README

Google Cast Remote Display Unity Plugin EAP

November 6th 2015 Release

brought to you by the luigi-eng team and friends

**Important:** This is an early access partner release. The plugin and its content is subject to change before public release. Also we welcome your feedback to fixes and improvements (email James Grafton - [jrgrafton@google.com](mailto:jrgrafton@google.com) or [googletv-partner-remote-display-unity-eap@googlegroups.com](mailto:googletv-partner-remote-display-unity-eap@googlegroups.com))!

# What’s New (November 6th, 2015)

* **Breaking Changes:**
  + GetSelectedCastDeviceId has been removed from CastRemoteDisplayManager. A new function has been created in CastRemoteDisplayManager, called GetSelectedCastDevice. The CastDevice class has read-only information on the device that is currently being cast to, including device ID, name, and status.
  + CastRemoteDisplayAudioListener.cs is deprecated in Unity 5.3+ and moved to **Assets/Plugins/GoogleCastRemoteDisplay/Deprecated.** You no longer need to use this in Unity 5.3:
    - Do not use if you are developing with Unity 5.3+
    - This component is required because Unity had a bug on iOS platforms where OnAudioFilterRead was not being called unless the component was present at scene initialization time.
    - In Unity 5.3+, this bug was fixed and the CastRemoteDisplayManager will attach the required component at runtime. Simply set the RemoteAudioListener property to any AudioListener.
    - If you are still on Unity < 5.3, you will need to use this component, and re-attach it to your audio listener, as it has a new RequireComponent() that only triggers if you add it again.
* **General Updates:**
  + Fixes to showing correct set of Cast devices after foregrounding.
  + More improvements and polish to Default UI.
  + **Note**: If the Simulator is not displaying in the Unity Editor, try switching your target platform to Android.
* **Android Specific Updates:**
  + Plugin merged into a single aar file
  + Better logic for handling device discovery.
  + Plugin now requires Google Play Services 8.3.0+, which is now available for all devices and fixes numerous issues. See [these release notes](https://developers.google.com/android/guides/releases) and [video](https://www.youtube.com/watch?v=nAUeEJ51Cko) for details. Please make sure to use an up to date version of google play services when building the project (Included in AndroidDependencies.zip for your convinience)
    - Fixes to Cast notification dialog
    - Fixes persistent notification after the app is terminated while casting
    - Stability fixes
* **iOS Specific Updates:**
  + Core Cast SDK now updated to use latest fixes and improvements from [Cast SDK Release 2.10](https://developers.google.com/cast/docs/release-notes).
  + Disabling Adaptive Bitrate” now fixes the bitrate down to 3 Mbps instead of 5 Mbps.
  + Fixes to video/color corruption and image quality.

# What’s New (October 15th, 2015)

* **Breaking Changes:**
  + All remote display APIs are moved under namespaces: Google.Cast.RemoteDisplay and Google.Cast.RemoteDisplay.UI
  + All CastRemoteDisplayManagerEvents have been capitalized. e.g. castDevicesUpdatedEvent -> **C**astDevicesUpdatedEvent
  + Video bitrate and audio bitrate are removed from CastRemoteDisplayConfiguration. See “Disable Adaptive Bitrate” below under “iOS Specific Updates”.
* **General Updates:**
  + Added IsCasting method to CastRemoteDisplayManager, which is true from the moment RemoteDisplaySessionStartEvent fires and until the session ends.
  + The first time cast dialog UI is optional and can be configured by a checkbox in the CastDefaultUI inspector
  + More refinements to the default cast UI
* **Android Specific Updates:**
  + Rendering to a remote display should be less CPU and GPU intensive in general, especially when the game is paused or backgrounded or not casting.
* **iOS Specific Updates:**
  + Added a “High” TargetDelay option under CastRemoteDisplayConfiguration. Selecting “High” gives more time to encode, buffer, and send graphics to the TV, which may improve image quality. But this also increases latency between user input and seeing results on the TV. This might be useful for games that do not require fast reflexes or non-interactive content like cutscenes.
  + Added a “Disable Adaptive Bitrate” toggle under CastRemoteDisplayConfiguration. By default this this is turned off. If turned on, this disables adaptive bitrate (automatically adjusting bitrate on the fly depending on network congestion) and forces the bitrate to be always 5 Mbps.
  + GoogleCastRemoteDisplay-0.0.0-NoAdaptiveBitrate.a is no longer included. Instead use “Disable Adaptive Bitrate” toggle as described above.
  + Backgrounding will not kill the remote display session within 15 seconds - but eventually iOS will kill the session.
  + Rendering should no longer overheat mobile device when the game is paused, backgrounded, or not casting.

# What’s New (October 1st, 2015)

* **Breaking Change:**
  + Changed the type of the events published by the CastRemoteDisplayManager from C# events to **UnityEvents**, which are serializable and can be displayed on the inspector. The difference is that you call **AddListener**() and **RemoveListener**() on them instead of += and -=
  + Changed the required signature of the callback for **remoteDisplayErrorEvent** to be consistent with the other callbacks (they all take a single argument of type CastRemoteDisplayManager). Added a new API GetLastError() which returns an error object containing the information needed to present an error dialog.
  + Renamed RemoteDisplayRenderTexture -> **RemoteDisplayTexture**. This is because this and the new **RemoteDisplayPausedTexture** support regular textures now (useful to implement a static loading screen, for example).
  + All RemoteDisplay prefabs now enforce uniqueness (only one object of each type). CastRemoteDisplayManager, CastRemoteDisplaySimulator, and DefaultCastUIController will persist themselves across multiple scenes. The gameObjects will persist across scenes, but their configuration may need to be updated, depending on game logic (most notably the camera.) Existing implementations that persist the manager should also remove this logic. (Note: This is not a breaking change for all partners, but may be for some.).
* **General Updates:**
  + Added RemoteDisplayPausedTexture property - which allows the game to automatically show a pause screen on the TV when the game is paused or backgrounded.
  + Improvements to the default UI
  + Plugin is working properly with Unity 5.2
  + Added CastError class
  + Added GetLastError() API
* **iOS Specific Updates:**
  + **Looking for feedback:** Added experimental support for adaptive bitrate - which will automatically adjust the bitrate on the fly depending on network congestion. This is automatically enabled and you do not have to configure anything. Note that lower-end iOS devices could render “frozen” frames (let us know!). If you need to remove this feature, see “Files in this Release” below.
  + Backgrounding will no longer stop the remote display session and tear down the graphics on the TV (until iOS automatically kills the connection in 15 seconds).
  + Foregrounding will now automatically restore the remote display session.
  + Changing the resolution in CastRemoteDisplayConfiguration to 480p with Metal is fixed - but will produce color artifacts (work still in progress).

# What’s New (September 18, 2015)

* **Breaking changes:**
  + CastRemoteDisplayManager.RemoteDisplayWidth and RemoteDisplayHeight are moved to CastRemoteDisplayConfiguration (see below)
* **General updates**
  + Added the CastRemoteDisplaySimulator - which lets you simulate remote display while running your game in the Unity Editor (see below)
  + Added CastRemoteDisplayConfiguration - which lets you configure preset TV resolutions and other settings - which are largely iOS specific (see below).
  + The code implementing the DefaultUI is now publicly accessible in the GoogleCastRemoteDisplay/UI/ subfolder.
  + The names and outlets used in the DefaultUI have been updated.
* **Android specific updates**
  + CastRemoteDisplayConfiguration supports customizable resolutions if using a remote display camera. Custom render textures will ignore this setting.
  + OpenGL ES3 is now supported on Android
  + JNI crash when there are too many cast devices is now fixed
  + Expanded support for more devices (devices that support EGL10+, etc)
* **iOS specific updates**
  + CastRemoteDisplayConfiguration supports customizable video and audio bitrates, framerates, and target delay settings.
  + **Known issue:** iOS has a different experience for backgrounding, suspending, etc. This will be addressed in a future release (hopefully the next one!).
  + **Known issue:** Changing the resolution in CastRemoteDisplayConfiguration to 480p with Metal is not supported and will crash the game.

# What’s New (September 4, 2015)

* **Breaking change:** AutoCast is removed. This is replaced by DefaultUI (see below).
* **Breaking change:** ICastRemoteDisplayConnectivityListener removed. The only way to listen for events is to now use these events from the August 21 release:
  + CastRemoteDisplayManager.castDevicesUpdatedEvent
  + CastRemoteDisplayManager.remoteDisplaySessionStartEvent
  + CastRemoteDisplayManager.remoteDisplaySessionEndEvent
* **Known issue:** on iOS, using the new DefaultUI to reconnect after backgrounding brings up a black screen. This will be addressed in the next release.
* **Known issue:**iOS and Android have different experiences for backgrounding, suspending, etc. These will be addressed in the next release.
* **Known issue:** OpenGLES2 is only supported on Android. OpenGLES3 can crash your device. See Android setup steps below to ensure OpenGLES2 is configured.
* Added a DefaultUI (replaces AutoCast), a basic GUI for selecting cast devices. This is the initial release and will change over time to give an example of how to implement UX compliance.
* New event, remoteDisplay**Error**Event, for errors such as unsupported platforms (e.g. iOS version below 8, Android version below KitKat) and for unsupported Google Play Services. Errors will disable the game object owning the CastRemoteDisplayManager component.
* iOS will no longer automatically reconnect when returning from the background regardless if the user disconnected or not before backgrounding (which was causing a black screen error after resuming). Instead, game logic should decide what to do (e.g. save the deviceID and enable/disable the plugin) using Unity’s [OnApplicationPause](http://docs.unity3d.com/ScriptReference/MonoBehaviour.OnApplicationPause.html).
* Minor bug fixes and stability improvements.

# What’s New (August 21, 2015)

* Added support for RenderTextures, which enables composite rendering. No longer need to use a camera for remote display. The CastRemoteDisplayManager will use a camera if set, otherwise will pick up the provided render texture.
* Can now switch remote display cameras and render textures dynamically while the game is running.
* New events that can be used instead of specifying a connectivity listener. The CastRemoteDisplayManager now exposes the following events:
  + castDevicesUpdatedEvent
  + remoteDisplaySessionStartEvent
  + remoteDisplaySessionEndEvent
* iOS will no longer collide with other plugins due to duplicated symbol exports.
* iOS now includes an optional static debug library, with extra logging (see “Files in this Release” below).
* iOS no longer requires the developer to set the audio sample rate manually.
* iOS the metal textures no longer need to be manually flipped upside down.
* Android manifest merging is now supported.
* Android: We no longer require the AndroidManifest to use our own activity. This improves interoperability with other plugins that do require custom activities. However, this exposes a bug. See below.

# Known issues (August 21, 2015)

* Android: If an activity other than the default one provided by this plugin is used, closing the app from the “Recent app drawer” in android will not properly terminate the remote display session and the android notification will stay on the screen. The only solution is to re-launch the app twice. This is due to a bug in Unity, which exposed a bug in Android. We are working with both teams to get this resolved. We have implemented a workaround that will go live to all Android devices at the end of September which completely removes the effects of this bug by killing the notification and terminating the remote display session, although logs will still indicate a service is not being properly killed. For development purposes, if you must use a different activity, we recommend terminating the remote display session before closing the app.

# What’s New (August 4, 2015)

# Added CastDevice struct, used by CastRemoteDisplayManager to get a list of devices and select devices - see “Unity Scripting API” below

* AutoCast will only work on the initial connection - see “Setup - Unity” below.

# Android bug fixes including errors with StopRemoteDisplaySession and disconnecting the app

# iOS no longer requires calling CastRemoteDisplayManager.Start after calling StopRemoteDisplaySession

* iOS bug fixes including enabling/disabling the remote display camera correctly when enabling/disabling the CastRemoteDisplayManager

# What’s New (July 23, 2015)

* Significant changes to how to set up in Unity - see “Setup - Unity” below
* Significant changes to how CastRemoteDisplayManager sends event callbacks - see “Unity Scripting API” below
* Added CastRemoteDisplayManager.StopRemoteDisplaySession method
* For iOS, you should **not** disable all audio listeners in the scene, and you should not add an audio listener on the CastRemoteDisplayManager. See how audio is set up in “Setup - Unity” below.
* Android bug fixes including fixing launching on devices with EGL errors
* iOS bug fixes including fixing build errors for arm7/64 architecture settings

# What’s New (July 10, 2015)

* Initial release. Expect basic functionality, and lots of bugs. The purpose of this release is to act as a proof of concept of integration with the Unity development environment and deployment on Android and iOS devices.
* iOS + Metal flips the remote screen upside down (Open GL does not do this). This problem is described [here in the Unity docs](http://docs.unity3d.com/Manual/SL-PlatformDifferences.html). We will investigate automatically flipping the remote display for Metal. In the meantime, you can manually flip your remote display camera [by adding code to OnPreCull, OnPreRender, and OnPostRender](http://docs.unity3d.com/ScriptReference/MonoBehaviour.OnPreCull.html).

# Requirements

* Unity 5 (You can try Unity 4.6 at your own risk)
* Cast capable device (e.g. Chromecast, Android TV)
* Android device with KitKat+
* iOS device with iOS 8+

# Before you start

1. Although the Unity Plugin hides a lot of the details of the Google Cast Remote Display API, it is suggested you review [the video and documentation](https://developers.google.com/cast/docs/remote).
2. **You must set up an application ID** as described [here](https://developers.google.com/cast/docs/remote#registration). You can use the same application ID for your Android and iOS app.
3. Make sure your Unity development machine already has the appropriate Android and iOS development tools installed (e.g. Android SDK, XCode, etc).

# Files in this release

* **README**  
  this file
* **google\_cast\_remote\_display\_plugin.unityPackage**the Google Cast Remote Display plugin for Unity
* **AndroidDependencies.zip**Android dependencies that you will need to add to your project (see Android setup steps below)
* **AnrdoidManifest.xml Template**  
  An example of a top level AndroidManifest file (see Android setup steps below)
* **GoogleCastUnityRemoteDisplay-0.0.0-Debug.a**The debug version of the iOS unity remote display native library. Replace **GoogleCastUnityRemoteDisplay-0.0.0-Release.a** in Assets/Plugins/iOS/google-cast-remote-display\_lib with this file if you want to get additional logging information (e.g. adding detailed log data for crashes)

# Setup

## **Unity**

1. Open a scene in Unity
2. Select “Assets > Import Package > Custom Package”
3. Choose **google\_cast\_remote\_display\_plugin.unityPackage**
4. Verify that you see files were added under these project folders:
   1. Assets / Plugins / GoogleCastRemoteDisplay
   2. Assets / Plugins / Android / google-cast-remote-display\_lib
   3. Assets / Plugins / iOS / google-cast-remote-display\_lib
5. Add the **CastRemoteDisplayManager.prefab** from Assets / Plugins / GoogleCastRemoteDisplay to your scene hierarchy.   
   **This must be at the top level of your scene hierarchy.**
6. Make sure your scene has the following:
   1. A **main camera** for showing graphics on the phone
   2. A **remote display camera** for showing graphics on the TV
   3. A game object with an enabled **audio listener** - this can be any camera
7. **[EDITED Nov 6] Skip if you are using Unity 5.3+, this is deprecated in Unity 5.3+:**  
   Select the game object with the enabled **audio listener** and add the script component, **CastRemoteDisplayAudioListener.cs** from Assets/Plugins/GoogleCastRemoteDisplay. This audio listener will be used to send audio to the TV.   
   **Note**: You should only have one enabled audio listener in your scene at a time. Multiple enabled audio listeners will cause audio problems.
8. Select the CastRemoteDisplayManager in your hierarchy and update these fields:
   1. For **Remote Display Camera**, select the remote display camera in your scene that you want to show graphics on the TV (you do not need to specify a main camera - but the main camera should NOT be the same as the remote display camera)
   2. For **Remote Audio Listener,** select the game object with **CastRemoteDisplayAudioListener.cs** that you set up in step #7.
   3. For **Cast App Id**, enter [the remote display application ID that you registered](https://developers.google.com/cast/docs/remote#registration).
   4. If you want to display a static texture while the game is backgrounded on the phone or tablet, set a texture to the **RemoteDisplayPausedTexture** field.
   5. For **Configuration**, you can use the defaults.
      1. Resolution affects the texture used by **Remote Display Camera**
      2. Frame Rate, Target Delay, Disable Adaptive Bitrate are only supported on iOS
9. If you want try testing your game right away with a simple pre-built UI to select a cast device, follow these steps:
   1. Add the **DefaultCastUIController.prefab** from Assets / Plugins / GoogleCastRemoteDisplay to your scene hierarchy.   
      **This must be at the top level of your scene hierarchy.**
   2. Update the **displayManager** field with the CastRemoteDisplayManager in your scene.
   3. Verify that your scene hierarchy has a Unity [EventSystem](http://docs.unity3d.com/ScriptReference/EventSystems.EventSystem.html), otherwise the UI will not work.
10. If you want to try simulating remote display within the Unity editor, follow these steps (it’s easier to use the pre-built UI from step 9 beforehand, but not required):
    1. Add the **CastRemoteDisplaySimulator.prefab** from Assets / Plugins / GoogleCastRemoteDisplay to your scene hierarchy.  
       **This must be at the top level of your scene hierachy.**
    2. Update the **Cast Devices** section with the CastRemoteDisplaySimulator in your scene (e.g. change Size to “1” and fill in dummy values for Device ID, Device Name, and Status.
    3. Press the play button to run your scene within Unity and then press the Cast button on the top right corner (or make the appropriate call on the CastRemoteDisplayManager) to start simulating remote display.
11. Follow the Android specific steps for building and deployment on Android (there are no additional steps for iOS) those devices.

## **Android**

The Android portion of the plugin depends on these Android libraries:

* Media Router
* Google Play Services **[EDITED Nov 6th]** 8.3+
* App Compat

Normally, you can get the latest versions of these dependencies from your Android SDK installation folder. But we’ve included a copy of these for EAP in AndroidDependencies.zip for your convenience.

1. Place these dependencies in Assets / Plugins / Android. This folder should now have these files:
   1. **[EDITED Nov 6th]** cast\_remote\_display\_unity3d\_lib.aar
   2. appcompat
   3. google-play-services\_lib
   4. mediarouter
2. If you don’t have a top level Android Manifest, you can use the **AndroidManifest.xml Template file** included in this drop. Place it in Assets / Plugins / Android and name it AndroidManifest.xml. Don’t forget to update your package name.
3. Remote display requires Api level 19, but your App can still set a lower API level if it doesn’t require cast functionality to work.
4. Copy the **Bundle Identifier** string under “Identification” (e.g. com.Unity3D.StarTrooper)

# Unity Scripting API

The Unity Scripting API for the Google Cast Remote Display is still under development. Future EAP releases will provide more details and expand the API.

There are two key classes, found under Assets / Plugins / GoogleCastRemoteDisplay:

**Google.Cast.RemoteDisplay.CastRemoteDisplayManager**

The entry point in Unity for interacting with the Android and iOS native implementations of Google Cast Remote Display. Behaves as a singleton and there should only be one. Important methods include:

* **GetInstance:**  Returns the instance of the class, or null if no object has been created.
* **RemoteDisplayCamera:**  The Camera that should be displayed in the remote display. Can be set at runtime. Ignored if RemoteDisplayTexture is set.
* **RemoteDisplayTexture:** The texture that should be displayed in the remote display. Can be set at runtime.
* **RemoteDisplayPausedTexture:** The texture that should be displayed in the remote display if the game gets backgrounded on the phone or tablet. Can be set at runtime.
* **GetCastDevices**: this returns a list of **CastDevice** structs, which includes the:
  + **deviceId**: use this to select cast devices using the scripting API
  + **deviceName**: the name of the cast device to show in your GUI
  + **status**: the current status text of the cast device to show in your GUI
* **SelectCastDevice**: this should only be called after the OnCastDevicesUpdated method is called in the connectivity listener. Use the **deviceId** field of the **CastDevice** struct from **GetCastDevices**.
* **[EDITED Nov 6th] GetSelectedCastDevice:**  Returns the device we are currently casting to, or null if no cast session is currently active.
* **IsCasting** : Whether there is an active cast session. This will be set to true from the moment the **RemoteDisplaySessionStartEvent** fires and until the session ends.
* **StopRemoteDisplaySession:** Stops casting.
* **GetLastError:** Returns the last error encountered by the plugin, or null if no error has occurred.
* **Published events:**
  + **CastDevicesUpdatedEvent:** fired when the list of available cast devices has been updated
  + **RemoteDisplaySessionStartEvent:**  fired when a cast session starts
  + **RemoteDisplaySessionEndEvent:** fired when a cast session ends
  + **RemoteDisplayErrorEvent:** fired when an error is encountered  
      
    All event listeners must take a parameter of type **CastRemoteDisplayManager** and return **void.**

**Google.Cast.RemoteDisplay.UI.DefaultCastUIController (under UI / Scripts)**

The entry point for an example of a UI for selecting cast devices. This is not meant to be an API, but an example you can use for testing purposes, or use as a reference to implement your own custom game UI.

Expect this UI code to change over time.