[{"rating":5}]	%rating% and a value of 5
[{"mood":"Chill"}]	%mood% and a value of 'Chill'
[{"year":"$year(%date%)"}]	%year% and a year value from the full date tag
[{"checked":"JS:todayDate)"}]	%checked% and a date value using JavaScript.

Table 1: Automatic track tagging examples.

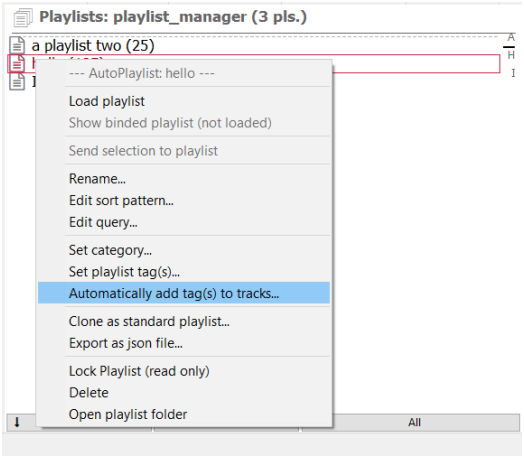

Figure 13: Add track tags to selected playlist.

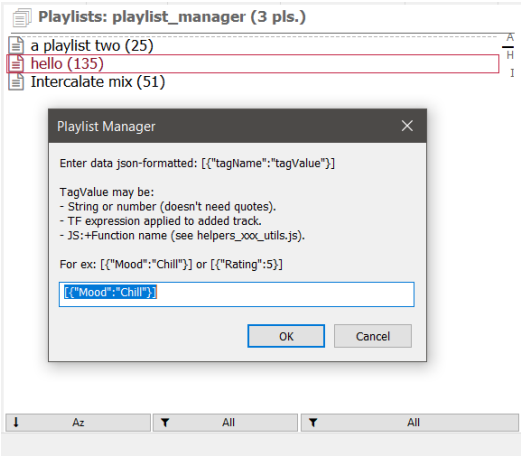

Figure 14: Track tags popup.

As noted, the use of arbitrary JavaScript functions is allowed, but they must be defined at a helper file '.\helpers\helpers_xxx_utils.js'. In this case, the function it simply returns the date Y-M-D-h-m-s in which was tagged. Users may add their own functions to it.

Playlists may have multiple track tags, in that case all of them would be applied to the tracks when required. Other features include:

- 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.
- Allows multiple conditions (must follow json format) [V]. Look at playlist metadata for more info 11.3.

The feature must be explicitly enabled on the header menu [7] to work; i.e. a playlist with track tags will not apply them until done so.

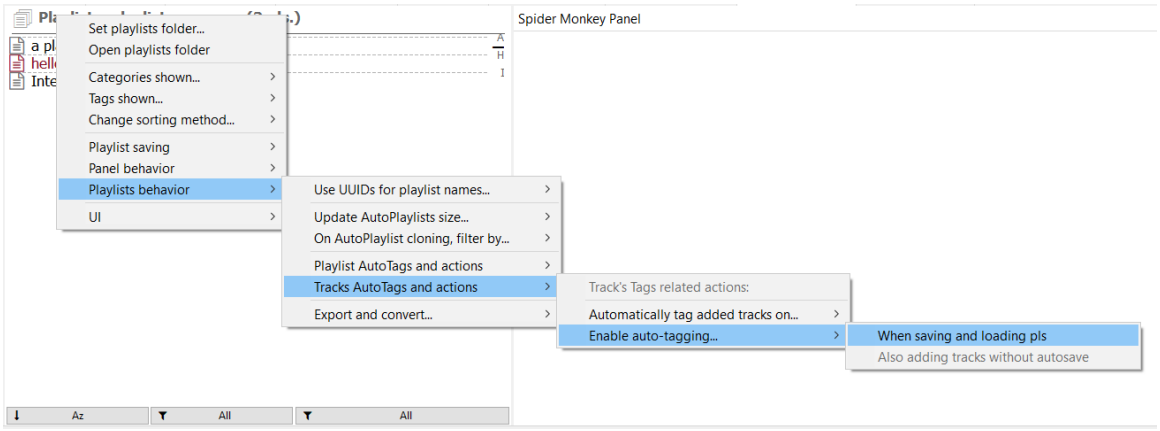

Figure 15: Enabling Automatic track tagging on the header menu.

When a playlist have track tags set, the tooltip shows the tags which would be written in case it's applied:

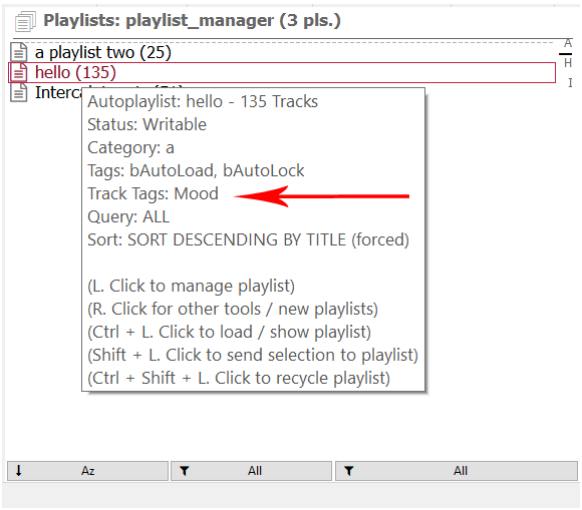

Figure 16: Track tags on tooltip.

4 Exporting Auto-playlist

4.1 Exporting or importing Auto-playlist files

The original idea of a playlist manager was that found on Auto-playlist Manager by marc2003 which consisted only on a list of Auto-playlists which could be loaded on demand... thus to make it easy to transfer all those Auto-playlist to this manager there is an option to directly import its json files[19] on the list contextual menu [7].

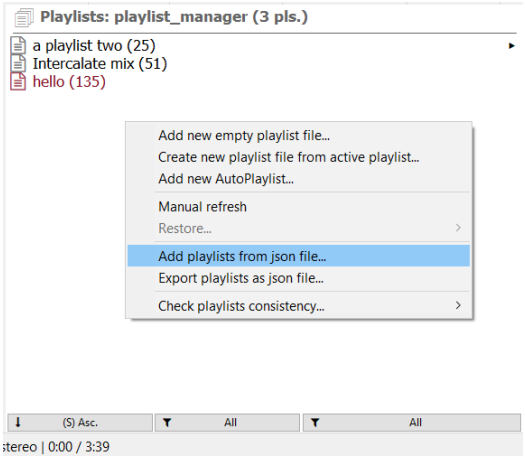

Figure 17:

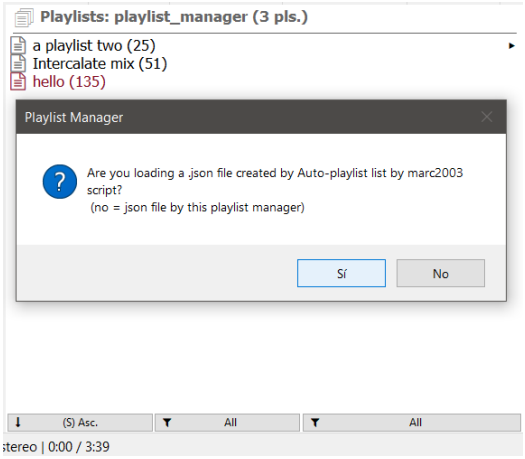

Figure 18:

At the importing process 2 options may be chosen depending on the json file being originary from marc2003's panel or from this one[20]. All Auto-playlists found will be checked for validity and added to the current playlist on the manager.

Exporting the Auto-playlists from this manager is equivalent to the marc2003's one, just copy/paste the appropriate json file[21]. To import it at another panel instance just follow the steps written previously, and choose the appropriate option (files from this panel).

Note the json file from this manager contains both Auto-playlists and .fpl virtual playlists for metadata purposes [14], but the latter are discarded when importing using the menus as described[22]. Anyway the format from this manager is not backwards compatible with marc2003's script, so it doesn't affect in any way for regular users.

Alternatively, Auto-playlists from the panel may be selectively exported using the playlist contextual menu option [4.2].

[19] marc2003's json file (at foobar profile folder) will be at '.\js_data\autoplaylists..json'.

[20] Since marc2003's panel follows its own schema for Auto-playlists, some internal conversion is needed [15]

[21] For ex. if the panel is set to track 'H:\My Music\Playlists', then the playlist json file (at foobar profile folder) will be at '.\js_data\playlistManager_Playlists.json'.

[22] Contrary to marc2003's script, whose json file only has Auto-playlists.

4.2 Export as json file

Single Auto-playlists may be exported as json files, instead of following the general procedure (which exports all of them at once), using the selected playlist contextual menu [7].

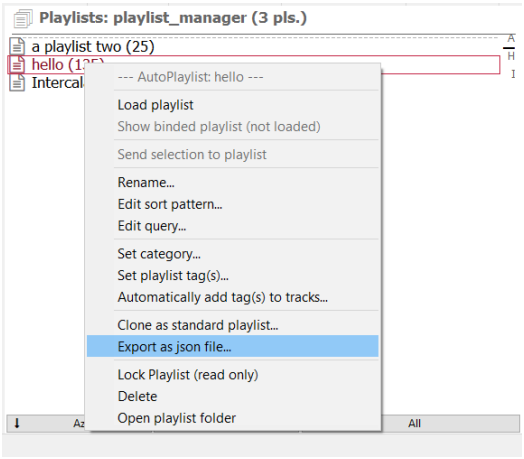

Figure 19: Export selected Auto-playlist.

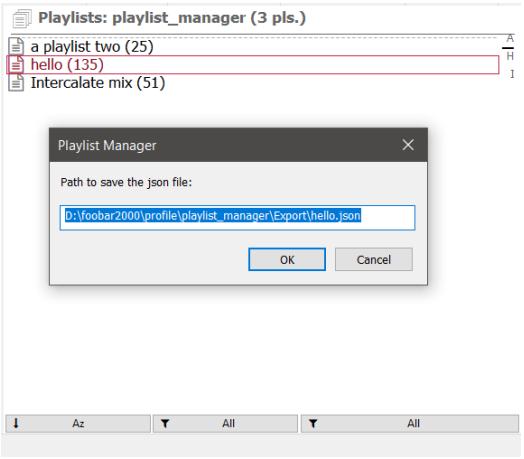

Figure 20: Path to exported json file.

Alternatively all Auto-playlists may be exported at once using the list contextual menu too. This option has an advantage over the general procedure of just copying the json file: .fpl playlists may be filtered before exporting, thus exporting only the Auto-playlists[23].

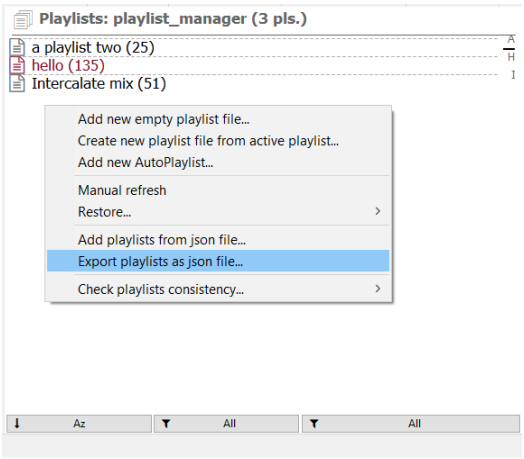

Figure 21: Export all Auto-playlists.

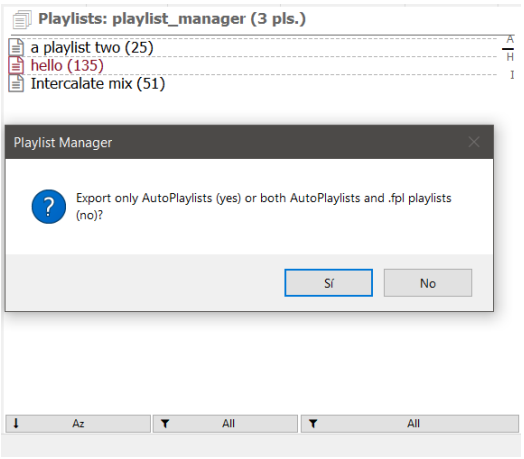

Figure 22: Auto-Playlists and .fpl popup.

[23]Therefore, choosing both playlists types at exporting process is equivalent to simply copying the associated panel json file.

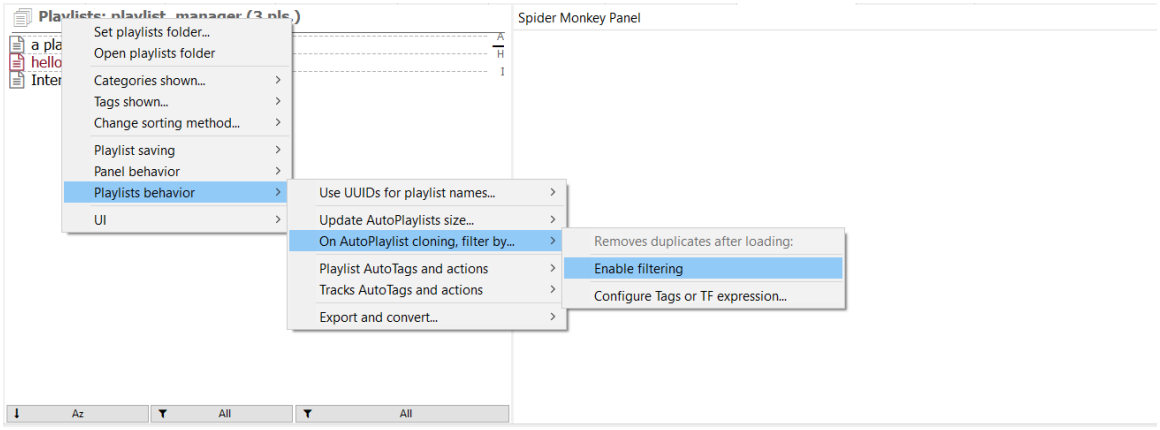

Figure 25: Enabling filtering duplicates for Auto-playlists clones on header menu.

4.3 Clone as standard playlist

Auto-playlists may be converted to standard playlists (writable formats [IV]) for further editing, sorting or exporting as plain-text files. A new playlist file will be created in the tracked folder with similar name and duplicating the tracks and sorting from the Auto-playlist.

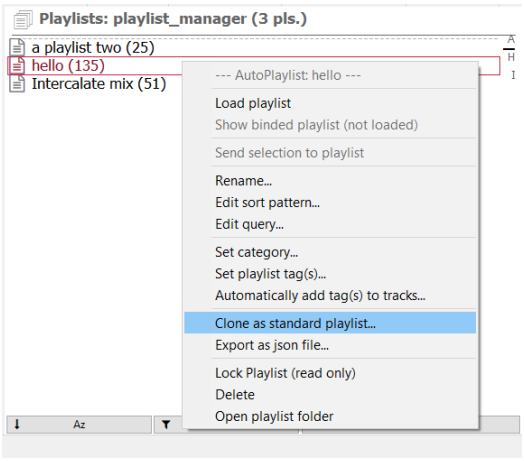

Figure 23: Clone selected Auto-playlist.

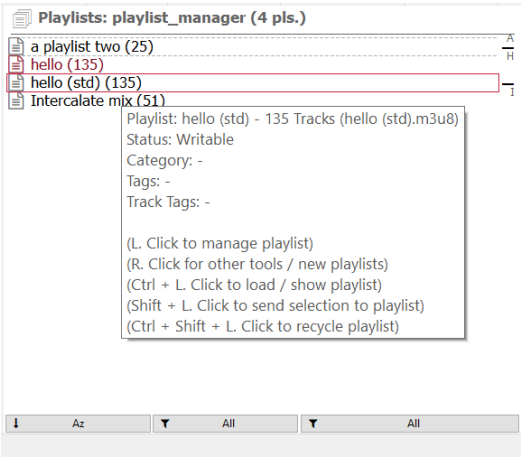

Figure 24: Cloned Auto-playlist as standard playlist.

Additionally, duplicates may be automatically removed according to tag(s) or TF expression(s) by setting 'On AutoPlaylist cloning, filter by....' option. By default is set to 'artist,date,title'[24].

[24] Automatizes the process of removing duplicates by tags after cloning using tools like those found on Playlist-Tools-SMP and automatically fixes one of the worst quirks of Auto-Playlists (having multiple versions of the same tracks)

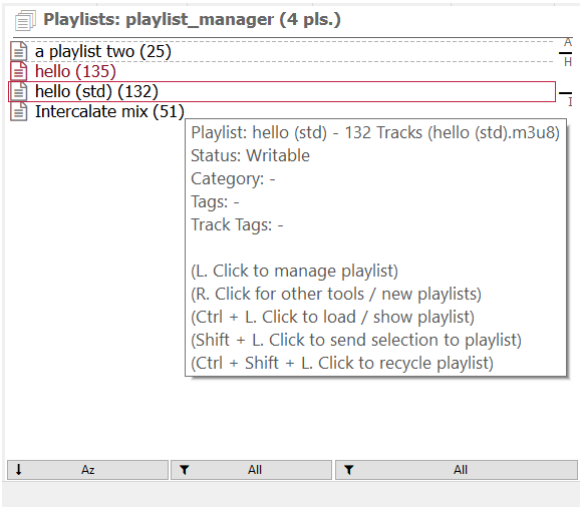

Figure 26: Cloned Auto-playlist now has 3 less tracks (132) than the original (135).

Once an Auto-playlist has been converted, the regular exporting tools may be used [5] to export or convert not only the playlist but also its tracks (for ex. exporting to a portable player).

5 Exporting playlists and files

5.1 Copy Playlist file

Exports (a copy of) the selected playlist file to the given path, the final filename may be changed[25]. This is equivalent to open the tracked folder and copying/pasting the file to the desired location.

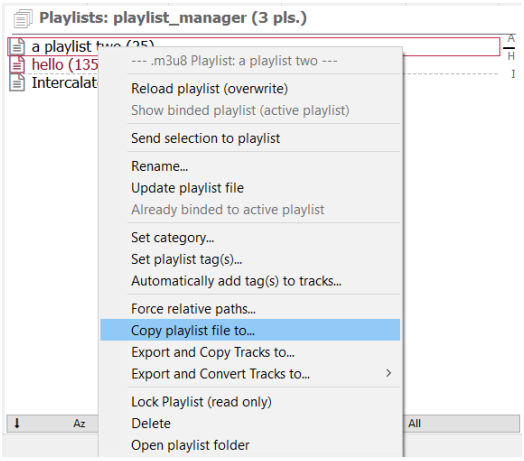

Figure 27: Copy selected playlist file.

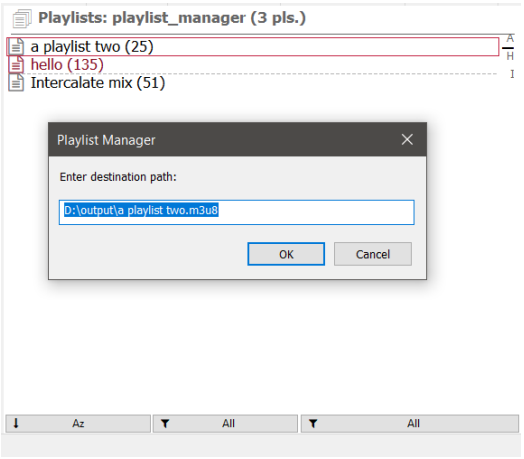

Figure 28: Output popup.

D:\foobar2000\profile\playlist_manager\example.m3u8 → **D:\output\example.m3u8**

This is the only option available for readable-only formats [IV] which have a physical file (.fpl). The Auto-playlist counterpart would be exporting as json file [4.2].

5.2 Export and copy tracks to

Exports (a copy of) the selected playlist file along their tracks to the given path[26], the final playlist filename may be changed.

[25] If the folder does not exists, it will be created too.

[26] It's obviously recommended to choose a new folder without any content, since it will be filled with all tracks plus the playlist.

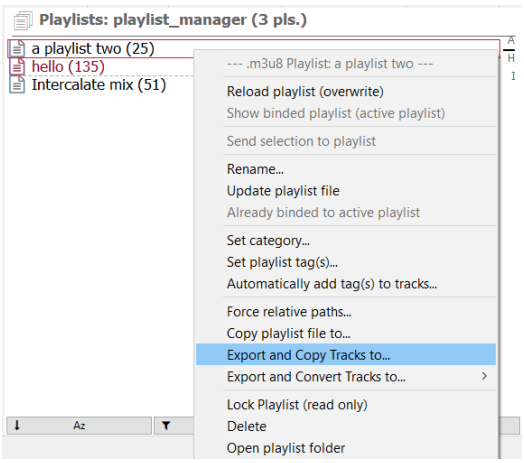

Figure 29: Copy selected playlist file and its tracks.

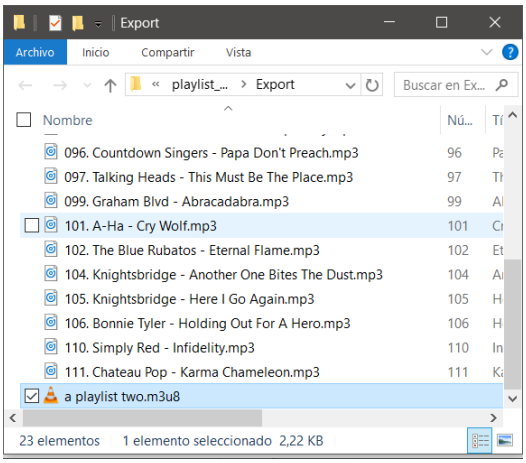

Figure 30: Output folder with all media files along the exported playlist.

Since the tracks are exported 'as is'[27] along the playlist, the exported playlist will be edited to use relative paths, no matter what the original used. This is done to easily load the playlist at any point along its files as a portable solution.

[...]
#EXTINF:259,Jaki Graham - Round And Around
D:\My library\Big Retro Hits 90s\004. Round And Around.mp3
[...]

⇓⇓⇓

[...]
#EXTINF:259,Jaki Graham - Round And Around
.\004. Round And Around.mp3
[...]

This exporting option is not available for readable-only formats [IV]. To use it on Autoplaylist, first clone it as standard playlist [4.3], then proceed as usual. To do something similar with .fpl playlists, manually convert them to a writable format and then proceed as usual.

5.3 Export and convert tracks to

Exports (a copy of) the selected playlist file along their tracks to the given path, the final playlist filename may be changed. The tracks are converted on the process[28] and the exported playlist is edited to use relative paths [5.2].

Since the files are converted, instead of being copied, they are ready to use not only on other PCs of Foobar2000 instances but also on portable players, phones, etc. i.e. it may be used as a one way sync tool integrated within the manager and working directly on playlists.

[27] No conversion is done, so the file formats will remain the same and they will be perfect copies of the original files.
[28] Using pre-defined Converter presets.

The feature allows to save predefined sets of converter preset + destination folder on the menu for easy access[29]. The entries may be configured, added or removed on the header menu [7].

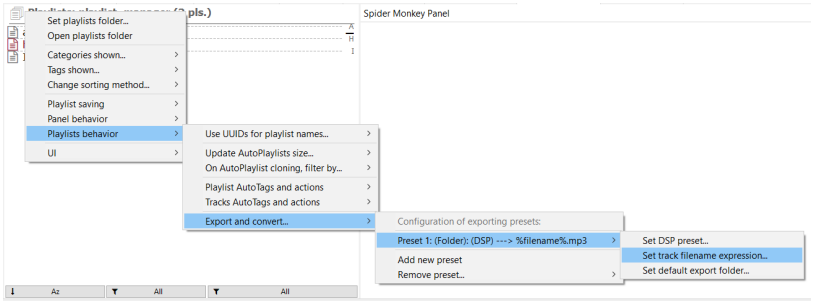

Figure 31: Setting export and convert presets: converter preset, filename mask and output folder.

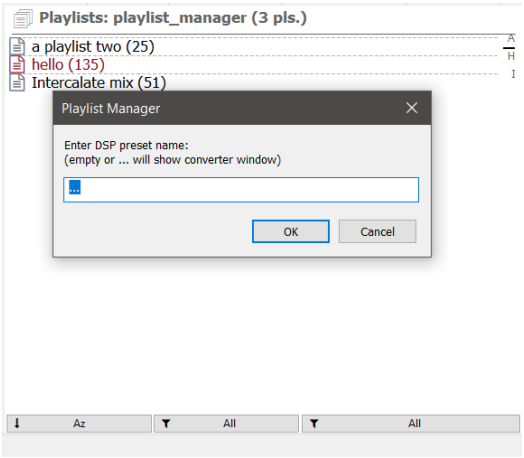

Figure 32: Converter window will be shown at execution.

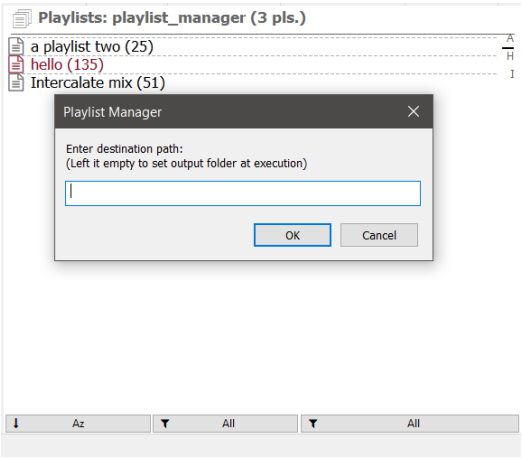

Figure 33: Output path will be asked at execution.

[29]For ex. it's possible to have an entry to export playlist to the Ipod and another one for the server, both with different converter configurations and destination folders.

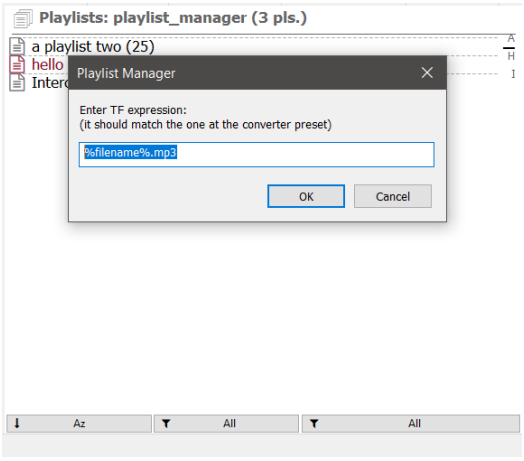

Figure 34: Filename mask must match the one at the converter preset to properly modify names on the output playlist file.

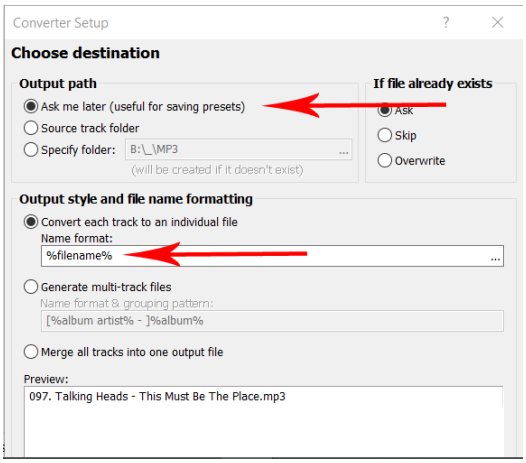

Figure 35: The converter preset can be set to ask for the output path at execution. Note filemask matches the previous one.

On the selected playlist contextual menu, every entry shows the destination folder[30], the converter preset name and the filename mask[31]. When the folder or the preset is not set, '(Folder)' and/or '(DSP)' is shown instead (and they will have to be manually set at execution).

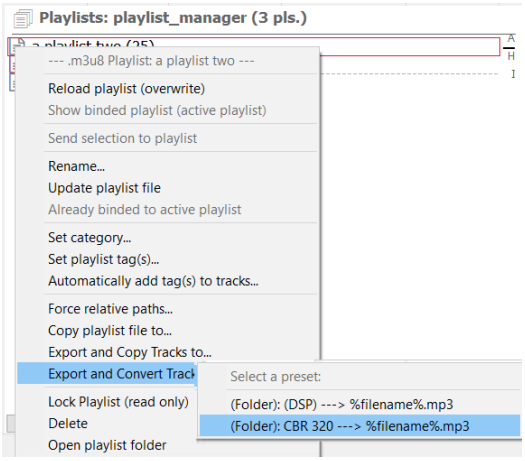

Figure 36: The current preset has a converted preset defined and the filemask, but output folder will be asked at execution.

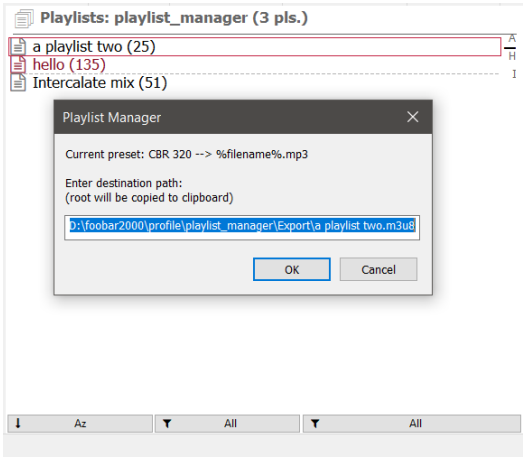

Figure 37: Setting output path. Converter preset should be set to also ask output path before conversion to reuse it in both windows.

This exporting option is not available for readable-only formats[IV]. To use it on Autoplaylist, first clone it as standard playlist [4.3], then proceed as usual. To do something similar with .fpl playlists, manually convert them to a writable format and then proceed as usual.

[30] Along the disk letter in parenthesis.
[31] Must match the one found at the converter preset to work as intended.

6 Additional tools

6.1 On selected playlist

The following set of tools can be found on the selected playlist contextual menu [7].

6.1.1 Force relative paths

Paths within a playlist can be converted to relative paths stripping all but the filename[32]. It's a destructive action, which edits the playlist file and can not be undone... although it a backup can be found at the Recycle Bin [1.6]:

[...]
#EXTINF:259,Jaki Graham - Round And Around
D:\library\Big Retro Hits 90s\004. Round And Around.mp3
[...]

⇓⇓⇓

[...]
#EXTINF:259,Jaki Graham - Round And Around
.\004. Round And Around.mp3
[...]

6.2 On entire list

The following set of tools can be found on the list contextual menu [7].

6.2.1 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-SMP[33].

6.2.2 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

[32] It's easy to see this is equivalent to using the 'export and copy tracks' feature [5.2] without copying the tracks and overwriting the original playlist file
[33] Look for 'Playlist Revive'.

not dead items, since they do exist outside Foobar2000 database.

6.2.3 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 ex[34]:

[...]
#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 ∗

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-SMP[35]

6.2.4 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 fixed[36].

[34] Only the tracks with an ∗will be considered duplicates.
[35] Look for 'Duplicates and tag filtering'. Foobar2000's main menu 'Edit/Remove duplicates', after loading the playlist, can also be used.
[36] It 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 [7].

7 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

8 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).

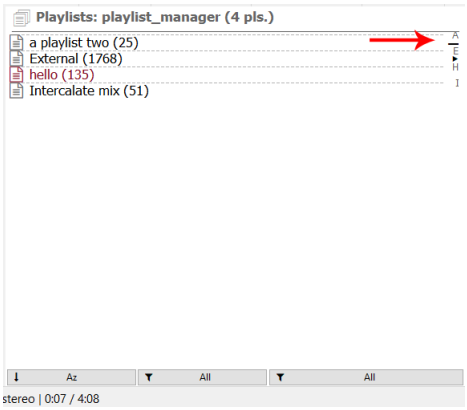

Figure 38: Now playling ('External'), loaded ('a playlist two') and two non loaded playlist.

9 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 purposes[37], not for playlist categorization.

9.1 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.

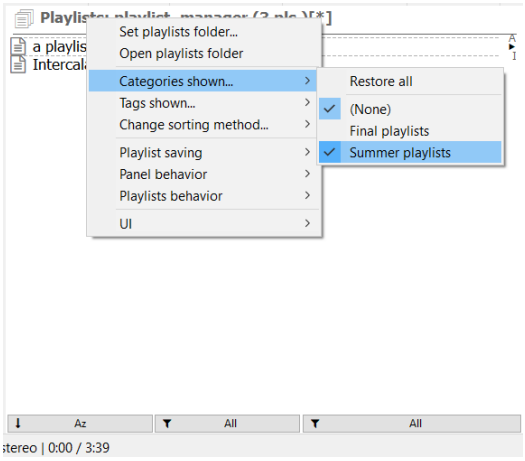

Figure 39: 'Final playlists' category has been exluded.

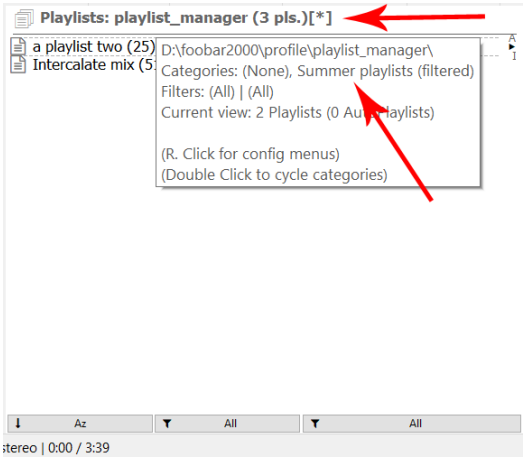

Figure 40: The header and tooltip indicates a filter is active.

An additional way to filter by category is cycling the current category being shown (one by one) [7]. 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.

[37] In fact is also used for some complex playlist automatic actions.

9.2 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.

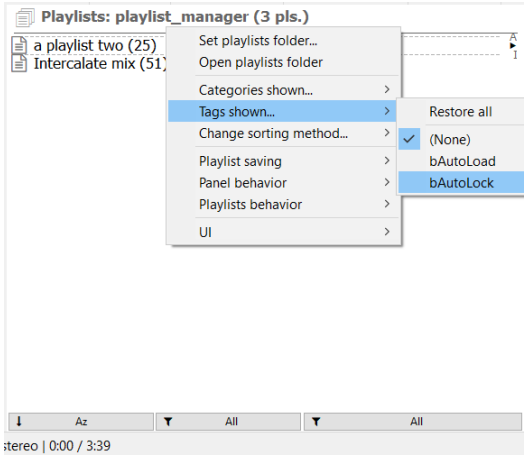

Figure 41: 2 tags have been exluded, showing only playlists without tags.

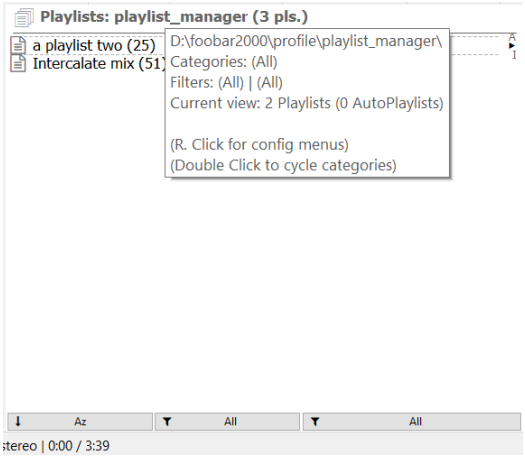

Figure 42: Note the header does not warn about filtering for tags.

9.3 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.

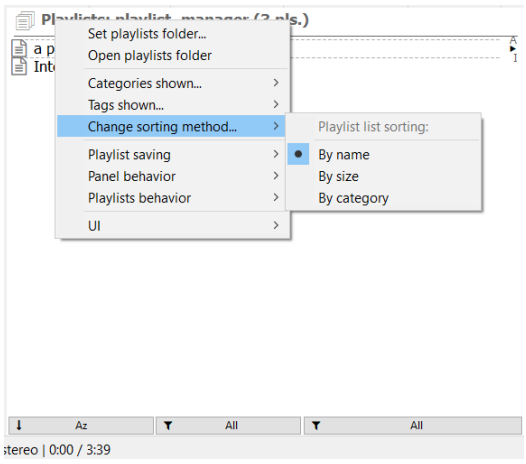

Figure 43: Sorting menu on header menu with the different modes.

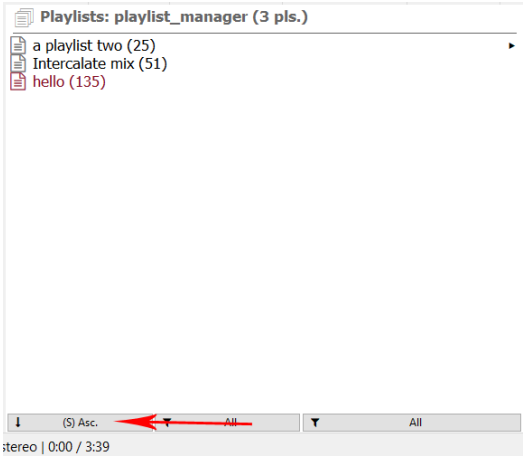

Figure 44: Sorted by size (note the '(S)' on the sorting order button).

Additionally, name / category separators may be shown on the UI to easily identify where the next char begins (123, A, B, ...). 'Size' mode does not make use of this feature.

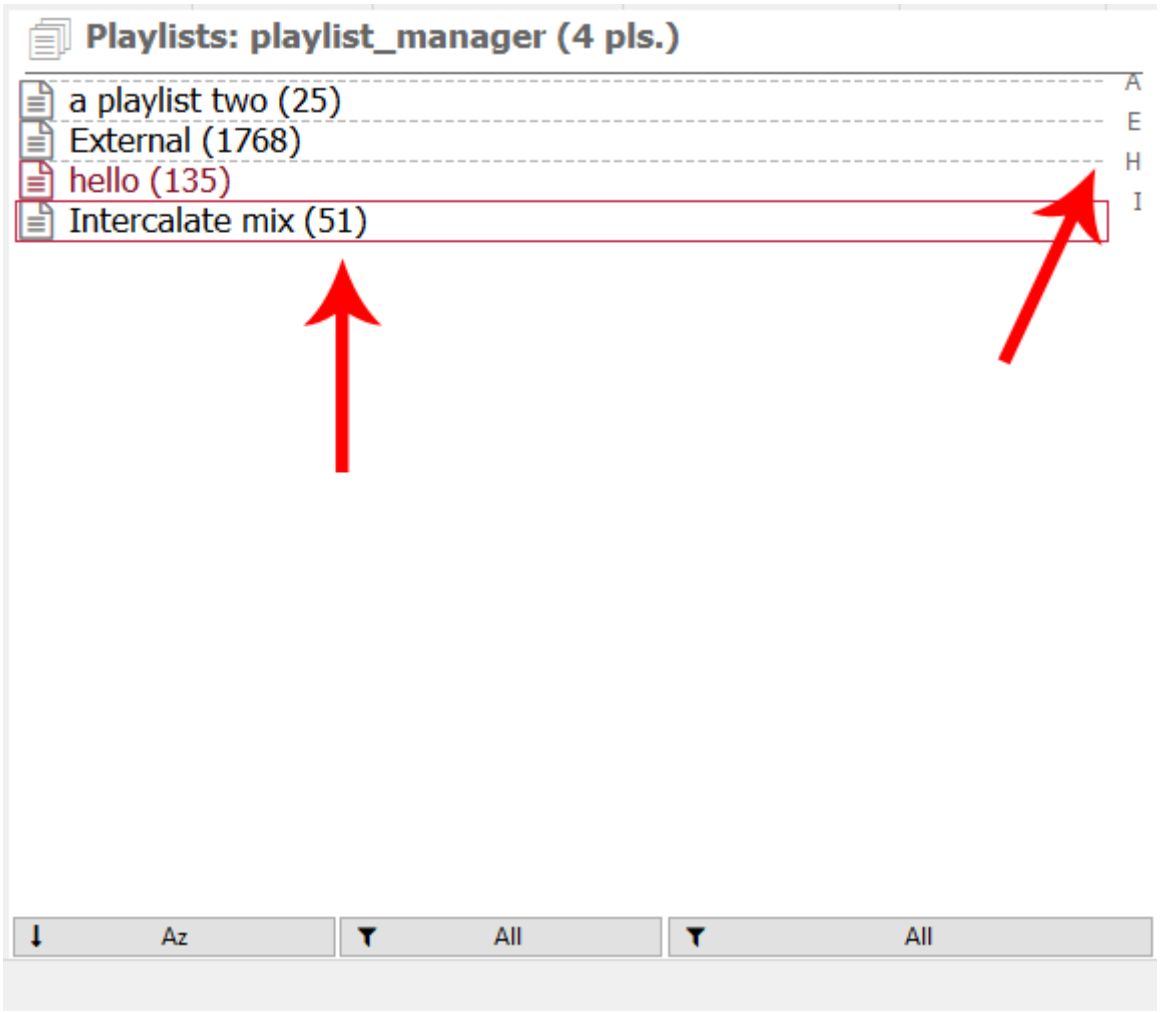

Figure 45: 'A', 'H' and 'I' separators are shown for names.

9.4 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).

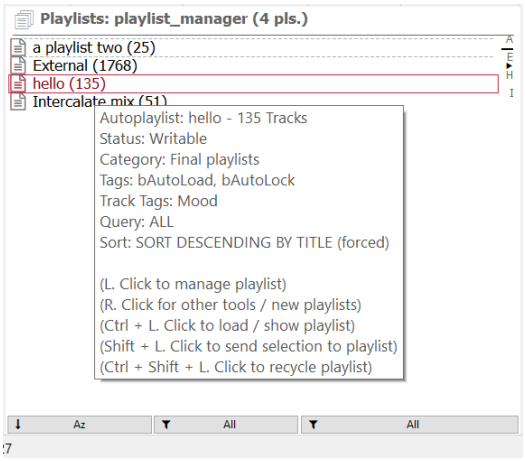

Figure 46: Playlist tooltip.

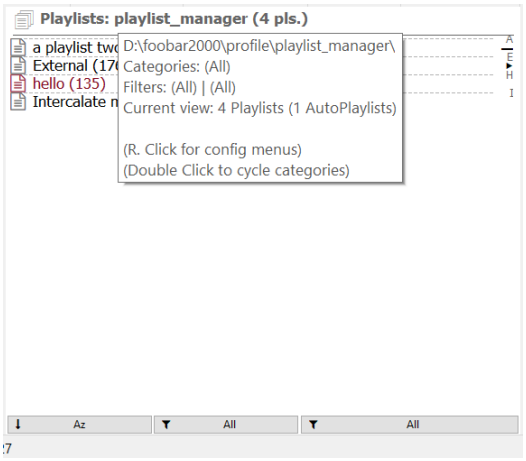

Figure 47: Header tooltip.

10 Customization

10.1 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.

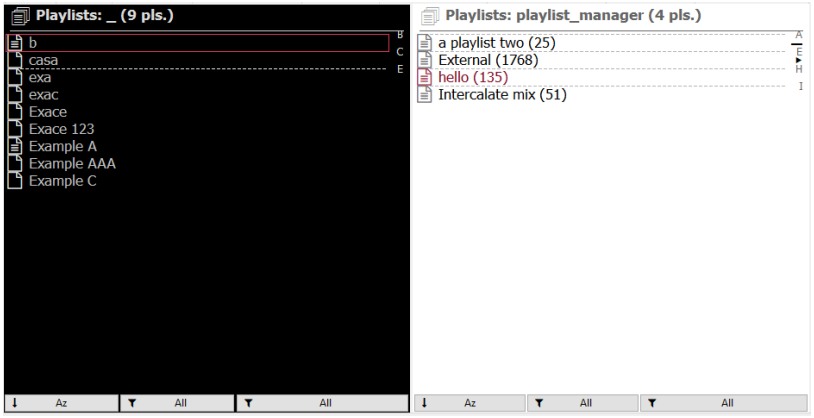

Figure 48: 2 panel instances with different UI colors set.

10.2 Others

- Tooltip: Shortcuts info [7] can be shown or hidden.

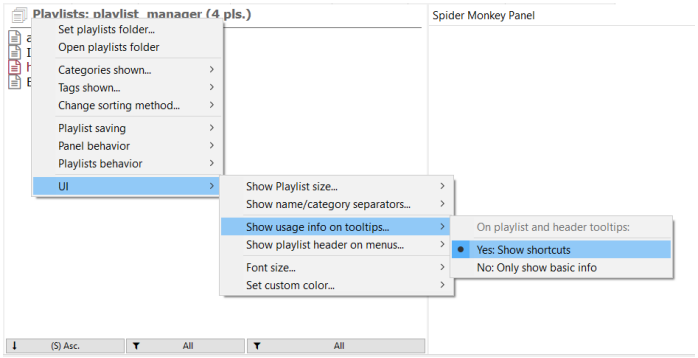

Figure 49: Header menu to enable shortcuts info.

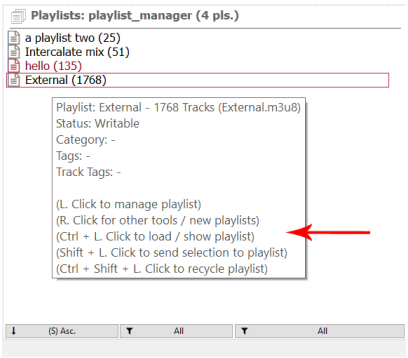

Figure 50: Tooltip.

- Menus: menu header for selected playlist contextual menu [7] can be shown or hidden.

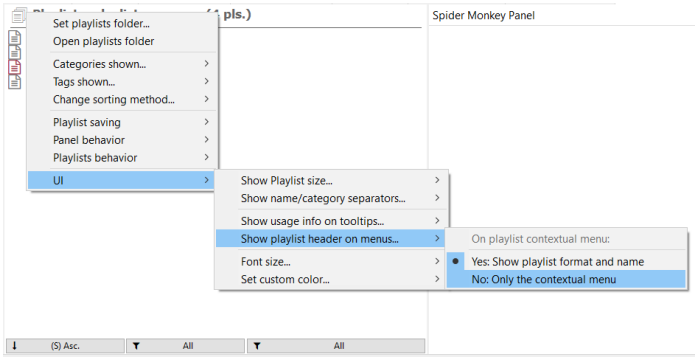

Figure 51: Header menu to enable playlist headers.

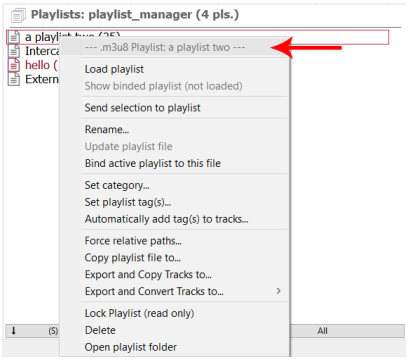

Figure 52: Contextual menu.

- 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.

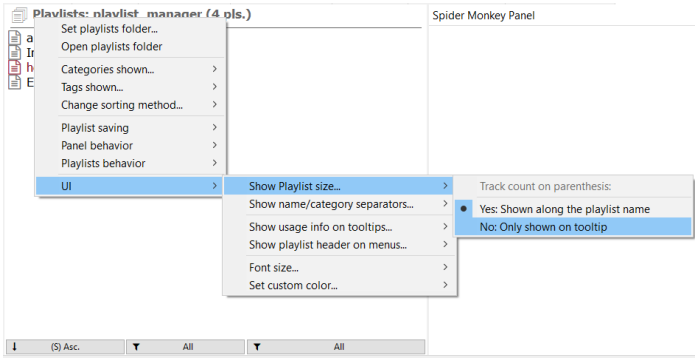

Figure 53:

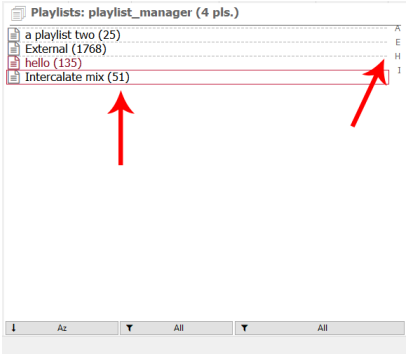

Figure 54: Size and separators are shown for playlists.

- Font size: applies to all text within the panel.

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 [6.2.1]. 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).[38]
  * **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 provided[39]. 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.

[38] A limited functionality version has been included which applies when cloning Auto-playlists if configured to do so [4.3].

[39] Technically 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 formats[40]: .fpl and Auto-Playlists.

In general, writable formats work with more features than their readable only counterparts[41]. This is specially true for .fpl playlists, which are pretty limited on all aspects[42]. Therefore, the use of .m3u8 playlists is preferred over other formats (.fpl or .pls) whenever it is possible.

11 Playlist metadata

11.1 Lock state

Playlist files may be locked or not. Locked files are considered read only for all purposes and therefore never rewritten unless forced to do so [by the user]. Native foobar playlists files (.fpl) are locked by default.

11.2 [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).
- '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.

[40] For all purposes, locked playlists may be considered into this category, no matter their extension.
[41] There are some more exceptions to this rule. For example, .m3u8 playlists have more features than .pls ones.
[42] Neither exporting [5], nor additional tools [6] work with them. And obviously, since they are a readable only format, no new tracks may be added to them using the manager.

11.3 Track tags

["tagName":"tagValue"]

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. Allows multiple conditions (must follow json format) [V] where the tag value(s) can be any of the following:

- Foobar2000 Title Format expressions (or %tags%)[43].
- JavaScript functions (defined at 'helpers\helpers_xxx_utils.js'), prefixed by 'JS:'.
- Value (string or number).

11.4 Category

Playlists may have a single category for easy classification or which can be used as virtual folder. For informative purposes or arbitrary keywords use [playlist] tags instead]11.2].

11.5 UUID

Suffixes added to the playlist names to separate them from non tracked playlists when loaded in foobar. Some also allow some level of name duplication.

- Invisible Unicode chars plus (*)[44]
- (a-f)[45]
- (*)[46]
- Or none[47].

[43] https://wiki.hydrogenaud.io/index.php?title=Foobar2000:Titleformat_Reference

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

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

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

[47] The 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 playling indicators[8].

12 .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 encoded[48].

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 2: M3U structure.

12.1 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 3: 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 [11]:

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 paths[49] can be found below:

[48] Although 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.

[49] Playlist file would be at current folder and tracks within 'music' subfolder.

Directive	Description	Example
#UUID	Playlist UUID [11.5]	#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) [11.3]	#TRACKTAGS:["Mood":"Chill"]
#PLAYLISTSIZE:	Playlist size (# of tracks)	#PLAYLISTSIZE:2

Table 4: Additional M3U extensions for playlist metadata support.

```
#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
```

13 .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)[50].

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 5: 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 6: 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 paths[51] 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
```

[50] This 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.

[51] Playlist file would be at current folder and tracks within 'music' subfolder.

14 .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 program[52].

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 penalties[53]. This is the behavior followed by manager and it has been already reported to Foobar2000 developers [54] 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) [6].

To allow additional metadata for .fpl playlists, considering the files are non-editable, an external json file is used [V]. 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": ["Rating": 5]

Table 7: 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 cached[55]. 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.

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 [{...}, {...}]:

[52]Instead of loading the tracks and retrieving their metadata from the physical files.
[53]Which is one of the most common use-case of playlists within Foobar2000.
[54]Check Why m3u8 loading is so slow on hydrogenaud.io-
[55]For 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.

```
[
    {
        "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"]
    }
]
```

15 Auto-Playlists

Structure and format is exactly the same than .fpl use-case, so check it for reference [14]. 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 8: 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[56]. 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": []
    }
]
```


[56] For ex. if the panel is set to track 'H:\My Music\Playlists', then the playlist json file (at foobar profile folder) will be at '.\js_data\playlistManager_Playlists.json'.

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!

- **What's json?** It's a standard file structure. Check https://en.wikipedia.org/wiki/JSON for more info.

- **Can't edit or update a playlist:** Check the playlist status. It's probably locked, unlock it to be able to make changes.

Part VI

Tips

16 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 SymLinks[57] 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.

17 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.

18 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 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.

[57] Symbolic 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.

- **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.

19 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.

20 Working with locked playlists

- **Forget about lock status on Foobar2000 playlists.** There are some plugins or even Spider Monkey Panel scripts which allow to lock/unlock native playlists in some way or another. While it may come useful for advanced users, regular users should simply stick to the manager lock features. Playlists now have a physical file counterpart which can be locked, so there is no need to lock playlists within UI for changes[58].

- **Instead of forbidding edits, just reload the playlist.** The native approach focuses on forbidding specific actions on playlists (reordering, adding, removing items, etc.). Locking the playlist file allows any of those edits on the loaded playlist and if you want to revert them just undo them or reload the playlist to discard all of them and revert it to the original version.

- **Edits on locked playlists can be saved if forced to do so but never automatically.** By default, any change made to a locked playlist loaded within Foobar2000 is not auto-saved and therefore only temporarily stored while the playlist remains opened... but this can be bypassed on demand without unlocking the playlist using the 'Force playlist file update' entry on the selected playlist contextual menu [7].

[58] This also simplifies some quirks about playlist locks which involve the type of lock and owners...

21 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.[59] i.e. '.\profile\playlist_manager\server\' equals to 'D:\foobar2000\profile\playlist_manager\server\'.

- **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.

[59] When checking tracks, their root is considered to be the playlist path.