For the most recent version of the asset documentation, please visit <u>https://hodge.io</u>
This PDF version is provided for convenience.

Thank you for downloading the iOS Music asset! I hope you enjoy using this software. Please reach out to me if you have any questions.

- Alexander Hodge

Email: **support@hodge.io**

Introduction

iOS Music enables access to the music library on a user's device through various means according to the original source and ownership of the music content. I've abstracted this technical implementation within the native plugin of the asset into three "modes" in which the asset can be deployed and used in your project depending on what you want to achieve and what is important to you. In all modes, metadata is extracted and provided to you via callback functions whenever available. A brief summary of these three modes is provided in the table below. I encourage you to think about and decide which is most appropriate for your project.

	Native player mode	**Audio Source mode**	**Apple Music mode**
Description	A music picker graphical user interface is provided, displaying the entire contents of the local user music library (**NOTE:** this doesn't include music which has been added to the library through the Apple Music streaming service), and the user may select song(s) for playback. Playback of music is handled through a managed instance of a native iOS audio player class.	A music picker graphical user interface is provided, displaying a subset of the user's music library - specifically, music for which there exists a non-DRM and locally stored representation of the content i.e. music that the user truly "owns", and the user may select song(s) for playback. Upon finishing selection, the selected song(s) are automatically cued for playback and individually extracted and converted into a format which can be loaded as an AudioClip and played back within Unity's audio system via an Audio Source. This extraction and conversion process happens asynchronously, but its time depends on the properties of the audio data being extracted ex: sample rate, duration, device specs, etc. Typically, a 3-4 minute song should take a couple of seconds maximum to extract, convert, and begin playback.	A programmatic interface is provided for starting playback of any media item from the Apple Music streaming service using the the media item's Product ID. Product IDs can be found by using iTunes Link Maker available at https://linkmaker.itunes.apple.com/ Make sure to select the appropriate Store Country when using iTunes Link Maker, as a song will have different Product IDs across regions. Ex: 401136641 (The Beatles - Revolver), on the Canadian Store.
Music source	Music content retains its native MPMediaItem representation.	Music content is extracted and converted from its native (MPMediaItem) representation to an intermediate .m4a file saved to the Unity app's persistent data path. This .m4a file is then loaded as an AudioClip using UnityWebRequest APIs.	Music content is streamed using the system (global) media player. Individual items are added to the user's Apple Music library when cued for playback in this mode.
Example use cases	A lightweight and simple to implement method of incorporating a user's music library into your game.	Any form of music visualization, interactive music application, or effects processing implementation where you require access to the music's audio data (samples).	In-game music discovery, game soundtracks using popular music.

A note on the various ways in which music can appear in a device's library, and the iOS Music asset's capabilities with each method:

1. Transferring music from an iTunes Library to an iOS device: The iOS Music asset supports playback in both native and Audio Source modes of all non-protected (DRM) music assets found in the iOS Music Library which have been added from an iTunes Library.
2. iCloud Music Library: The iOS Music asset supports playback in both native and Audio Source modes of music assets added to the device through iCloud Music Library. The music must be added to a user's iCloud Music Library through iTunes, then the user must update their iCloud Music Library with the new content by choosing File > Library > Update iCloud Music Library in iTunes. Finally, the user must download this new content on their device by navigating to it in the iOS Music app and tapping the "Download" icon.
3. Apple Music: The iOS Music asset does NOT currently support any playback of "downloaded" music from the Apple Music service. By default, the asset filters out these items from appearing in the picker interface when making a song selection through the plugin.

It is important that you understand the various ways in which music can be added to an iOS Music Library, and the capabilities associated with each. If you have any questions about specific capabilities that have not been explained in the asset description, please contact me at my publisher email address.

Quick Start - Prefab and example scenes

The iOS Music asset included a prefab of the iOS Music controller which can be added to any scene from which you might want to access the user's music library. An example scene is also provided, including a GUI for interacting with the asset's different modes. Feel free to re-use elements from the example scene in your own game.

API Definitions

These APIs are provided as part of the iOS Music class musicManager.

SetupiOSMusic() - Optional function that you can use to trigger the permissions prompt at any time instead of waiting for the user to interact with the music library.

OpenNativeMusicPicker(bool appendToPlaylist) - Opens the native music picker (MPMediaPickerController) interface with a bool argument to specify if the selection will be appended to the current playlist or not.

PlayPause() - A convenience function for interacting with the native player. If currently playing, calling this function will pause playback. If currently paused, calling this function will resume playback.

LoadAudioClip() - Similar to *OpenNativeMusicPicker()* except upon finishing selection of music, the extraction and conversion process occurs and an AudioClip is generated and played through an Audio Source.

PreviousSong()/NextSong() - Convenience functions for navigating within a playlist.

RandomSongWithNativeAudioPlayer() - Gets a randomly selected song from the user's music library and plays it with the native audio player.

RandomSongWithAudioSource() - Gets a randomly selected song from the user's music library that is compatible with Audio Source mode and plays it.

QueryAppleMusic(string productID) - Query the Apple Music streaming service for a media item corresponding to the Product ID argument. If one is found and the user is an Apple Music subscriber, playback begins immediately.

StopAppleMusic() - Stops playback of all Apple Music streaming content.

Permissions

Important note for customers deploying to iOS 10+:
An iOS app now must request permission to access the device's media library, and the user can approve or deny the request. You can easily add this feature to your app with the Xcode project generated by Unity. The solution is to add an entry in the Xcode project's Information Property List "Privacy - Media Library Usage Description", of type String, and with a message to show to the user explaining why the app needs access. You can find this Property List in the "Info" tab of the Unity-iPhone build target. Without creating this entry, the app will not present the media library to the user and will immediately dismiss any view controller attempting to access it, which results in the plugin's callback functions returning null values.

Playlists

As of version 2.0, iOS Music supports playlist functionality using both the native iOS player and Audio Sources! When a song finishes playing, the plugin will automatically load the next song in the playlist. The iOS Music plugin implements a "lazy loading" system for playback via an Audio Source. When the plugin loads a song via an Audio Source, you can expect to wait a short time (up to a few seconds) while the song is extracted from the device's music library and converted to a suitable format. Playback will begin automatically once the conversion has taken place. Opening the iOS Music library when a song is already playing will pause the currently playing song.

The plugin also supports an "Append to Playlist" mode which will add newly selected songs to the current playlist instead of replacing the current playlist.

Metadata

As of version 1.2, iOS Music will extract metadata for the selected song, including song title, artist, album name, genre, duration (in seconds), lyrics, and BPM. The plugin also extracts and loads the song artwork as a Texture. See the demo scene for an example of how to access the extracted metadata.

Cancelling/dismissing the song picker interface

The asset's main script, iOSMusic.cs, contains a callback function for handling user cancellation of song selection. Use this function UserDidCancel() to handle the cancellation in a manner that's appropriate for your game.

Batch retrieval and playback of local playlists
The asset includes support for retrieving the names of all of the user's local music playlists, and starting continuous playback of a playlist by name. See the Playlist-functionality region within iOSMusic.cs for an example workflow.

Experimental: Custom Music UI
The asset includes experimental support for making your own custom music UI to present local/ non-cloud, music library information in your own game's visual style or theme instead of relying on the native iOS user interface elements in the provided picker interface. ***See the provided example scene.*** This is a basic interface for doing a batch retrieval of music library contents by their persistent IDs, storing them in a managed hashtable, and starting playback later. The retrieved metadata is cached for each unique item, allowing you to present information like artist and song name, etc. This feature is currently experimental, and may change significantly in the future.

Source Code
Full source code is included with the purchase of this asset and modifications are encouraged.