NetJockey RESTful API Draft 1

by Tomas Jasek

1 Definitions

1.1 Song

Song is a single **video clip**. In all cases, it comes from **YouTube**. It has:

1. **duration** - how long it is (in seconds)
2. **title** - aka name of the video
3. **id** - youtube video ID
4. **thumbnailUrl** - url to thumbnail (direct link to image)
5. **uuid** - unique ID of this queue entry(alphanumeric value only)

JSON definition:

```
{
  "duration": "150",
  "title": "Mermaid Van - DJ Command Remix",
  "id": "asdffhah",
  "thumbnailUrl": "",
  "uuid": "r8tng309n6tyh0n69gmy8"
}
```

1.2 Song Queue

Song queue is a FIFO structure for storing playlist.

Note: Array of songs is in such order, that its 1st element(at position 0) is the song that is currently playing, following song will play after it and so on. The last element of the array will be the last song that is played.

JSON definition:

```
{
  "playlist": [ /* array of songs */ ]
}
```

1.3 RoomInfo

RoomInfo object stores meta-data about a room.

1. **name** - human-readable name of room
2. **id** - machine-readable identifier of room(alphanumeric chars only, no other chars, up to 256 chars)

JSON definition:

```
{
  "name": "gaspiGASM",
  "id": "5894y9"
}
```

1.4 Room

Each room has:

1. **roomInfo** object
2. **current song time** - how long the current song is currently playing(in seconds)
3. **song queue** - defined above

JSON definition:

```
{
  "roomInfo" : {
    /* see RoomInfo above */
  },
  "currentSongTime": "150",
  "queue": {
    /* see Song Queue above */
  }
}
```

2 API versioning

This is version 1 of the API. API versions will increment when there are important changes to the system. Version numbers affect the URL that is used to access endpoints.

2.1 Full endpoints URL

API endpoints are accessible via:

```
http://[server]:[port]/v[versionNumber]/[endpoint]
```

For example:

```
http://netjock.ey/v1/rooms
```

3 RESTful API endpoints

3.1 GET `/rooms`

Obtains list of available rooms as JSON array. In this case, roomInfo object is sent for each room. Other fields of room are not relevant. Only public rooms are listed.

Response type: Custom

```
{
  "rooms" : [
    /* array of roomInfo */
  ]
}
```

3.2 GET `/room/[roomID]`

Return the entire room object(may be useful for initialization).

If room with given ID is not found, status will be set to **404**.

Response type: Room

3.3 GET `/room/[roomID]/info`

Access `roomInfo` object of a room with given identifier.

If room with given ID is not found, status will be set to **404**.

Response type: Room Info

3.4 GET `/room/[roomID]/queue`

Access playlist of a room.

If room with given ID is not found, status will be set to **404**.

Response type: Song Queue

3.5 GET `/room/[roomID]/time`

Returns only `currentSongTime` field from SongQueue object.

If room with given ID is not found, status will be set to **404**.

Response type: JSON with just currentSongTime from room structure.

```
{
  "currentSongTime": "150"
}
```

3.6 POST `/room/[roomID]/add-song`

Adds song to queue of given room.

Example Request Body:

```
{
  "url": "http://youtube.com/watch?v=abcdefgh137"
}
```

If room with given ID is not found, or URL is malformed or unsupported, status will be set to **400**.

3.7 POST `/rooms/add`

Creates a new room and assigns it an id. It's possible to rename a room later.

Request body: Custom

```
{
  name: "Richyho NonCancer Zone"
}
```

Response type: Wrapped RoomInfo

```
{
  "roomInfo": {/* roomInfo object */}
}
```

3.8 POST `/room/[roomID]/rename`

Renames an existing room. ID is unaffected.

Request body: Custom

```
{
  "name": "New Name Of The Room"
}
```

Response: 200 if success, 404 = room not found, 400 is other error