DisOrder Playlist Design

Introduction

Users should be able to construct lists of tracks which can then be queued as a unit.

It is convenient for these lists to be kept on the server: it can the allow shared access to playlists and allow users to access their playlists from multiple locations.

Playlist Concepts

All playlist commands require the `play` right.

Playlist names take the form `[USER.]PLAYLIST`, with `USER` being the owning user and `PLAYLIST` the name of the playlist (which must be alphanumeric and nonempty).

Playlists can be public, private or shared.

- a public playlist can be read by anyone, but only modified by its owner.
- a private playlist can only be read or modified by its owner.
- a shared playlist can be read or modified by anyone.

Playlists with no owner are always shared.

The server manages playlists but does not have a command to play them. Clients should use `playlist-get` and then `play`. The server is responsible for maintaining shared state, not for actually processing it.

For the benefit of clients, the event log will get notifications of any playlists readable by their user.

New Commands For Managing Playlists

playlist-get

```
playlist-get NAME
```

Returns the contents of the playlist in a body, one track to a line, or 555 if the playlist does not exist.

playlist-set

```
playlist-set NAME
TRACK
TRACK...
.
```

Sets the contents of a playlist. The request body should have one track per line. If the playlist did not already exist it is created as a private playlist (if it has an owner) or a shared one (if it does not).

The tracks need not actually exist.

Playlists can only be modified if the lock on them is held (see below). This applies even to non-shared playlists.

NB. This is the first example of a *command* that takes a body. Previously only responses had bodies.

playlists

```
playlists
```

Lists all playlists readable or modifiable by the calling user. (So private playlists belonging to other users are not returned.) There is no ordering guarantee.

playlist-delete

```
playlist-delete NAME
```

Deletes a playlist. The caller must be able to modify it. There is no way to recover playlists; it assumed that the user community is internally coöperative.

playlist-set-share

```
playlist-set-share NAME STATUS
```

Sets the playlist to `public`, `private` or `shared`.

playlist-get-share

```
playlist-get-share NAME
```

Returns `public`, `private` or `shared`.

playlist-lock

```
playlist-lock NAME
```

Locks a playlist. Other users won't be able to edit the playlist while it is locked. Locks belong to connections, and

are lost upon disconnection. A given connection may only hold one lock at a time and should not hold the lock any longer than necessary.

If the playlist is already locked then an error is immediately returned. The client should either give up or wait a little while and try again.

playlist-unlock

`playlist-unlock`

Unlocks a locked playlist.

Event Log

TODO some of these messages are described as only being sent to the playlist owner. The infrastructure to support this doesn't exist yet so this restriction might be left out. This does expose playlist names where they shouldn't be but this isn't considered unacceptable initially.

The following event log messages will be added:

playlist_created

`playlist_created NAME STATUS`

Only sent for public or shared playlists and to the owner.

playlist_deleted

`playlist_deleted NAME`

Only sent for public or shared playlists and to the owner.

playlist_modified

`playlist_modified NAME STATUS`

Sent for public or shared playlists, and to the owner, and when a playlist changes state. (For instance if a public playlist goes private this message will still be sent.)

Database

We'll have a single database `playlists.db`. This will be a hash, with the keys being playlist names.

Values will be key-value pairs with the following known keys:

- `sharing` - should be `public`, `private` or `shared`.
- `count` - the total number of tracks
- `0`, `1`, `2`, ... - the first, second, third etc track

In principle there might be numbered keys beyond `count`; these will be ignored.

`disorder-dump` will save and restore the contents.

Command-Line Client

Most of this is obvious and doesn't need designing. When setting a playlist's contents, the new contents will be read from stdin terminated either by "." or EOF, or (via an option) from a named file. For `playlist-set` the lock will be automatically taken and the locking commands won't be directly available.

No dot-unstuffing is done as all valid track names start "/" anyway.

Web Interface

Playlists won't be implemented in the web interface at least in the first release.

Disobedience

Original Outline

The obvious approach here is to re-use the main queue display, with its drag+drop support.

The choose/recent/new popup menus will gain a new submenu allowing selected tracks to be added to any of the playlists modifiable by this user. We'll cache the set of playlists and update it from `playlist_` event log messages, so there need be no additional delay.

We'll have a new "Playlists" main-menu item with an option to create a playlist and an option for each known playlist. This will bring up the edit playlist window for that playlist, which will contain play and delete buttons as well as the ability to modify it.

Menu

The **Edit** menu will have a new **Edit Playlists** option which will bring up the playlists window. More on this below.

The **Control** menu will have an **Activate Playlists** submenu with all playlists that the user can read. Select one will just copy that playlist's tracks to the queue.

Playlist window

This will consists of two halves.

The left hand side will be a list of all playlists, with add and remove buttons below. The right hand side will be a queuelike in which the current playlist can be edited. Note that it's necessary for one track to be able to appear more than once in a playlist (suppose your playlist is *Carmina Burana* and you only have one copy of *O Fortuna*).

Dragging from the list of playlists will allow all the tracks in it to be dragged into a point in the queue. Only one playlist at a time will be draggable this way.

Selecting a playlist will make it the current playlist and will display it in the right hand panel.

Dragging tracks form the choose tab (or elsewhere) into a playlist will insert them into the playlist, if possible at the drop point. This is the only supported way of modifying existing playlists.

Dragging tracks from the playlist into the queue will insert those tracks in the queue at the drop location.

Right clicking over the playlist will bring up a menu with the following options:

- Track properties
- Select all tracks
- Deselect all tracks
- Remove track form playlist
- Play track

- Play whole playlist

Pressing the 'add' button will pop up a window with a text entry field and a group of radio buttons allowing choice between public, private and shared status for the new playlist. The OK button will be grayed out unless the new name and status is acceptable. The new playlist will be empty and will be selected. (Of course there will be a Cancel button too.)

Pressing the 'delete' button will pop up a window asking if the user is sure. Playlist deletion is irrevocable (which is arguably a bug but not one that is likely to be fixed in this version).

Playlist Mutation And Locks

The goal is to avoid mutations being *silently* lost. If a mutation fails because the playlist changed too much for it to apply then that's OK as long as the user sees an error message. Better is for the mutation to happen but (for instance) for tracks to end up in the wrong place - they can then be rearranged to somewhere more sensible.

The algorithm for removing selected tracks is:

1. acquire the lock
2. refetch the playlist
3. remove the tracks
4. store the playlist
5. update the list store
6. release the lock

Similarly to insert tracks into a playlist:

1. acquire the lock
2. refetch the playlist
3. insert the tracks
4. store the playlist
5. update the list store
6. release the lock

Locks are never held for long.

Testing this with dueling copies of Disobedience will be a nuisance!