Pubhub Web Services

Audio Streaming API Integration Guide

November 2020

v1.0

Table Of Contents

Table Of Contents **2**

Revision History **2**

API Reference **3**

Offline Security Guidelines **5**

Android 5

iOS/iPadOS 5

Revision History

Date	Version	Comment
2020-11-19	1.0	First version of service released.

API Reference

Live service endpoint:
https://audio.api.streaming.pubhub.dk/

QA service endpoint:
https://audio.api.streaming.qa.pubhub.dk/

`GET /v1/stream/hls/{orderId}/playlist.m3u8`

Standard URL used for HLS streaming and download.

This URL is to be used for streaming audio books. Order ID will have to be obtained through the Pubhub Library API or Pubhub Retailer API.
The same URL can be used for download for offline play, however make sure to follow the storage security guidelines described in Offline Security Guidelines.

`GET /v1/stream/hls/{orderId}/playlist.m3u8?offline=true`

HLS playlist with offline://-scheme.

Alternative URL used for HLS streaming, where the encryption key's url scheme is "offline://" instead of "https://".
This can be used to intercept the decryption key request during download and playback, enabling the application to encrypt/decrypt the HLS key.

`GET /v1/stream/hls/{orderId}/playlist.m3u8?offline=true&mediaPlaylist=true`

HLS playlist with offline://-scheme wrapped in a HLS master playlist.

Wraps the above offline://-scheme in a HLS master playlist.
This is required for native iOS 10 playback, as it does not support standard HLS playlists containing encrypted MP3 audio.
This is not needed for iOS 11+.

`GET /Sample.ashx?isbn={isbn}`

Short MP3 audiobook sample.

POST /v2/states/

Synchronize audio stream playback positions.

POST data format:

```
[
  {
    "id": {orderId:string},
    "startUtc": {:datetime},
    "endUtc": {:datetime},
    "startPosition": {seconds:long},
    "endPosition": {seconds:long}
  },
  …
]
```

POST data example:

```
[
  {
    "id": "106d9ba0-125c-4b44-a410-2889d53f42bd",
    "startUtc": "2020-01-15T18:58:53.977Z",
    "endUtc": "2020-01-15T18:58:54.114Z",
    "startPosition": 615,
    "endPosition": 620
  },
  {
    "id": "3e132725-d0cd-4c78-bf0b-960f4f0b2a45",
    "startUtc": "2020-01-15T18:58:55.921Z",
    "endUtc": "2020-01-15T18:58:59.016Z",
    "startPosition": 100,
    "endPosition": 103
  },
  {
    "id": "3e132725-d0cd-4c78-bf0b-960f4f0b2a45",
    "startUtc": "2020-01-15T18:58:59.608Z",
    "endUtc": "2020-01-15T18:59:06.265Z",
    "startPosition": 103,
    "endPosition": 109
  }
]
```

GET /v2/states/latest/{orderId}

Retrieve latest synchronized audio playback position.

Return format:

```
{
  LastUpdated: {:datetime},
  Position: {seconds:long}
}
```

Offline Security Guidelines

For security reasons, when downloading audiobooks it is important that the HLS decryption keys are stored safely and cannot be retrieved by users. Do not store the .key-file together with the .m3u8/.ts files.

We strongly recommend either storing the HLS keys in the iOS Keychain/Android Keystore, or storing the HLS keys in a local database after encrypting them using a private key from the keychain/keystore.

Android

Using Exoplayer's `FileDataSource` for offline storage is not sufficient for the level of security we require. All files, including the key file, are accessible in the device storage, and it is trivial to copy and rename the audiobook's files from this storage.

In order to provide a sufficient level of security, we recommend implementing a custom `DataSource.Factory` using `AesCipherDataSource` and a `DataSink.Factory` using `AesCipherDataSink` to be used by the download manager and playback media source as shown here: https://github.com/google/ExoPlayer/issues/5193#issuecomment-445786954.

Alternatively, for better performance the downloader and media source can be modified so that only the HLS encryption key is stored securely/encrypted.

The keys used by the AES cipher to encrypt the HLS encryption key should be created and stored in the Android keystore system.

iOS/iPadOS

On iOS/iPadOS `AVAssetResourceLoaderDelegate` can be added to `AVURLAsset.resourceLoader` to intercept the HLS key request, where it can be encrypted and stored locally using a key from the iOS Keychain, or stored directly in the keychain.

When using the “`?offline=true`” parameter in the API, as described in the API reference, the key url scheme will be “`offline://`”, and can thus easily be intercepted both in the download delegate and during playback.