THE LANGUAGE FILE

View online: https://www.construct.net/en/make-games/manuals/addon-sdk/guide/language-file

In the .c3addon file, **en-US.json** is the default language file containing all the addon's strings that are shown in the editor UI in a JSON format. Moving these strings to a separate file makes it possible for the software to be fully translated.

All strings must be provided in US English (hence the filename en-US.json) since this is the default language of Construct 3, and the common language from which all other languages are translated.

Overall structure

The overall structure of the language file is as follows, taken from the plugin SDK template. Note that in general, the language file uses an ID as a key on the left, and the string to display as the value on the right.

```
{
	"languageTag": "en-US",
	"fileDescription": "Strings for MyCustomPlugin.",
	"text": {
		"plugins": {
			"mycompany_myaddon": {
				"name": "My Custom Plugin",
				"description": "Description for my custom plugin.",
				"help-url": "https://www.scirra.com",
				"properties": {
					"test-property": {
						"name": "Test property",
						"desc": "A test number property. Displayed by 'Al
					}
				},
				"aceCategories": {
					"custom": "Custom"
				},
				"conditions": {
					"is-large-number": {
						"list-name": "Is large number",
						"display-text": "[i]{0}[/i] is a large number",
						"description": "Test if a number is greater than
						"params": {
							"number": {
								"name": "Number",
```

```
                                                        "desc": "Number to test if greate
                                                }
                                        }
                                }
                        },
                        "actions": {
                                "do-alert": {
                                        "list-name": "Do alert",
                                        "display-text": "Do alert",
                                        "description": "Do a dummy alert."
                                }
                        },
                        "expressions": {
                                "double": {
                                        "description": "Double a number.",
                                        "translated-name": "Double",
                                        "params": {
                                                "number": {
                                                        "name": "Number",
                                                        "desc": "The number to double."
                                                }
                                        }
                                }
                        }
                }
        }
}
```

The addon SDK provides a JSON schema to help you write language files, as it provides autocomplete and validation in compatible editors.

Required fields

`"languageTag"` must be "en-US".

`"fileDescription"` is not used in the editor. It provides a hint to translators. It can simply say "Strings for [addon name]".

`"text"` represents the root node of the string tree, and "plugins" represents strings for all plugins. These should be left as they are.

`"mycompany_myaddon"` should be the **lowercase addon ID.** Note if your addon ID contains any uppercase characters, they should be lowercased for this key in the language file. The remaining strings all belong inside this key, since it represents your plugin.

`"name"` is the name of your plugin as it appears in the editor. This can be changed at any time, but the plugin ID should not be changed after release.

`"description"` is a short sentence or two describing what your plugin does.

`"help-url"` is a URL to documentation or support for your plugin.

Note: themes only need to use the `"name"`, `"description"` and `"help-url"` fields.

Strings for properties/parameters

For each PluginProperty your plugin uses, there must be a key with the property ID under `"properties"`. For effects, there must be a key with the parameter ID under a `"parameters"` instead, but it otherwise works the same. The required strings for each property are:

- `"name"` — the name of the property, which appears to the left of the field
- `"desc"` — the property description, which appears in the footer of the Properties Bar

For example given the following property:

```
new SDK.PluginProperty("integer", "test-property", 0)
```

The following language strings can be used under the `"properties"` key:

```
"test-property": {
        "name": "Test property",
        "desc": "A test number property. Displayed by 'Alert' action."
}
```

Some properties require additional keys.

Combo properties

The `"combo"` property type needs an extra `"items"` key to set the visible name of each item. Each key underneath this should be the ID of the combo item, and its value the name to use. Here is an example from the Audio plugin. The property is created as:

```
new SDK.PluginProperty("combo", "timescale-audio", {
        initialValue: "off",
        items: ["off", "sounds-only", "sounds-and-music"]
})
```

Note the items defined here are IDs rather than displayed strings. The strings to display are set in the language file like this:

```
"timescale-audio": {
        "name": "Timescale audio",
```

```
        "desc": "Choose whether the audio playback rate changes with the time scale.",
        "items": {
                "off": "Off",
                "sounds-only": "On (sounds only)",
                "sounds-and-music": "On (sounds and music)"
        }
}
```

Link properties

The `"link"` property type needs an extra `"link-text"` key to set the text of the clickable link. An example is below.

```
"make-original-size": {
        "name": "Size",
        "desc": "Click to set the object to the same size as its image.",
        "link-text": "Make 1:1"
}
```

Category names

When defining ACEs, category IDs are used rather than category names. The `"aceCategories"` key defines the displayed name of each category. The following example displays all ACEs in the category ID `"customCategory"` as being in a section labelled `"My custom category"`.

```
"aceCategories": {
        "customCategory": "My custom category"
}
```

Category names are shared across actions, conditions and expressions.

ACE strings

Strings for conditions, actions and expressions are listed in the keys `"conditions"`, `"actions"` and `"expressions"` respectively. Note they are not sorted by category here; each section lists all ACEs of that type.

Similar to properties, each key under each section is the ID of the action, condition or expression. As with the definitions themselves, actions and conditions work slightly differently to expressions.

Action and condition strings

The required keys are:

- `"list-name"` — the name that appears in the condition/action picker dialog.

- `"display-text"` — the text that appears in the event sheet. You can use simple BBCode tags like `[b]` and `[i]`, and use `{0}`, `{1}` etc. as parameter placeholders. (There must be one parameter placeholder per parameter.) For behaviors only, the placeholder `{my}` is substituted for the behavior name and icon.
- `"description"` — a description of the action or condition, which appears as a tip at the top of the condition/action picker dialog.

Expression strings

The required keys are:

- `"description"` — the description that appears in the expressions dictionary, which lists all available expressions.
- `"translated-name"` — the translated name of the expression name. In the en-US file, this should simply match the expression name from the expression definition. This key mainly exists so it can be changed in other languages, making it possible to translate expressions in some contexts. Note when actually typing an expression the non-translated expression name must always be used.

Parameters

Actions, conditions and expressions can omit the `"params"` key if they have no parameters. However if they have any parameters this key must be present, and each parameter must have its own key inside with the ID of the parameter. Similar to the plugin properties, each key must have a `"name"` and `"desc"`. For `"combo"` parameters there must also be a property `"items"` (which work the same as `"combo"` property types: each item ID maps to its display text). For `"combo-grouped"` parameters, there must also be a property `"itemGroups"` (see below for an example).

Example

The *Is large number* condition in the plugin SDK uses the following definition in aces.json:

```
{
        "id": "is-large-number",
        "scriptName": "IsLargeNumber",
        "highlight": true,
        "params": [
                {
                        "id": "number",
                        "type": "number"
                }
        ]
}
```

Its corresponding language strings are defined in en-US.json as follows:

```
"is-large-number": {
	"list-name": "Is large number",
	"display-text": "[i]{0}[/i] is a large number",
	"description": "Test if a number is greater than 100.",
	"params": {
		"number": {
			"name": "Number",
			"desc": "Number to test if greater than 100."
		}
	}
}
```

"combo-grouped" example

When using the `"combo-grouped"` parameter type, the definition in aces.json might look like this:

```
{
	"id": "dinosaur",
	"type": "combo-grouped",
	"itemGroups": [{
		"id": "theropods",
		"items": ["tyrannosaurus", "velociraptor", "deinonychus"]
	}, {
		"id": "sauropods",
		"items": ["diplodocus", "saltasaurus", "apatosaurus"]
	}
}
```

Its corresponding language strings can be defined in en-US.json inside the `"params"` key as follows:

```
"dinosaur": {
	"name": "Dinosaur",
	"desc": "Choose a dinosaur",
	"itemGroups": {
		"theropods": {
			"name": "Theropods",
			"items": {
				"tyrannosaurus": "Tyrannosaurus",
				"velociraptor": "Velociraptor",
				"deinonychus": "Deinonychus"
			}
		},
		"sauropods": {
			"name": "Sauropods",
```

```
                        "items": {
                                "diplodocus": "Diplodocus",
                                "saltasaurus": "Saltasaurus",
                                "apatosaurus": "Apatosaurus"
                        }
                }
        }
}
```