Introduction

Welcome to the Juice WRLD API documentation. This comprehensive database contains detailed information about Juice WRLD's entire discography, including released tracks, unreleased songs, unsurfaced material, and studio recording sessions. The database is organized into four main categories with rich metadata for each track.

Tracker

The Tracker is your hub for exploring the full collection. Use filters and search to find songs quickly:

Category: switch between released, unreleased, unsurfaced, and recording sessions.
Era: narrow results by Juice WRLD career era.
Search: search across titles, credited artists, and producers.
Pagination/Infinite Scroll: browse large result sets smoothly.

Player

The Player shows tracks that are playable and streamable. You can preview, queue, and download playable content.

Playable list: songs with available audio files.
Stream: click a song to start playback.
Queue: build a playlist during a session.
Share: create a shareable playlist link when needed.

Radio

Radio mode surfaces a random playable track. It’s a great way to discover songs without browsing manually.

Randomize: fetch a random playable song with a single click.
Quick play: instantly play or add to your queue.

Files

The Files page lets you browse the organized collection, preview cover art, and download individual files or ZIPs of selected items.

Browse folders: navigate the Compilation, Original Files, and other directories.
Search: filter by name or extension (e.g., “.mp3”, “.wav”, “.zip”).
Cover art: hover or open file info to view embedded artwork when available.
Download: download a single file directly.
ZIP selection: select multiple items and create a ZIP for bulk download.

Sharing & Playlists

For larger or curated sets, generate a shareable playlist link.

Create share: build a playlist and save a share link.
Preview: use the share preview to check details before opening.
Load from link: paste a shared link to load playlist contents.

Juice WRLD API Documentation
Complete reference for all available endpoints and features

API Status
API Online
Base URL: https://juicewrldapi.com/juicewrld
Available Endpoints
The Juice WRLD API provides several endpoints for accessing different types of data. Each endpoint is documented below with detailed information about parameters, responses, and usage examples.

Songs
Get song data with filtering and search
Statistics
Get counts and statistics by category
Categories
Get available song categories
Eras
Get Juice WRLD career eras with details
Audio Player
Playable songs with audio streaming
File Browser
Browse, fetch info, cover art and download files
ZIP Jobs
Create ZIPs from selected paths with progress
Shared Playlists
Create and retrieve shareable playlists
Radio
Fetch a random playable song
Songs API
Get All Songs
Retrieve a paginated list of songs with optional filtering
GET
/songs/
Description: Retrieve a paginated list of songs with optional filtering
Parameters:
page
Optional
Page number for pagination
page_size
Optional
Number of results per page (default: 20)
category
Optional

Filter by category (released, unreleased, unsurfaced, recording_session)
era
Optional
Filter by era name
search
Optional
Search in song names, credited artists, and track titles (normalizes special characters, e.g. "dont" matches "don't")
searchall
Optional
Search in song names, credited artists, producers, and track titles (normalizes special characters)
lyrics
Optional
Search within song lyrics content
file_names_array
Optional
Format file_names as array instead of string (values: "true", "1", "yes")
Example Request:
```
GET /songs/?page=1&page_size=50&category=released&search=dont go
```
Sample Response:
```
{
  "count": 1234,
  "next": "https://juicewrldapi.com/juicewrld/songs/?page=2",
  "previous": null,
  "results": [
    {
      "id": 1,
      "name": "Lucid Dreams",
      "category": "released",
      "era": {
        "name": "Goodbye & Good Riddance"
      },
      "credited_artists": "Juice WRLD",
      "producers": "Nick Mira"
    }
  ]
}
```
Get Song by ID
Retrieve detailed information about a specific song
GET
/songs/{id}/
Description: Retrieve detailed information about a specific song
Parameters:
id
Required
Unique identifier for the song

Example Request:

GET /songs/1/

Sample Response:

```
{
  "id": 1,
  "name": "Lucid Dreams",
  "category": "released",
  "track_titles": [
    "Lucid Dreams",
    "Lucid Dreams (Forget Me)"
  ],
  "credited_artists": "Juice WRLD",
  "producers": "Nick Mira",
  "length": "3:59"
}
```

Statistics API

Get Statistics

Get overall statistics about songs and categories

GET

/stats/

Description: Get overall statistics about songs and categories

Example Request:

GET /stats/

Sample Response:

```
{
  "total_songs": 39847,
  "category_stats": {
    "released": 156,
    "unreleased": 2847,
    "unsurfaced": 15234,
    "recording_session": 21610
  },
  "era_stats": {
    "JuiceTheKidd": 5234,
    "Affliction": 1847,
    "Juice WRLD 999": 8934,
    "Goodbye & Good Riddance": 4521,
    "World On Drugs": 3245,
    "Death Race For Love": 7632
  }
}
```

Categories API

Get Categories

Get list of available song categories

GET

/categories/

Description: Get list of available song categories

Example Request:

```
GET /categories/
```

Sample Response:

```
{
  "categories": [
    {
      "value": "released",
      "label": "Released"
    },
    {
      "value": "unreleased",
      "label": "Unreleased"
    },
    {
      "value": "unsurfaced",
      "label": "Unsurfaced"
    },
    {
      "value": "recording_session",
      "label": "Studio Sessions"
    }
  ]
}
```

Eras API

Get Eras

Get list of all eras with details

GET

/eras/

Description: Get list of all eras with details

Example Request:

```
GET /eras/
```

Sample Response:

```
[
  {
    "id": 1,
    "name": "JuiceTheKidd",
    "description": "JuiceTheKidd Era",
    "time_frame": "(~2014-February 2017)"
  },
  {
    "id": 2,
    "name": "Affliction",
    "description": "Affliction Era",
    "time_frame": "(February 2017-May 2017)"
  },
  {
    "id": 3,
    "name": "Juice WRLD 999",
    "description": "Juice WRLD 999 Era",
    "time_frame": "(May 2017-May 2018)"
  },
  {
    "id": 4,
    "name": "Goodbye & Good Riddance",
    "description": "Goodbye & Good Riddance Era",
    "time_frame": "(May 2018-August 2018)"
  },
  {
    "id": 5,
    "name": "World On Drugs",
    "description": "World On Drugs Era",
    "time_frame": "(August 2018-December 2018)"
  },
  {
    "id": 6,
    "name": "Death Race For Love",
    "description": "Death Race For Love Era",
    "time_frame": "(December 2018-December 2019)"
  }
]
```
Audio Player API
Stream Audio
Stream or download an audio file directly by file path. Supports HTTP range requests for playback.
GET
/files/download/?path={file_path}
Description: Stream or download an audio file directly by file path. Supports HTTP range requests for playback.
Parameters:
path
Required
File path relative to the comp directory
Example Request:
```
GET /files/download/?path={file_path}
```
Sample Response:
```
"Binary audio data with Content-Type: audio/mpeg headers"
```
File Browser API
Browse Files
List files and folders
GET
/files/browse/
Parameters:
path
Optional

Directory path to browse (relative to comp)
search
Optional
Filter items by name
Example Request:
GET /files/browse/?path=Compilation
Get File Info
Fetch metadata for a file
GET
/files/info/
Parameters:
path
Required
File path (relative to comp)
GET /files/info/?path=Compilation/song.mp3
Get Cover Art
Extract embedded cover art
GET
/files/cover-art/
Parameters:
path
Required
Audio file path (relative to comp)
GET /files/cover-art/?path=Compilation/song.mp3
Download File
Stream or download a file
GET
/files/download/
Description: Stream or download a file. Supports HTTP Range requests for efficient audio streaming and seeking.
Parameters:
path
Required
File path (relative to comp)
HTTP Headers:
Range
Optional
HTTP Range header for partial content requests. Format: "bytes=start-end" or "bytes=start-". Enables seeking and efficient streaming for audio players.
Example Request:
GET /files/download/?path=Compilation/song.mp3
Example with Range Request:

```
fetch('/juicewrld/files/download/?path=Compilation/song.mp3', {
  headers: {
    'Range': 'bytes=0-1048575'
  }
})
```

Response:
200 OK: Full file content (when no Range header is sent)
206 Partial Content: Partial file content (when Range header is sent)
Content-Range: Header indicating the byte range returned (e.g., "bytes 0-1048575/5242880")
Content-Length: Size of the returned range
Accept-Ranges: "bytes" indicating range requests are supported
ZIP Jobs API
Start ZIP Job
Create a background ZIP
POST
/start-zip-job/
Body:
{ "paths": ["Compilation/song1.mp3", "Compilation/song2.mp3"] }
ZIP Job Status
Poll progress
GET
/zip-job-status/{job_id}/
Cancel ZIP Job
Cancel an in-progress job
POST
/cancel-zip-job/{job_id}/
ZIP From Selection
Immediate ZIP stream for selected paths
POST
/files/zip-selection/
Body:
{ "paths": ["Compilation/Folder"] }
Shared Playlists API
Create Shared Playlist
Generate a shareable playlist
POST
/playlists/share/
Get Shared Playlist
Fetch playlist by share ID
GET
/playlists/shared/{share_id}/
Get Share Info
Preview metadata for a share
GET
/playlists/shared/{share_id}/info/
Radio API
Get Random Radio Song
Fetch a random playable song with full metadata
GET
/radio/random/
Description: Returns a random playable song from the compilation with full song

metadata. The response includes file information and matching song data from the database.
Example Request:
GET /radio/random/
Sample Response:

```
{
  "id": "Compilation/1. Released Discography/24. Singles & Features/2. YouTube - Singles & Features/1. 2016 - YouTube Singles & Features/AYE AYEEE.mp3",
  "title": "AYE AYEEE",
  "path": "Compilation/1. Released Discography/24. Singles & Features/2. YouTube - Singles & Features/1. 2016 - YouTube Singles & Features/AYE AYEEE.mp3",
  "size": 5195736,
  "modified": "2025-10-18T19:19:53.784271",
  "hash": "d199a85e510b32b9ef3c02a29044a41d",
  "song": {
    "id": 94173,
    "public_id": 2306,
    "name": "AYE AYEEE",
    "original_key": "AYE AYEEE",
    "category": "released",
    "era": {
      "id": 126,
      "name": "YouTube",
      "description": "YouTube releases grouping",
      "time_frame": "(January 2024-January 2024)",
      "play_count": 32
    },
    "path": "Compilation/1. Released Discography/24. Singles & Features/2. YouTube - Singles & Features/1. 2016 - YouTube Singles & Features/AYE AYEEE.mp3",
    "track_titles": [
      "AYE AYEEE",
      "Ayeee",
      "God Damn"
    ],
    "credited_artists": "JuiceTheKidd",
    "producers": "Dez Wright",
    "engineers": "Unknown",
    "recording_locations": "Homewood Public Library Digital Media Lab Recording Studio, Homewood, Chicago, IL.",
    "record_dates": "File Re-exported\nNovember 17, 2016.",
    "length": "3:22",
    "bitrate": "",
    "additional_information": "Released on YouTube as a single. The Track was originally untitled.",
    "file_names": "My Song 4",
    "instrumentals": "N/A",
    "preview_date": "First Previewed\nJanuary 14, 2016.",
    "release_date": "Released\r\nJanuary 14, 2016.",
    "dates": "",
    "session_titles": "",
    "session_tracking": "",
    "instrumental_names": "",
    "notes": "{\"Project:\": \"YouTube\", \"#:\": 1, \"Project | Track
Title(s):\": \"AYE AYEEE\\nAyeee | God Damn\", \"Credited Artist(s):\":
\"JuiceTheKidd\", \"Producer(s):\": \"Dez Wright\", \"Engineer(s):\": \"Unknown\",
\"Additional Information:\": \"Released on YouTube as a single. The Track was
originally untitled.\", \"File Name(s) | Metadata:\": \"My Song 4\",
\"Instrumental Name(s):\": \"N/A\", \"Recording Location(s):\": \"Homewood Public
Library Digital Media Lab Recording Studio, Homewood, Chicago, IL.\", \"Record
Date(s):\": \"File Re-exported\\nNovember 17, 2016.\", \"Preview Date:\": \"First
Previewed\\nJanuary 14, 2016.\", \"Release Date:\": \"Released\\r\\nJanuary 14,
2016.\", \"Duration:\": \"3:22\", \"Category:\": \"Single Track\"}",
    "lyrics": "",
    "snippets": [],
    "date_leaked": "",
    "leak_type": "Single Track",
    "image_url": "/assets/youtube.webp"
  }
}
```

Data Models

Song Model

```
{
  "id": 1,
  "public_id": 123,
  "name": "Song Title",
  "original_key": "Original JSON Key",
  "category": "released|unreleased|unsurfaced|recording_session",
  "path": "Compilation/folder/song.mp3",
  "era": {
    "id": 1,
    "name": "Era Name",
    "description": "Era Description",
    "time_frame": "Time Period",
    "play_count": 0
  },
  "track_titles": ["Title 1", "Title 2"],
  "credited_artists": "Artist Names",
  "producers": "Producer Names",
  "engineers": "Engineer Names",
  "recording_locations": "Studio Locations",
  "record_dates": "Recording Dates",
  "length": "Duration",
  "bitrate": "Bitrate Info",
  "additional_information": "Extra Info",
  "file_names": "File Name(s)",
  "instrumentals": "Instrumental Names",
  "preview_date": "Preview Date",
  "release_date": "Release Date",
  "dates": "Additional Dates",
  "session_titles": "Session Titles",
  "session_tracking": "Session Tracking",
  "instrumental_names": "Instrumental Names",
  "notes": "Combined Notes",
  "lyrics": "Song Lyrics",
  "snippets": [],
  "date_leaked": "Leak Date",
  "leak_type": "Leak Type",
  "image_url": "Era Image URL"
}
```

Era Model

```
{
  "id": 1,
  "name": "Era Name",
  "description": "Era Description",
  "time_frame": "Time Period"
}
```

Usage Examples

JavaScript (Fetch API):

```
// Fetch songs with filtering
fetch('https://juicewrldapi.com/juicewrld/songs/?category=released&page=1')
  .then(response => response.json())
  .then(data => {
    // Handle songs data
    const songs = data.results;
    const total = data.count;
    displaySongs(songs, total);
  })
  .catch(error => handleError('Failed to fetch songs', error));

// Search songs (normalizes special characters - "dont" matches "don't")
fetch('https://juicewrldapi.com/juicewrld/songs/?search=dont go')
  .then(response => response.json())
  .then(data => console.log('Found songs:', data.results));

// Search all fields including producers
fetch('https://juicewrldapi.com/juicewrld/songs/?searchall=nick mira')
  .then(response => response.json())
  .then(data => console.log('Found songs:', data.results));

// Search by lyrics
fetch('https://juicewrldapi.com/juicewrld/songs/?lyrics=lucid dreams')
```

```
  .then(response => response.json())
  .then(data => console.log('Found songs:', data.results));

// Get random radio song
fetch('https://juicewrldapi.com/juicewrld/radio/random/')
  .then(response => response.json())
  .then(data => {
    console.log('Random song:', data.title);
    console.log('Full metadata:', data.song);
  });

// Get playable files from Compilation
fetch('https://juicewrldapi.com/juicewrld/files/browse/?
path=Compilation&search=.mp3')
  .then(response => response.json())
  .then(data => displayPlayableSongs((data.items||
[]).filter(i=>i.type==='file')));

// Stream audio with range request for seeking
fetch('https://juicewrldapi.com/juicewrld/files/download/?
path=Compilation/song.mp3', {
  headers: {
    'Range': 'bytes=0-1048575'
  }
})
  .then(response => {
    if (response.status === 206) {
      console.log('Partial content received');
      return response.blob();
    }
    return response.blob();
  });
```
Python (Requests):
```
import requests

# Get songs with filtering
response = requests.get(
    'https://juicewrldapi.com/juicewrld/songs/',
    params={
        'category': 'released',
        'search': 'dont go',
        'page': 1
    }
)

if response.status_code == 200:
    data = response.json()
    print(f"Found {data['count']} songs")
    for song in data['results']:
        print(f"- {song['name']}")

# Search all fields including producers
response = requests.get(
    'https://juicewrldapi.com/juicewrld/songs/',
    params={'searchall': 'nick mira'}
)

# Search by lyrics
response = requests.get(
    'https://juicewrldapi.com/juicewrld/songs/',
    params={'lyrics': 'lucid dreams'}
)

# Get random radio song
response = requests.get('https://juicewrldapi.com/juicewrld/radio/random/')
if response.status_code == 200:
    data = response.json()
    print(f"Random song: {data['title']}")
    if data.get('song'):
        print(f"Full metadata: {data['song']['name']}")

# Get playable files from Compilation
playable = requests.get('https://juicewrldapi.com/juicewrld/files/browse/',
params={'path':'Compilation','search':'.mp3'})
if playable.status_code == 200:
    items = playable.json().get('items', [])
    print(f"Playable files: {len(items)}")

# Stream audio with range request for seeking
headers = {'Range': 'bytes=0-1048575'}
audio_response =
requests.get('https://juicewrldapi.com/juicewrld/files/download/', params={'path':
'Compilation/song.mp3'}, headers=headers, stream=True)
if audio_response.status_code == 206:
    print(f"Partial content: {audio_response.headers.get('Content-Range')}")
```
cURL:
```
# Get all songs
curl "https://juicewrldapi.com/juicewrld/songs/"

# Get songs with filtering
curl "https://juicewrldapi.com/juicewrld/songs/?category=released&search=dont go"

# Search all fields including producers
curl "https://juicewrldapi.com/juicewrld/songs/?searchall=nick mira"
```

```
# Search by lyrics
curl "https://juicewrldapi.com/juicewrld/songs/?lyrics=lucid dreams"

# Get specific song
curl "https://juicewrldapi.com/juicewrld/songs/1/"

# Get statistics
curl "https://juicewrldapi.com/juicewrld/stats/"

# Get categories
curl "https://juicewrldapi.com/juicewrld/categories/"

# Get eras
curl "https://juicewrldapi.com/juicewrld/eras/"

# Get random radio song
curl "https://juicewrldapi.com/juicewrld/radio/random/"

# Browse playable files in Compilation
curl "https://juicewrldapi.com/juicewrld/files/browse/?
path=Compilation&search=.mp3"

# Stream audio (save to file)
curl "https://juicewrldapi.com/juicewrld/files/download/?path={file_path}" -o
song.mp3

# Stream audio with range request (for seeking)
curl -H "Range: bytes=0-1048575"
"https://juicewrldapi.com/juicewrld/files/download/?path={file_path}" -o
song_part.mp3
```