

Quick Start Guide

Native Media Player v.2.0.0

Supported OS : Android / iOS

32 & 64 bit Android 7.0+ (Sdk 26) / 64 bit iOS 11.0+

If you need any help, please contact me at amaiichigopurin@gmail.com

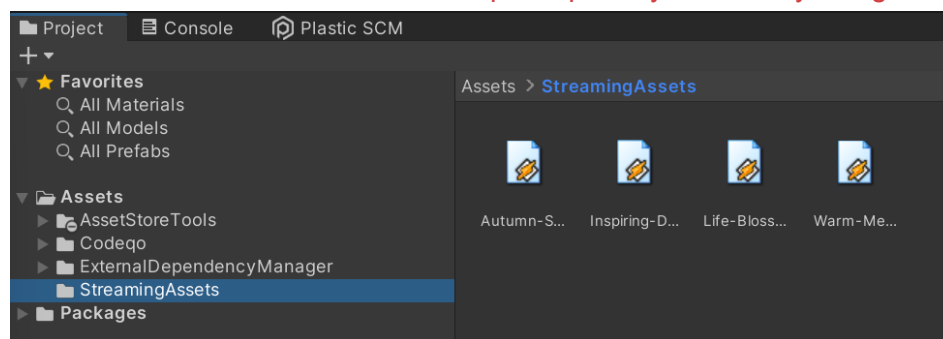
// how to start

It takes less than 10 minutes to import Native Media Player to your project. Here's a step by step guide to jump in. Some of the information can be outdated. Please visit the following online documents for the most updated documentations.

<https://johann-song.gitbook.io/codegos-native-plugins/native-media-player/introduction>

1. Create StreamingAssets folder

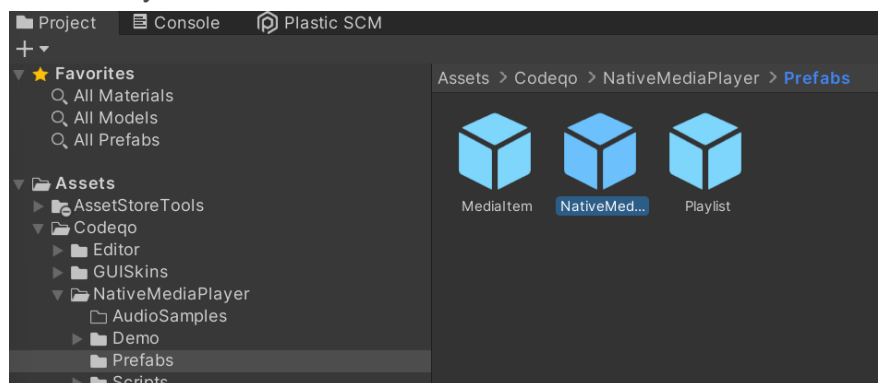
If you are using local audio files, create a new folder named 'StreamingAssets' under Assets. Put your audio files under this folder. **You can skip this part if you are only using remote URLs.**



Make sure the folder name is correct

2. Add NativeMediaPlayer to your scene

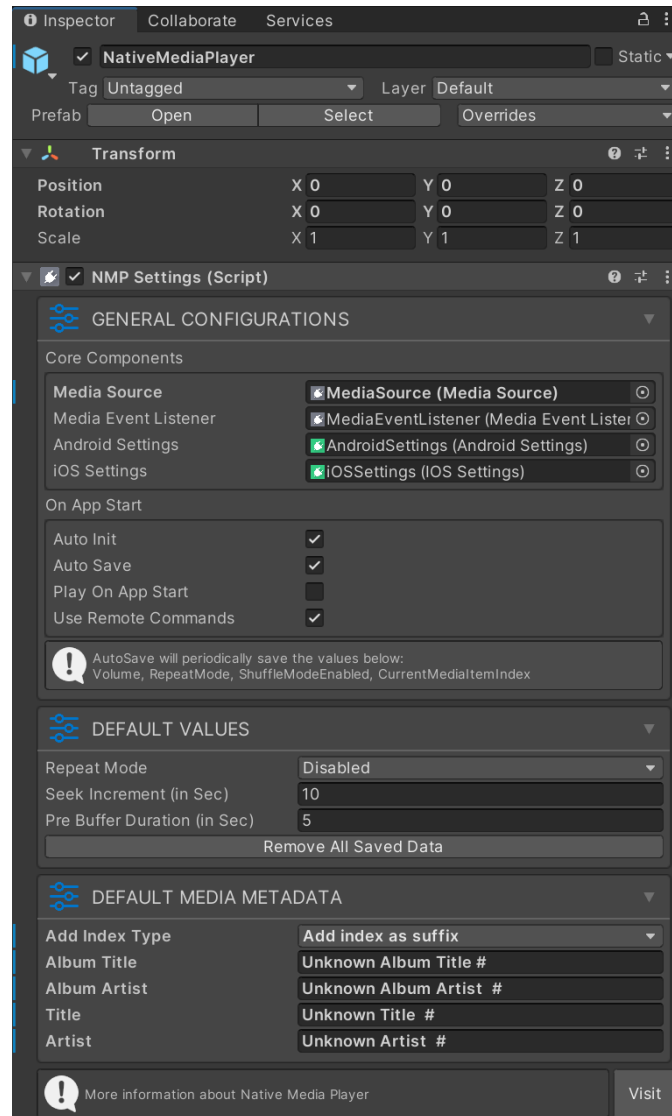
Import [MediaPlayer.prefab](#) to your scene. It's located under Codeqo\NativeMediaPlayer\Prefabs.



Drag it to your hierarchy

3. Configure default settings

Configure NMPSettings using the Editor Inspector.



All default settings can be configured here

4. Configure core components

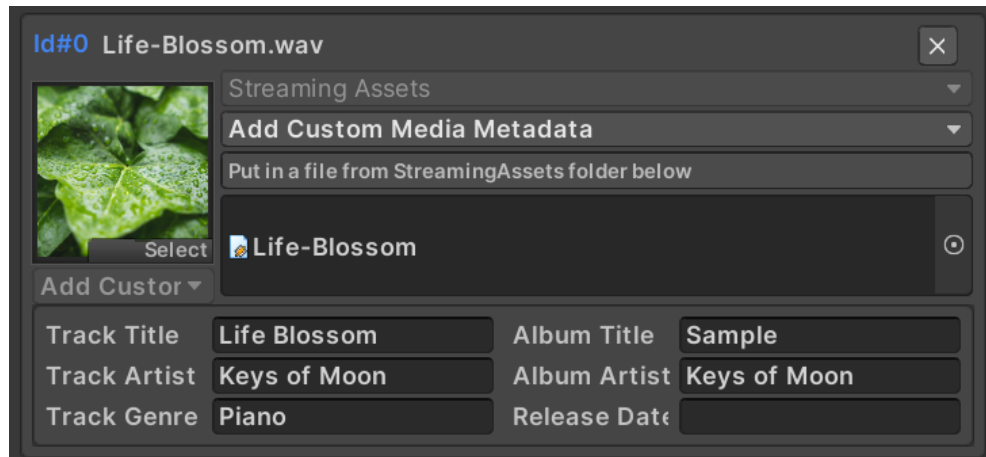
Inside `NMPSettings.prefab`, there are 4 objects each representing 4 major components to run this plugin. Each component is **CRUCIAL**, so take your time and do not skip this step. Configure each inspector's values carefully. Visit online documents for more information.

Online Documents:

<https://johann-song.gitbook.io/codeqos-native-plugins/native-media-player/introduction/quick-start>

5. Add track information

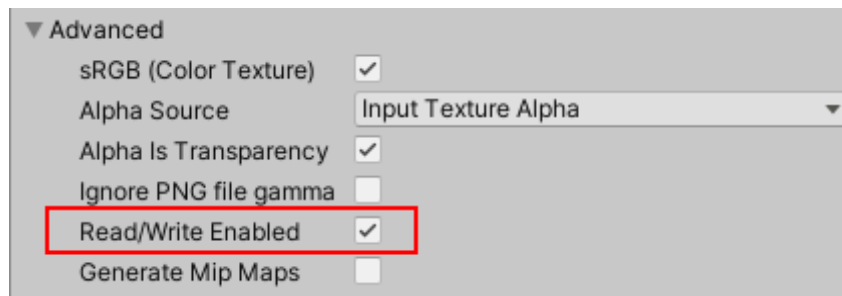
Import a Playlist prefab, multiple Playlist prefabs or a Medialtem prefab to your scene and attach them to the MediaSource. Configure track information(s) on the prefab's Editor Inspector.



Enter track information to Playlist.prefab or Medialtem.prefab

6. Add album arts

When adding an album art image, click the texture and check 'Read/Write Enabled' under Advanced in the inspector.



C# Code Reference

// namespace

☐ `using Codeqo.NativeMediaPlayer;`

// core methods

☐ `MediaPlayer.Init(MediaItems _defaultMediaItems)`

Initiate the plugin with default media items.

💡 You can automatically initiate the plugin with [MediaManager.cs](#)

☐ `MediaPlayer.SetMediaSource(MediaItems _mediaItems)`

Set new media items to the plugin.

💡 Call this whenever you update the media source. This will remove the previously set media items.

// basic player methods

☐ `MediaPlayer.Play()`

☐ `MediaPlayer.Stop()`

☐ `MediaPlayer.Pause()`

☐ `MediaPlayer.Resume()`

☐ `MediaPlayer.Previous()`

☐ `MediaPlayer.Next()`

☐ `MediaPlayer.FastForward()`

☐ `MediaPlayer.Rewind()`

// get and set methods

☐ `MediaPlayer.GetDuration()`

Get the total duration of your current track.

💡 This is mainly used for tracking the track progression on the seek bar.

☐ `MediaPlayer.GetCurrentPosition()`

Get the current position of your current track.

💡 This is mainly used for tracking the track progression on the seek bar.

☐ `MediaPlayer.SetCurrentPosition(float _time)`

Set the current position of your current track.

💡 `SetCurrentPosition(0)` makes the track start again from the beginning.

✔ Use this on your seek bar if you are not using the [MediaManager.cs](#) that comes with this plugin.

// C# public variables

☐ **bool MediaPlayer.isInit (read-only)**

Returns true if the plugin has been initiated.

☐ **bool MediaPlayer.AutoSave**

Set this true if you want the player to automatically save and load following variables.

💡 AutoSave variables

float MediaPlayer.Volume

int MediaPlayer.RepeatMode

bool MediaPlayer.ShuffleModeEnabled

int MediaPlayer.CurrentMediaItemIndex

☐ **int MediaPlayer.PlaybackState (read-only)**

Returns the current playback state.

💡 Enum refs.

int PlaybackState.Idle = 1

int PlaybackState.Buffering = 2

int PlaybackState.Ready = 3

int PlaybackState.Ended = 4

☐ **int MediaPlayer.RepeatMode**

Get or set the repeat mode.

💡 Enum refs.

int RepeatMode.Disabled = 0

int RepeatMode.RepeatOne = 1

int RepeatMode.RepeatAll = 2

☐ **bool MediaPlayer.ShuffleModeEnabled**

Get or set the shuffle mode.

☐ **float MediaPlayer.Volume**

Get or set the player volume.

☐ **MediaItems MediaPlayer.CurrentMediaItems**

Returns currently set media items (on Unity side)

💡 Check [MediaItems.cs](#) and [MediaMetadata.cs](#) in this guide

☐ **int MediaPlayer.GetCurrentMediaItemIndex (read-only)**

Get the current media item's index number.

☐ **int[] MediaPlayer.GetShuffleOrder (read-only)**



Get the current shuffle order.

- ☐ `string MediaPlayer.Error`
Returns the latest error message.





// player state variables (Read-Only)

- ☐ `bool MediaPlayer.isPlaying`
Returns true if the player is playing.
- ☐ `bool MediaPlayer.isLoading`
Returns true if the player is loading/reading your media source.

// Medialtems.prefab

- ☐ `Medialtems.cs`
MonoBehaviour script that manages multiple media items.
 Use the prefab and read the descriptions in the inspector.
 You will need to add at least one of this to your scene unless you are creating Medialtems with codes.

// Medialtem variables

- ☐ `MediaLocation MediaLocation`
Select an enum to indicate where your sources are located.
 Use the following enums to indicate the data path where your media sources are located.
`short MediaLocation.StreamingAssets = 0`
`short MediaLocation.RemoteURL = 1`
- ☐ `string Medialtem[int index].Title`
Title of the media item
 Related parameters
METADATA_KEY_TITLE (Android), MPMedialtemPropertyTitle (iOS)
- ☐ `string Medialtem[int index].Artist`
Artist of the media item
 Related parameters
METADATA_KEY_ARTIST (Android), MPMedialtemPropertyArtist (iOS)
- ☐ `string Medialtem[int index].AlbumTitle`
Album title of the media item
 Related parameters
METADATA_KEY_ALBUM (Android), MPMedialtemPropertyAlbumTitle (iOS)

- ☐ **string** `Medialtem[int index].AlbumArtist`
 Album artist of the media item
 💡 Related parameters
 METADATA_KEY_ALBUM_ARTIST (Android), MPMedialtemPropertyAlbumArtist (iOS)
- ☐ **string** `Medialtem[int index].Genre`
 Title of the media item
 💡 Related parameters
 METADATA_KEY_GENRE (Android), MPMedialtemPropertyGenre (iOS)
- ☐ **string** `Medialtem[int index].ReleaseDate`
 Released date of the media item
 💡 Related parameters
 METADATA_KEY_DATE (Android), MPMedialtemPropertyReleaseDate (iOS)
- ☐ **Sprite** `Medialtem[int index].Art`
 Artwork of the media item
 💡 Related parameters
 METADATA_KEY_ART (Android), MPMedialtemPropertyArtwork (iOS)
- ☐ **string** `Medialtem[int index].StreamingAssetName`
 Filename of the media item under StreamingAssets folder
 💡 Only visible in the inspector if MediaLocation is set to StreamingAssets
- ☐ **string** `Medialtem[int index].RemoteURL`
 Remote URL of the media item
 💡 Only visible in the inspector if MediaLocation is set to RemoteURL
- ☐ **bool** `Medialtem[int index].CustomMediaMetadataEnabled`
 Set it true if you want to enter your own custom media metadata
 💡 You can easily edit the media metadata in the inspector
 💡 If this is **false**, the plugin will try to retrieve the metadata on the native side.

// MediaPlayer.prefab

- ☐ **MediaPlayerManager.cs**
 MonoBehaviour script that manages general plugin behaviors.
 💡 Use the prefab and read the descriptions in the inspector.
 💡 Add Volume slider and Seek Bar slider to this for easy UI management.
 ⚠️ You must add one of this to your scene for plugin to work properly.

// RemoteActions.prefab

☐ RemoteActions.cs

MonoBehaviour script that manages native remote player actions.

💡 This prefab is included in the [MediaManager.prefab](#)

// MediaEventListener.prefab

☐ MediaEventListener.cs

MonoBehaviour script that receives callbacks from the native plugin.

💡 This prefab is included in the [MediaManager.prefab](#)

⚠️ Don't change the name of this prefab object.

// AndroidConfigurations.prefab

☐ AndroidConfigurations.cs

MonoBehaviour script that manages Android notification settings.

💡 This prefab is included in the [MediaManager.prefab](#)

This plugin uses following APIs

1. Android MediaPlayer (v1.5.0) / StreamingAssets
Supported Formats : <https://developer.android.com/guide/topics/media/media-formats>
2. Android ExoPlayer (v.2.17.0) / Remote URL
Supported Formats : <https://exoplayer.dev/supported-formats.html>
3. iOS AVAudioPlayer / StreamingAssets
Supported Formats : No information available
4. iOS AVPlayer / Remote URL
Supported Formats : No information available