GOOGLE PLAY

View online: https://www.construct.net/en/make-games/manuals/construct-3/plugin-reference/google-play

The **Google Play** plugin allows you to integrate your game with **Google Play Game Services**. Users can log in, submit and view scores, and unlock achievements.

Setting up Google Play Game Services

Using Google Play Game Services requires that you have an account registered with the Google Play Developer service. There may be a small one-time registration fee if you need to sign up.

For each game that you wish to use Google Play Game Services for, find the **Play Games Services** section in the developer console and follow the steps to set it up.

Once added you can edit the game details such as its description and any associated images. You can also add achievements and leaderboards - which must be added before you can use them in the Google Play plugin - as well as configure testing and publishing.

To ensure your finished game has permission to access the Google Play Game Services, you need to link your "app" to your "game". Click the *Linked apps* section while editing the game in the developer console, the following steps vary depending on if you are publishing as a web app or a android app.

Publishing as a Web app

Click the button to link a Web app. You will need to fill out details such as the URL it is played from and some authorisation details. Once completed, you should be given a client ID. This should be in a format similar to:

12345678987-abcdefghijklmnopqrstuvwxyz1234567890.apps.googleusercontent.com

Copy and paste this client ID in to the Google Play plugin's *Client ID* property. It is not actually necessary to fill in the *Application ID* property, but you can add it anyway: the application ID is the number that appears beside the game name in the header of the Developer Console, e.g. "My Super Game - 12345678". The application ID in this case is 12345678.

Publishing as an Android app

Click the button to link an Android app. You will need to select an application that you have already registered with Google Play and provide the SHA-1 signature for the certificate you will use to sign your application. It is important that you have provided the signature and that when you are testing that you sign your application with the correct certificate. Your application will be unable to login unless you do this!

Once completed, you should be given a client ID and application ID. For an Android app you **only need you application ID**. This should be in a format similar to:

12345678987

Copy and paste this application ID in to the Google Play plugin's *Game ID* property.

Enabling access in preview mode

When testing your project, your game will run from *https://preview.construct.net*. Since this is is different to the published URL of your game, Google Play Game Services will block access unless you add the preview URL as a permitted location.

To do this, in the Developer Console under *Game details*, find the header that says *API CONSOLE PROJECT* and follow the link to the API console project for your game. In the API console, select *APIs & auth*, then *Credentials*. Notice the *Javascript Origins* field contains only your final URL; this also needs to contain any preview URLs to be allowed access. Click **Edit settings**. Under *Authorized Javascript Origins*, make sure the final URL and the preview URL appear on separate lines, e.g.:

mywebsite.com

preview.construct.net

Click **Update**. Now your game should be authorised to be accessed from preview mode as well as when you publish it.

> *Note that an origin is the scheme and domain only, excluding any path or filenames.*

Adding a Test User

When working with an unpublished Android application you will only be able to log in with the registered test users for that application. To add a test user click the *Testing* section while editing the game in the developer console, click **Add Testers** and type in the email address associated with the Google Account registered on your testing device.

Basic usage

On startup, the plugin automatically attempts to load Google Play Services. If this is successful, then *Is loaded* will be true when the project starts running. If it failed then *Is loaded* will be false and none of the plugin's features will work.

Next the Google Play plugin will automatically try to sign the user in. If they have successfully signed in before, *On signed in* will trigger soon after startup. If they have never signed in before, *On auto-sign in failed* will trigger; in this event you should show a button that allows the user to sign in. When they click this button, use the *Sign in* action; if it succeeds, *On signed in* will be triggered.

Once the user is signed in, you can make use of the other plugin features such as requesting leaderboards, submitting high-scores, and unlocking achievements.

Asynchronous actions

Most actions in the Google Play plugin are asynchronous. This means they are not completed immediately. Instead, the action starts a request which is sent off to the Google Play servers. A few moments later the server will respond, and a corresponding trigger will run in the Google Play plugin.

For example, the *Request player details* action does not complete immediately. The player details are not available until *On player details* received triggers, which should happen shortly after running the action. Only after the trigger runs can the player details be accessed.

Non-immediate requests and caching (Android only)

Google Play Games Services offers **Immediate** and **Non-immediate** versions of most methods. Immediate methods will attempt to update the user's achievements / leaderboard information on the server immediately, whereas non-immediate methods will only update the local Play Games application on the next server sync. In line with official advice we use non-immediate methods where available. To the users eyes it will behave nearly exactly the same, but during testing you may experience some issues with caching. For instance, if you are adding new leaderboards and achievements they may take up to an hour to appear on your device. This doesn't mean it will take an hour for a hour to unlock an achievement though! Local changes appear within the Play Games app immediately, even if it isn't synced with the server.

Google Play properties

Application ID

This is not currently necessary, but can be filled out with the application ID from the Google Play Developer Console.

Client ID

The client ID for the game from the Google Play Developer Console. This is only required for web applications. For more information see the section *Setting up Google Play Game Services* above.

Game ID

The Game ID is only used for Android applications. It is the same as the application ID. For more information see the section *Setting up Google Play Game Services* above.

Google Play conditions

Compare achievement state

Compare whether an achievement at an index is revealed, hidden or unlocked. The achievement list must have already been successfully received.

On achievement list success

On achievement list fail

Triggered after the *List achievements* action, depending on whether the request succeeded or failed. If successful, achievement information for the current player is then available.

On achievement metadata success

On achievement metadata fail

Triggered after the *Get metadata achievements* action, depending on whether the request succeeded or failed. If successful, achievement metadata (such as the achievement names and icons) is then available.

On achievement revealed

On achievement unlocked

Triggered after the *Reveal*, *Unlock* or *Increment* actions when an achievement has successfully been revealed or unlocked. When incrementing achievements, the achievement is unlocked when it has incremented through every step.

Is loaded

True if the Google Play plugin has loaded and is ready to use. If false, no features of the plugin will work.

Is signed in

True if the user has been successfully signed in (possibly automatically).

On auto-sign in failed

Triggered upon the first visit, when the user cannot be automatically logged in. It is necessary to display a 'Sign in' button and use the *Sign in* action to get the user to sign in.

On sign in failed

Triggered when an attempt to sign in fails, either due to an error or because the user cancelled the attempt.

On error

Triggered if an error occurs. The *ErrorMessage* expression will contain information about the error.

On player details received

Triggered after the *Request player details* action, when the player details have been successfully received. The *Player details* category of expressions now are set to their correct values for the currently signed in user.

On signed in

On signed out

Triggers when the user is signed in or signed out from Google Play Game Services.

On hi-score request success

On hi-score request fail

Triggered after the *Request hi-scores* action depending on whether the request succeeded or failed. If successful, the hi-scores list is then available.

On score submit success

On score submit fail

Triggered after the *Submit score* action, depending on whether the submission succeeded or failed. If successful the score should then appear in hi-score lists.

Google Play actions

Get metadata

Request metadata for the achievements list, such as the achievement names, descriptions and icons. If successful, *On achievement metadata* success is triggered.

Increment

Add to, or set, the number of steps in an incremental achievement. Once the full number of steps has been reached, the achievement is automatically unlocked.

List achievements

List the achievements for the currently signed in player. Optionally the list of achievements can be filtered to only those in a given state (e.g. revealed). If successful, *On achievement list success* triggers.

Show achievements (Android only)

Shows the native achievements dialog for the currently signed in player.

Reveal

If an achievement is hidden, set its state to 'revealed' for the currently signed in player. If revealing for the first time, *On achievement revealed* will be triggered.

Unlock

If an achievement is not already unlocked, set its state to unlocked for the currently signed in player. If unlocking for the first time, *On achievement unlocked* will be triggered.

Request player details

Request the details of the current player, such as their name and avatar image. If successful *On player details received* triggers and the *Player details* category of expressions can be used.

Sign in

If the player is not already signed in, pop up a window that allows them to sign in. Due to popup blockers, this may only work in a user input event, such as *On button clicked* or *On touch start*.

Sign out

If the player is already signed in, sign them out. This also allows for a different user to then sign in.

Request hi-scores

Request a hi-score list for a given leaderboard. Scores can be returned for public results, or "social" (from users connected to the currently signed in player), and a time limit can be applied such as to return only the day's best scores so far. The *top* type returns the very highest scores, and the *window* type returns the scores around the current player's own best score, allowing them to see where they appear in the rankings.

Submit score

Submit a new hi-score to a leaderboard. A *tag* can be provided, which is just a short string (up to 64 characters) associated with this score board entry, e.g. a short comment or an alternative alias for the player. If successfully submitted, *On score submit success* triggers.

Show leaderboards(Android only)

Shows the native leaderboards dialog for the currently signed in player.

Show leaderboard (Android only)

Shows the native leaderboard dialog with the specified leaderboard for the currently signed in player.

Google Play expressions

AchievementsCount

The total number of achievements available. The achievements list must already have been successfully requested.

AchievementNameAt(index)

AchievementDescriptionAt(index)

AchievementIDAt(index)

AchievementStepsAt(index)

AchievementTotalStepsAt(index)

AchievementTypeAt(index)

Retrieve information about a given achievement in the achievements list. The achievements list must already have been successfully requested.

AchievementUnlockedIconURLAt(index)

AchievementRevealedIconURLAt(index)

Retrieve the icon image URL for a given icon in either its unlocked or revealed state. This can be displayed using the Sprite object's *Load image from URL* action.

ErrorMessage

In *On error*, the relevant error message if available.

HiScoreCount

The number of hi-scores in the current returned list of results.

HiScoreTotalCount

The total number of scores in the leaderboard, which may be greater than the number of returned results (*HiScoreCount*).

HiScoreAt(index)

HiScoreFormattedAt(index)

Return a numerical value, or formatted string, for a score at a given index.

HiScoreRankAt(index)

HiScoreFormattedRankAt(index)

Return the numerical rank, or formatted string of the rank (e.g. "1st"), at a given index.

HiScoreNameAt(index)

Return the name of the player associated with the score at an index.

HiScoreTagAt(index)

Return the tag (a short string) that was submitted along with the score at an index.

HiScoreMyBest

HiScoreMyFormattedBest

HiScoreMyBestRank

HiScoreMyBestFormattedRank

HiScoreMyBestTag

Return the details for the current player's own best score, including the numerical and formatted versions of the score and rank.