# IPROJECT INTERFACE

**View online:** https://www.construct.net/en/make-games/manuals/addon-sdk/reference/model-interfaces/iproject

The `IProject` interface provides access to a project from the SDK.

## Methods

### GetName()

Return the project name.

### GetObjectTypeByName(name)

### GetFamilyByName(name)

### GetObjectClassByName(name)

Look up an object class by a case-insensitive string of its name, returning either an IObjectType (for *GetObjectTypeByName*), an IFamily (for *GetFamilyByName*), or either (for *GetObjectClassByName*), or `null` if not found.

### GetObjectClassBySID(sid)

Look up an object class by its SID (Serialization ID), returning either an IObjectType or IFamily, or `null` if not found.

> *"object" type properties store the SID of the chosen object class, so this method allows identifying the corresponding object class in the editor.*

### CreateObjectType(pluginId, name)

Add a new object type to the project. Returns a promise that resolves with an IObjectType representing the new object type. See Finding addon IDs to get a list of possible plugin IDs that can be used. `name` is the requested name to use for the object type. If the name is free, it will be used directly; however if the name is already in use, Construct will change the name to one which is available. Call `GetName()` on the returned IObjectType to determine what name it was assigned.

### GetSystemType()

Return an IObjectType representing the System plugin, which exists in every project.

### GetSingleGlobalObjectType(pluginId)

Return an IObjectType representing a single-global plugin in the project. Returns `null` if the given plugin ID does not exist, is not a single-global plugin, or the plugin has not been added to the project. See Finding addon IDs to get a list of possible plugin IDs that can be used.

### CreateFamily(name, members)

Create a new family in the project. *name* is an optional family name (pass `null` to use a default name). *members* must be an array of IObjectType representing the object types to add to the family. Families must be created with at least one object type, and if they have multiple object types, they must all be from the same kind of plugin (e.g. all Sprites). Returns an IFamily representing the created family.

### GetInstanceByUID(uid)

Look up an instance by its UID (Unique ID), returning either a IObjectInstance or IWorldInstance depending on the kind of instance, or `null` if not found.

### GetProjectFileByName(name)

Look up a project file by a case-insensitive string of its filename, returning an IProjectFile if found, else `null`.

### GetProjectFileByExportPath(path)

Look up a project file by a string of its path after export, returning an IProjectFile if found, else `null`. Note the path after export depends on the project *Export file structure* setting. In the legacy *Flat* mode, file paths are always at the root level (even if in a subfolder in the Project Bar) and names are case-insensitive. In the modern *Folders* mode, file paths correspond to the subfolders in the Project Bar and are case-sensitive. This method is useful for being able to identify in the editor the project file that corresponds to a relative URL in the runtime.

### GetProjectFileBySID(sid)

Look up a project file by its SID (Serialization ID), returning an IProjectFile if found, else `null`.

> *"projectfile" type properties store the SID of the chosen project file, so this method allows identifying the corresponding project file in the editor.*

### AddOrReplaceProjectFile(blob, filename, kind = "general")

Create a new project file in the project, or replace the content of the file if it already exists, using a `Blob` for the file content and a string for the filename. The *kind* defaults to `"general"`, which causes the file to be placed in the "Files" folder in the Project Bar. Other options are `"sound"`, `"music"`, `"video"`, `"font"` and `"icon"`.

---

**ShowImportAudioDialog(fileList)**

Bring up the *Import audio* dialog to import a list of audio files given in `fileList`. This will automatically transcode the audio files to WebM Opus (when supported for the audio formats), which is the main format Construct uses. Prefer importing PCM WAV files to ensure transcoding is supported and is lossless. The file list should be an array of `Blob` or `File`; if blobs, then ensure a `name` property is assigned to the blob object to indicate the intended filename.

> *Reading blobs from IZipFile automatically assigns a name property so the blobs can be directly passed to this method.*

---

**EnsureFontLoaded(f)**

Make sure a given font name is loaded so it can be used when drawing text. This is necessary for plugins that render text.

---

**UndoPointChangeObjectInstancesProperty(instances, propertyId)**

Create a new undo point that undoes changes to `propertyId`. `instances` must be either an IObjectInstance or an array of IObjectInstance. Call this method before changing an instance's property value and the action will be undoable.