CONSTRUCT GAME SERVICES PLUGIN

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

The **Construct Game Services** (CGS) plugin allows accessing game services hosted by Scirra, the makers of Construct. Features include authentication, cloud save, and submitting high scores both for the Construct Arcade and cross-platform leaderboards.

Scripting

When using JavaScript or TypeScript coding, the features of this object can be accessed via the ICGSObjectType script interface.

Authentication

If you want to allow the player to sign in, you can make a login system with these steps.

1. In your Construct Game Services account, create a game. Take note of the game ID.
2. Use an action like *Sign in with provider* to provide a login using one of several existing account providers like Google, Microsoft or Steam. (You may want to allow multiple providers.) You'll need to specify the game ID to sign in to.
3. When the player chooses this option, a popup window will appear asking them to log in, or if they are already logged in, to confirm that they wish to proceed. (In desktop exports this will instead launch the system browser, and in mobile exports it will use an in-app browser so the user does not have to leave the app.)
4. If the player signs in successfully, then *On success* will trigger for *Sign in with provider*. You can then access the *PlayerName* and *PlayerID* expressions, and make use of other features like submitting scores to cross-platform leaderboards.

See the Log in with providers example for a demonstration of this, including support for handling popup blockers and persistent sign in.

Handling popup blockers

When running in a web browser, it is possible the browser popup blocker will prevent the login popup window from opening. In this case *On sign in popup blocked* will trigger, and you can provide an option to retry opening the popup window with the action *Retry sign in popup*. You should display a message to the user explaining what happened and be sure to use the retry action in a user input trigger, such as a button click or mouse click, otherwise the second attempt may also be blocked.

You can test that your retry flow works correctly by checking the Construct Game Services plugin property *Simulate popup blocked*. When enabled this will not actually attempt to open a popup window on the first attempt, and will instead act as if the browser popup blocker prevented it from opening, triggering *On sign in popup blocked*. Then the second attempt made with *Retry sign in popup* will attempt to open the popup window normally. Turn this on to check your project handles popup blockers correctly, but be sure to turn it off again afterwards!

In desktop and mobile exports the popup will never be blocked - this only needs to be handled for web browsers.

Sign in persistent

If the user signs in with *Allow persisting* enabled, then their sign in is remembered. Upon returning the *Can sign in persistent* condition will be true, and you can offer an option to log in again using the *Sign in persistent* action. This can continue with the previous sign in without showing any user interface.

If the user signs out, it will delete their remembered sign in, so *Can sign in persistent* will not be true when they return to your game.

Leaderboards

Scores can be submitted to either the Construct Arcade or a cross-platform leaderboard.

Construct Arcade leaderboards

To publish a game to the Construct Arcade with leaderboards, follow these steps:

1. Add the Construct Game Services plugin to your project
2. Add a *Submit score* action to submit the player's score at the appropriate time (such as when a 'Game over' screen appears). Leave the *Leaderboard ID* parameter empty.
3. Publish to the Construct Arcade following the normal process
4. Play your game until a score is submitted
5. Reload the page and a leaderboard will have appeared with your score

Note that you do not need to add any authentication features to your project to submit a score to the Construct Arcade, as it handles authentication automatically.

Cross-platform leaderboards

To make use of leaderboards on other platforms, follow these steps:

1. In your Construct Game Services account, create a game (if you haven't already), and then create a leaderboard in that game. Take note of both the game ID and leaderboard ID.
2. Add a sign-in feature to your game using the authentication features, as players must be signed in to be able to submit a score. You'll need to specify the game ID to sign in to.

3. Once a player is signed in, you can use the *Submit score* action, specifying the leaderboard ID to submit to.

You should then be able to find the score on the leaderboard in your Construct Game Services account. You can display scores in your game using the *Get leaderboard scores* action.

See the Online leaderboards example to try it out and see how it works.

Cloud save

Construct Game Services allows saving relatively small amounts of data for the signed in player. This can be used for a range of purposes, but is ideal for saving the user's progress online. This allows players to restore their progress on any device that they log in to the same account with, rather than only saving to the local device storage (which is what Construct's savegame feature does by default).

To set up a cloud save feature, you only need to have a user successfully signed in (see the *Authentication* section above). Then you can change Construct's savegame feature to save online by using the *Save game to JSON string* and *Load game from JSON string* actions, and saving and loading the JSON string with the Cloud Save *Create* and *Get* actions. See the Cloud Save example for a demonstration.

You can store data under different key names, such as "slot1", "slot2" etc. if you wanted to provide different save slots. However for the simple case of a single online save per user, you only need a single key name like "savegame".

You can also create separate storage *buckets* in your Construct Game Services account, which are accessible to all players. However in the simple case of using storage private to a player's own account, you can leave the bucket ID empty.

CGS properties

Simulate popup blocked

When enabled, act as if the first attempt to open the login popup window is blocked, requiring the use of *Retry sign in popup*. See the section *Handling popup blockers* above for more information.

CGS conditions

Can sign in persistent

True if the user previously successfully signed in when *Allow persisting* was enabled and did not sign out. This means the *Sign in persistent* action can be used to continue with the previous sign in.

Is signed in

True if the user is currently signed in.

On sign in popup blocked

Triggered during sign in if the browser prevented the login popup window from appearing. See the section *Handling popup blockers* above for more information.

On create success

On create error

Triggered after the cloud save *Create* actions with the corresponding bucket ID and key, depending on the outcome of the operation.

On get success

On get binary success

Triggered after the cloud save *Get* action with the corresponding bucket ID and key completes successfully. When the *Get* action *Type* is *text*, *On get success* triggers, and the downloaded data is provided by the *CloudSaveText* expression; when *Type* is *binary*, *On get binary success* triggers, and the downloaded data is placed in the specified Binary Data object.

On get error

Triggered after the cloud save *Get* action with the corresponding bucket ID and key fails to complete. This could occur if the user went offline, or if a network error occured.

On success

On error

Triggered after an action depending on whether it completed successfully or failed.

CGS actions

Sign in with provider

Attempt to sign the user in with a third-party identity service like Google or Microsoft. The game ID to sign in to from your Construct Game Services account must be provided. If *Allow persisting* is enabled, a successful sign in is remembered, and may be re-used in future with the *Sign in persistent* action. The *Expiry* is the time in minutes that sessions remain active; the plugin will attempt to renew sessions shortly before expiry to extend the session. Signing in with a provider will open a popup window, and you can also specify the size of this popup window. Upon completing a sign in, *On success* will trigger for *sign in with provider*. In some cases if the sign in is cancelled or fails, *On error* will trigger.

Note that in some cases, sign in may be cancelled in a way that cannot be detected (neither On success nor On error will trigger). Be sure to design your project with this in mind - for example do not block the user interface until sign in finishes, as you cannot reliably detect that.

Sign in persistent

Attempt to sign in re-using a previous successful sign in that allowed persisting. This can only be used when *Can sign in persistent* is true. This will trigger either *On success* or *On error* for *sign in persistent* depending on whether the persistent sign in completed successfully.

Sign out

Sign out of any account the user is currently signed in to, and also delete any remembered sign in if it allowed persisting. Locally this operation completes immediately and so there is no corresponding success or error trigger. For completeness signing out will send a request in the background to ensure the session is ended on the server-side as well, but that is optional and if it fails the session will time out anyway.

Retry sign in popup

After *On sign in popup blocked* triggers, attempt to open the login popup window again. See the section *Handling popup blockers* above for more information.

Set player name

Set the name of the currently signed in player. In some cases the player name is obtained from the authentication provider, but in others a generic name will be used, in which case the player will likely want to change their name for features like leaderboards. The current player name is provided by the *PlayerName* expression.

Create (binary)

Create (text)

Create or replace the data for a key. A game bucket ID can optionally be provided, but if left empty it will use player private storage. The *text* variant uploads a string for the data, whereas the *binary* variant uploads the contents of a Binary Data object. *On create success* or *On create error* will trigger depending on whether the upload completed successfully.

Get

Download the data previously uploaded for a key with one of the *Create* actions. A game bucket ID can optionally be provided, but if left empty it will use player private storage. If this

fails, *On get error* will trigger. Otherwise when *Type* is *Text*, *On get success* will trigger, and the downloaded data will be in the *CloudSaveText* expression; when *Type* is *binary*, *On get binary success* will trigger, with the downloaded data stored in the specified Binary Data object.

Submit score

Submit a score to a leaderboard. The *Score* must be an integer (fractional scores are not supported). The *Leaderboard ID* may be left empty to submit a score on the Construct Arcade, which does not require authentication. Otherwise it may be set to the leaderboard ID to submit the score to, in which case authentication is required.

Get leaderboard scores

Request a page of scores from a given leaderboard ID, with a specified zero-based page number and number of results per page. This does not require authentication. The returned scores can optionally be filtered with several options. An ISO 3166-1 alpha-2 country code filter can be provided, e.g. "US" to only return scores submitted in the United States of America. A time range can also be specified, such as *Daily* to return today's scores. Weekly leaderboards run from Monday to Sunday. A range offset can also be specified to return a prior range - for example specifying a *Daily* range with offset 1 will return yesterday's scores. The *Culture* can be set to a locale to use for returned values, or left empty to use the leaderboard's default culture. *On success* will trigger for *get scores* when scores are retrieved successfully, in which case the leaderboard expressions can be used to access them.

CGS expressions

GameID

A string of the game ID currently signed in to.

PlayerID

A string with a unique ID to identify the currently signed in player.

PlayerName

The display name of the currently signed in player.

CloudSaveBucketID

CloudSaveKey

CloudSaveName

In a Cloud Save trigger, the bucket ID, key and name of the associated cloud save operation.

CloudSaveText

After *On get success* triggers (after the *Get* action with type *text*), the string of text data that was downloaded.

ScoreCount

After successfully retrieving leaderboard scores, the number of scores available.

ScoreAt(index)

ScoreFormattedAt(index)

ScoreRankAt(index)

ScoreFormattedRankAt(index)

ScorePlayerIDAt(index)

ScorePlayerNameAt(index)

ScoreCountryAt(index)

After successfully retrieving leaderboard scores, these expressions return information about a score at a given zero-based index. Scores and ranks are numbers, but the formatted versions return a string formatted according to the locale. Countries are returned as ISO 3166-1 alpha-2 country codes.

TotalPageCount

After successfully retrieving leaderboard scores, the total number of pages available.