Easy Feedback Documentation

Release 1.1.0

Noah Ratcliff

Nov 26, 2017

Contents:

CHAPTER 1

Getting Started

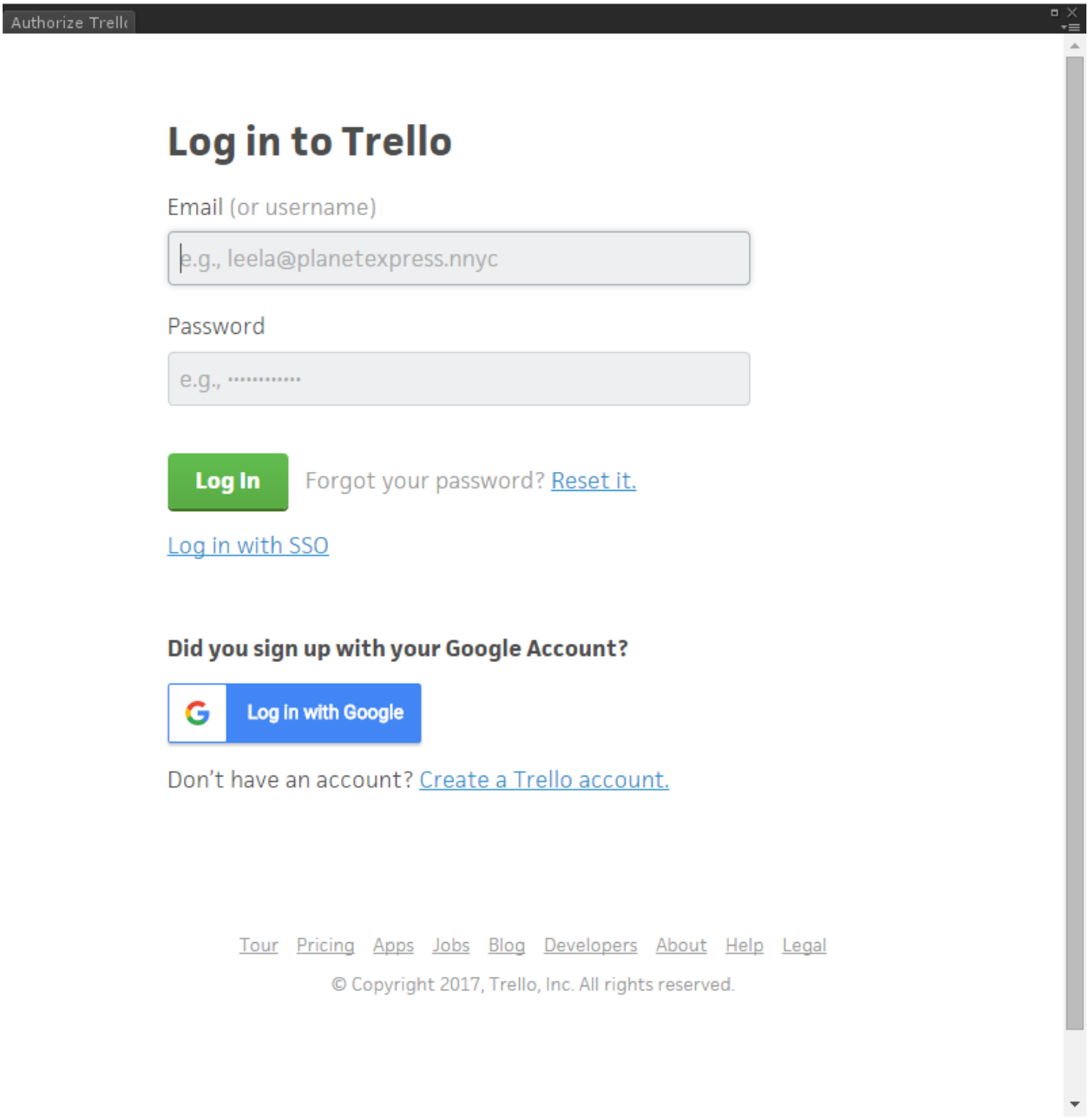
1.1 Authorizing with Trello

After adding the asset package to your project, you'll need to authorize Easy Feedback with Trello. To do this, open the Easy Feedback configuration window at `Edit -> Project Settings -> Easy Feedback Form` in the toolbar, and click "Get Trello API Token."

If you are using Unity 5.0 or above, you'll be presented with a window prompting you to log in to Trello. If you are using Unity 5 or below, you'll be directed to your browser to authorize.

Note: It is highly recommended that you create a new account for use with Easy Feedback, as an API key with write permission for the account is used to make changes to your feedback board, and will be included with builds of your

project.

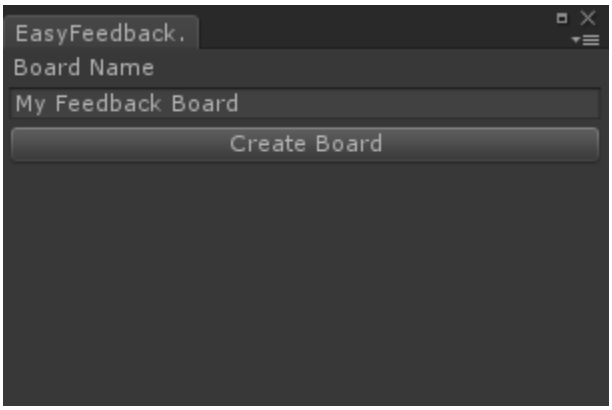


After logging in, click "Allow" to allow Easy Feedback to use your account.

Copy the token given to you on the next page, paste it in the "Token" field in the configuration window, then click "Authenticate With Token." Easy Feedback will now finish the authentication process, and load your Trello information.

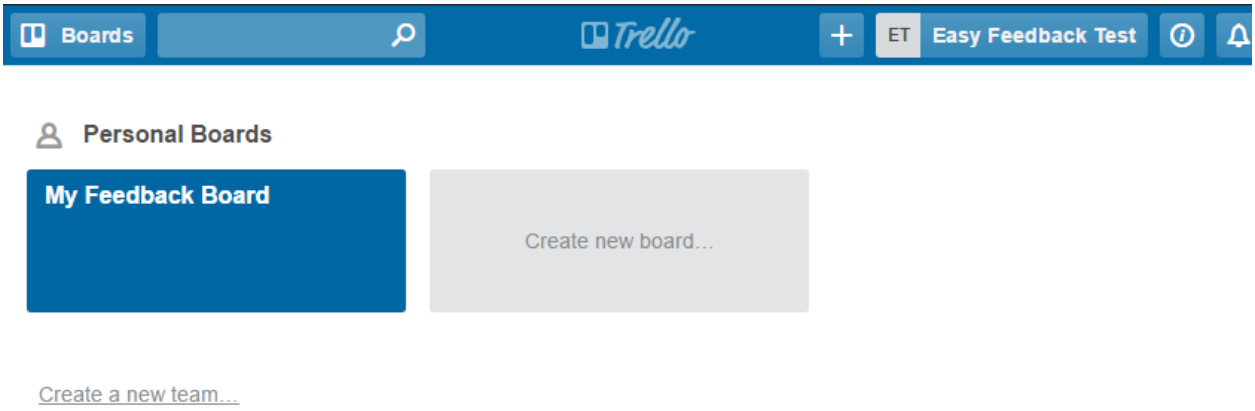
1.2 Setting up a feedback board

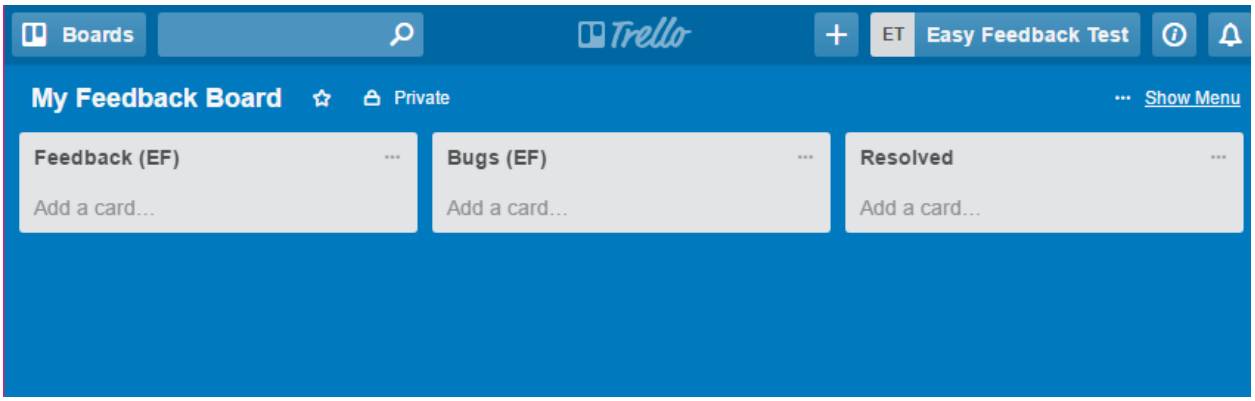
If this is your first time using Easy Feedback on this account, you won't have any boards. To set up a new board, click "New Board."



In the window that appears, enter the name of your new feedback board, then click "Create Board."

Your new board will now be available in the "Feedback Board" dropdown. If this is the first board for your account, it will be selected by default. You'll also be able to find the new board on your Trello account!





Easy Feedback is now all configured and ready to go! If all went well, the configuration window should now look something like this:

1.3 Adding the form to a scene

To add the form to a scene, simply drag the Feedback prefab into the scene.

If there isn't one already, add an EventSystem to the scene as well. To add an EventSystem, select `Game Object -> UI -> Event System` in the toolbar.

That's all you need to get started! Try running your project and submitting a report. If everything went well, your new report should appear on your feedback board!

CHAPTER 2

Configuring Easy Feedback

The Easy Feedback configuration window can be opened from `Edit -> Project Settings -> Easy Feedback Form`. Before authenticating with Trello, it will have few options. For help authenticating with Trello, see *Authorizing with Trello*.

2.1 Log out of Trello

Clears the current Trello API token, effectively logging Easy Feedback out of the currently authenticated Trello account.

2.2 New Board

Displays the "New Board" window, which creates a new feedback board on the authenticated Trello account.

2.3 Refresh Boards

Updates the local board information cache. Useful for when you've made changes to your feedback board outside of the Unity editor.

2.4 Feedback Board

The board on your account that all reports from Easy Feedback will be sent to. Only feedback boards will be listed here.

2.5 Subscribe to board

Whether or not the authenticated user is subscribed to the current feedback board. Depending on your settings, subscribing to a board will give you alerts when cards are added to the board.

Note: Changes to your subscribed state on Trello will change the value of this toggle.

2.6 Store Locally

If checked, reports will not be sent to Trello, and will instead be stored on the local machine.

Default: *unchecked*

Note: This is the only setting available in the configuation window when not authenticated with Trello. All others require authentication. See *Authorizing with Trello* for more help.

CHAPTER 3

Feedback Boards

Your feedback board is where all of the reports made in your game are sent. A feedback board is very customizable, but they all share some common properties that distinguish them from standard boards.

3.1 Anatomy of a Feedback Board

3.1.1 Categories (lists)

Report categories on your feedback form are just lists on Trello. To distinguish category lists from standard lists, all category list names must end with the `(EF)` tag. The name of the category on the feedback form is dictated by the name of the category list on Trello (the `(EF)` tag is not included in the category name on the feedback form). Lists without the `(EF)` tag will be ignored by Easy Feedback, and will not be included as categories on your feedback form.

Note: All Easy Feedback boards must have at least one category list, or they will not appear in the "Feedback Boards" dropdown on the Easy Feedback configuration window.

3.1.2 Labels

By default, all labels on a feedback board are treated as priorities for reports, and will appear in the priority dropdown. All label information for the current feedback board is included in the *EFConfig*.

3.1.3 Cards

Cards added to the feedback board by Easy Feedback are reports, and contain information submitted by the user.

You may add your own cards to the board, as all cards on the board are ignored by Easy Feedback.

3.2 Customizing your Feedback Board

Note: You *must* update the cached board information in your game for changes to categories or priorities to be reflected in your game.

3.2.1 Renaming categories

To change the name of a category, first change the name of the category list on Trello. Make sure to leave the `(EF)` at the end of the list name on Trello.

After changing the name on Trello, open the Easy Feedback configuration window from `Edit -> Project Settings -> Easy Feedback Form` and click "Refresh Boards" to update the category name on your form.

3.2.2 Adding a category

To add a category to your feedback form, first create a new list on your feedback board on Trello. Be sure to include `(EF)` at the end of your new list's name.

After creating the list on Trello, open the Easy Feedback configuration window from `Edit -> Project Settings -> Easy Feedback Form` and click "Refresh Boards" to update the categories on your form.

3.2.3 Removing a category

To remove a category from your form, either archive the list from your feedback board on Trello, or remove the `(EF)` tag from the end of the list name.

After editing the list on Trello, open the Easy Feedback configuration window from `Edit -> Project Settings -> Easy Feedback Form` and click "Refresh Boards" to update the categories on your form.

Warning: If you remove a category that is included in old builds of your game, the category will still be available in the feedback form on those builds, and any attempts to submit feedback to that category will fail.

3.2.4 Renaming priorities

To change the name of a priority, first change the name of the corresponding label on Trello.

After changing the name on Trello, open the Easy Feedback configuration window from `Edit -> Project Settings -> Easy Feedback Form` and click "Refresh Boards" to update the priority name on your form.

3.2.5 Adding a priority

To add a priority to your feedback form, first create a new label on your feedback board on Trello.

After creating the label on Trello, open the Easy Feedback configuration window from `Edit -> Project Settings -> Easy Feedback Form` and click "Refresh Boards" to update the priorities on your form.

3.2.6 Removing a priority

To remove a priority from your form, first delete the corresponding label on your feedback board on Trello.

After removing the label on Trello, open the Easy Feedback configuration window from `Edit -> Project Settings -> Easy Feedback Form` and click "Refresh Boards" to update the priorities on your form.

Warning: If you remove a priority that is included in old builds of your game, the priority will still be available in the feedback form on those builds, and any attempts to submit feedback to that priority will fail.

CHAPTER 4

The Feedback Form

The feedback form is where players write their report. The feedback form object is highly customizable, and Easy Feedback comes with some prefabs for quickly adding new input fields to your form.

4.1 Configuring the Feedback Form

The Feedback Form component has a few exposed fields that can be configured. Unlike the settings found in the configuration window, changing these values will only affect the form instance you are editing.

4.1.1 Feedback Key

The key that toggles the feedback form in-game.

Default: *F12*

4.1.2 Include screenshot

Whether or not to include a screenshot with the report.

Default: *checked*

4.1.3 Form

The `Form` object in the Feedback game object children.

4.1.4 Alert

The `Alert` object in the Feedback game object children.

4.2 Customizing your Feedback Form

By default, the feedback form has category and priority dropdowns, a summary text field, and a detail text field. Objects containing scripts that collect metadata information like system information are also included under the `MetadataCollectors` object.

All of these elements may be removed or replaced as needed. Additional elements may be added to the form as well.

4.2.1 Form elements

Form elements are any components that alter the report in some way. The report category dropdown, debug log collector, and priority dropdown are all form elements.

See also:

Report

4.2.2 Form fields

Form fields are any components that alter a *section* on the report in some way. The detail text field, as well as most metadata collectors are form fields.

`FormField` inherits from `FormElement`, but also exposes some varibles that make it easier to quickly alter how the form field appears on the report.

All form fields have these public variables:

- **Section Title:** The title of this field's section on the report.
- **Sort Order:** Order of the section in the report (lowest first).

4.2.3 Prefabs

Easy Feedback comes with a few form field prefabs for quick drag and drop customization. These prefabs can be found in the project window at `Easy Feedback -> Prefabs -> Fields`. To add these fields to your form, just add them as children of `Form` on the `Feedback` object.

Dropdown

A simple dropdown input.

Public variables:

- **Label:** The label to prepend to this field on the report. No label will be included if this field is left blank.

InputField

A text input field.

Public variables:

- **Label:** The label to prepend to this field on the report. No label will be included if this field is left blank.

Toggle

A checkbox.

Public variables:

- **Label:** The label to prepend to this field on the report. No label will be included if this field is left blank.
- **Default:** The default value of the toggle.

CHAPTER 5

Writing Custom Form Fields

Because every game is different, you may want to write a custom *FormField* to include game-specific information with your reports. The *FormField* API provides a quick and easy way to start adding your own custom sections in your reports.

Lets look at how we can create a simple field that adds the text "Hello World!" to a custom section.

First, we'll need to implement the abstract *FormField* class in our new script:

```
using EasyFeedback;

public class MyFormField : FormField
{
        public override void FormClosed()
        {

        }

        public override void FormOpened()
        {

        }

        public override void FormSubmitted()
        {

        }
}
```

In `Awake()`, *FormField* finds the FeedbackForm in parent game objects, and adds listeners for FormClosed, FormOpened, and FormSubmitted to their respective callbacks in *FeedbackForm*.

Now, let's add some code to add our custom section to the report:

```
using EasyFeedback;

public class MyFormField : FormField
```

```
{
        public override void FormClosed()
        {

        }

        public override void FormOpened()
        {

        }

        public override void FormSubmitted()
        {
                // add section if it doesn't exist already
                if(!Form.CurrentReport.HasSection(SectionTitle))
                        Form.CurrentReport.AddSection(SectionTitle, SortOrder);

                // set section text
                Form.CurrentReport[SectionTitle].SetText("Hello world!");
        }
}
```

Let's break down what's going on here.

First, we added all of our code to the `FormSubmitted()` function. This function is called by the *Feedback Form* right before the current report is sent off to Trello. It is recommended that you add any last-minute or one-time information to the report in this function.

Let's look now at each line in the function:

```
// add section if it doesn't exist already
if(!Form.CurrentReport.HasSection(SectionTitle))
        Form.CurrentReport.AddSection(SectionTitle, SortOrder);
```

`Form` is a reference to the parent *FeedbackForm* of this field, and `Form.CurrentReport` is the current *Report* for the form. The current report is reset by the *FeedbackForm* every time it is submitted to Trello. `CurrentReport.HasSection(string name)` returns whether or not the current report has a section with the given name. `SectionTitle` is a string that serves as the title of this field's section, and is set in the editor. So, the first line checks if the current report has the section set in the editor.

If the report does not already have the section, we go ahead and add it to the report with `CurrentReport.AddSection(string name, int sortOrder)`. `SortOrder` is another value set in the editor, and serves as the order of this field's section in the report (lowest first).

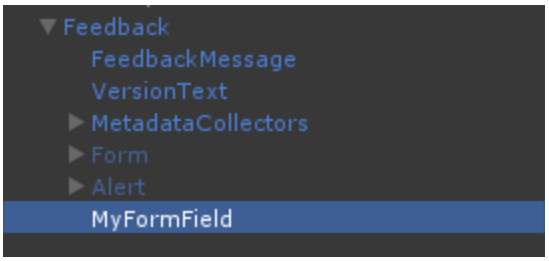
```
// set section text
Form.CurrentReport[SectionTitle].SetText("Hello world!");
```

Sections on the report are referenced by name via the report's indexer. Here, we're getting the section we just added to the report, and setting its text contents to the string "Hello world!"

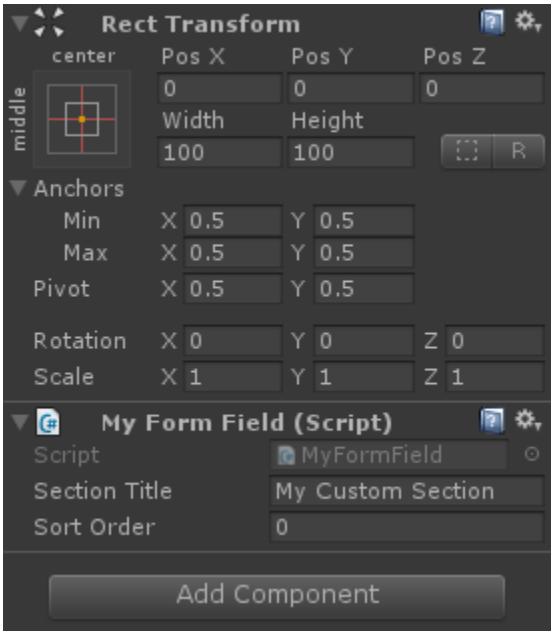
Now that we've written our custom field, let's add it to our feedback form!

First, we'll add a new child to the Feedback object for our field, and add the "MyFormField" script to it.

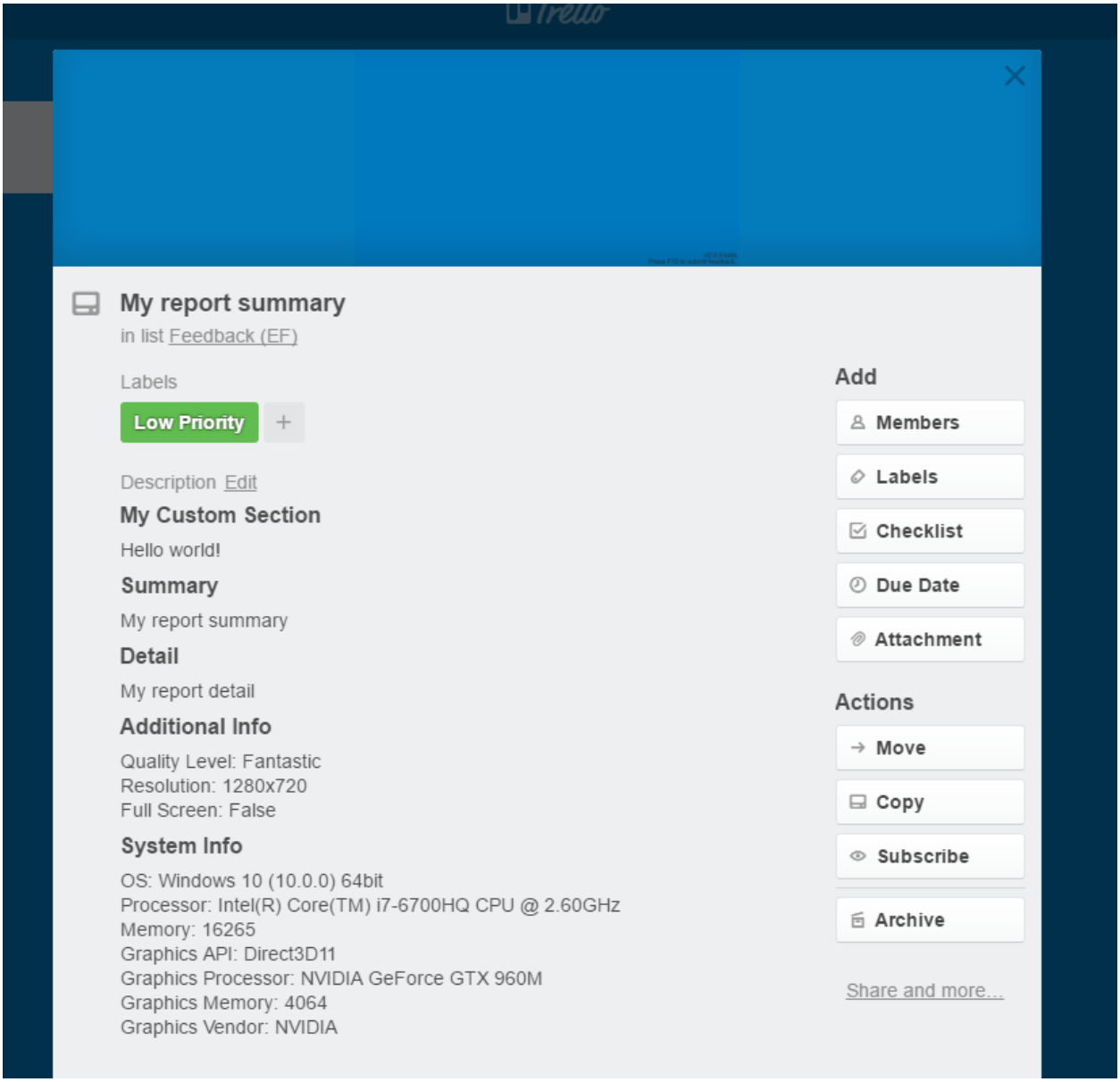
Note: Objects with FormField components must be a child of the Feedback object to work properly. They can be placed at any level in the heirarchy, as long as they are a child of the Feedback object. For example, in the Feedback prefab, FormFields that collect metadata information are organized under the MetadataCollectors object.



In the inspector, you'll see fields for the `SectionTitle` and `SortOrder` variables. We'll go ahead and call our section "My Custom Section" and we'll set the sort order to 0 so that it appears at the top of the report.



Let's test our new section! Run your scene, and submit a report. If all went well, our new custom section will appear at the top of the report!


My report summary
in list Feedback (EF)
Labels
Low Priority
Description Edit
My Custom Section
Hello world!
Summary
My report summary
Detail
My report detail
Additional Info
Quality Level: Fantastic
Resolution: 1280x720
Full Screen: False
System Info
OS: Windows 10 (10.0.0) 64bit
Processor: Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz
Memory: 16265
Graphics API: Direct3D11
Graphics Processor: NVIDIA GeForce GTX 960M
Graphics Memory: 4064
Graphics Vendor: NVIDIA
Add
Members
Labels
Checklist
Due Date
Attachment
Actions
Move
Copy
Subscribe
Archive
Share and more...

CHAPTER 6

Extending Easy Feedback

Although Easy Feedback comes with many options to start getting feedback as quickly as possible, every project is different, and custom integrations may be neccessary to collect game-specific metadata like player position or score. Luckily, Easy Feedback makes it easy to write your own custom fields to add additional behaviour to your feedback form.

This section serves as the scripting reference for Easy Feedback. *FeedbackForm*, *Report*, *ReportSection*, *EFConfig*, and *FeedbackBoard* contains information about their respective classes.

See *Writing Custom Form Fields* for a quick guide to getting started writing our own custom fields.

6.1 FeedbackForm

6.1.1 Namespace

EasyFeedback

6.1.2 Description

Controls the feedback form in the scene, and handles submission of the report.

6.1.3 Variables

Config: *EFConfig* object. *(EFConfig)*

FeedbackKey: Key to toggle feedback form. *(KeyCode)*

IncludeScreenshot: Whether or not to include a screenshot with reports. *(bool)*

Form: The transform of the feedback form. *(Transform)*

Alert: The transform of the alert pane. *(Transform)*

CurrentReport: The current *Report*. *(Report)*

IsOpen: Whether or not the form is currently being displayed. *(bool)*

6.1.4 Public Functions

public void Show(): Takes a screenshot (if IncludeScreenshot is true), then opens the form.

public void Hide(): Hides the form.

public void Submit(): Submits the form.

public void EnableForm(): Enables all of the Selectable elements on the form.

public void DisableForm(): Disables all of the Selectable elements on the form.

public void HideAlert(): Hides the currently displayed alert.

6.1.5 Callbacks

OnFormOpened: Called when the form is first opened, right before it is shown on screen.

OnFormSubmitted: Called right before the report is sent to Trello. This is where last-minute information should be added to the current *Report*.

OnFormClosed: Called when the form is closed, whether or not it was submitted.

6.2 Report

6.2.1 Namespace

EasyFeedback

6.2.2 Description

The report to be sent to Trello, or stored locally.

Reports contain a collection of *ReportSection*. Each section appears on Trello with a bold header and text underneath. Sections are ordered by sort order, with the lowest sort order value appearing higher on the report.

6.2.3 Variables

MAX_ATTACHMENTS: The maximum number of attachments allowed on a card by Trello. Because the screenshot is also an attachment, this number is one less than the true maximum. *(int)*

List: The list on Trello this report will be added to. *(EasyFeedback.APIs.List)*

Label: Labels to add to the card on Trello. *(EasyFeedback.APIs.Label)*

Title: The title of the card on Trello. *(string)*

Screenshot: Byte data for the screenshot to be included with this report. *(byte[])*

Attachments: Files attached to this report. *(List<EasyFeedback.FileAttachment>)*

6.2.4 Public Functions

public void AddSection(string title, [int sortOrder = 0]): Adds a new empty section to the report. Takes the title of the section *(string)*, and optionally the sort order of the section *(int)* as parameters. If not specified, the default sort order for the section will be 0.

public void RemoveSection(string title): Removes a section from the report. Takes the title of the section *(string)* to be removed as a parameter.

public void HasSection(string title): Returns whether or not the report has a section. Takes the title of the section *(string)* as a parameter. Returns a boolean value.

public string ToString(): Returns the contents of the report, formatted in Markdown, for Trello.

public string GetLocalFileText(): Returns the contents of the report, formatted for storing as a local text file.

public void AttachFile(string name, string filePath): Attaches the file located at *filePath* to the report. The name of the file on Trello is specified by the *name* parameter.

public void AttachFile(string name, byte[] data): Attaches a file to the report by byte array of data. The name of the file on Trello is specified by the *name* parameter.

6.3 ReportSection

6.3.1 Namespace

EasyFeedback

6.3.2 Description

A section of information on the report. Appears on Trello as a header (Title) with text underneath.

6.3.3 Variables

Title: The title of the section. *(string)*

SortOrder: The order of this element in the report (lowest first). *(int)*

6.3.4 Public Functions

public ReportSection(string title, [int sortOrder = 0]): Constructor. Takes the title and sort order of the section as parameters.

public ReportSection(string title, string text): Constructor. Takes the title and initial text contents of the section as parameters.

public void Append(string text): Appends text to the section's contents.

public void AppendLine(string line): Appends a line of text to the section's contents.

public void SetText(string text): Replaces the existing section text with specified text.

public string ToString(): Returns the section formatted in Markdown for Trello.

6.4 EFConfig

6.4.1 Namespace

EasyFeedback

6.4.2 Description

Configuration information for Easy Feedback.

6.4.3 Variables

StoreLocal: Save feedback locally, instead of sending it to Trello. *(bool)*

Token: Trello API token. *(string)*

Board: The selected *FeedbackBoard*. *(EasyFeedback.FeedbackBoard)*

6.4.4 Public Functions

EFConfig(): Constructor. This generally should not be called, as the EFConfig is created and managed as an asset in your project, and will be loaded by Easy Feedback.

6.5 FeedbackBoard

6.5.1 Namespace

EasyFeedback

6.5.2 Description

Information about the selected *Feedback Board*.

6.5.3 Variables

Id: The id of the board on Trello. *(string)*

ListNames: The names of all of the lists on the board. *(string[])*

ListIds: The ids of all of the lists on the board. *(string[])*

CategoryNames: The names of all of the category lists on the board. Defaults to "Feedback" and "Bug" if no feedback board is selected. *(string[])*

CategoryIds: The ids of all of the category lists on the board. Defaults to null and null if no feedback board is selected. *(string[])*

Labels: All of the labels on the board. Defaults to "Low Priority," "Medium Priority," and "High Priority" if no feedback board is selected. *(EasyFeedback.APIs.Label[])*

6.6 FormElement

6.6.1 Namespace

EasyFeedback

6.6.2 Description

Abstract class to be inherited by any components that are a part of the feedback form. Handles finding the *FeedbackForm* component in the parent object(s), and registers listeners for form callbacks.

6.6.3 Variables

Form: The *FeedbackForm* this object is a child of. (protected) *(EasyFeedback.FeedbackForm)*

6.6.4 Public Functions

public abstract void FormOpened(): Called by *FeedbackForm.OnFormOpened*.

public abstract void FormSubmitted(): Called by *FeedbackForm.OnFormSubmitted*.

public abstract void FormClosed(): Called by *FeedbackForm.OnFormClosed*.

6.7 FormField

6.7.1 Namespace

EasyFeedback

6.7.2 Description

Child of *FormElement*. To be inherited by any components that add a section to the report.

6.7.3 Variables

Form: The *FeedbackForm* this object is a child of. Inherited from *FormElement*. (protected) *(EasyFeedback.FeedbackForm)*

SectionTitle: The title of this field's section on the report. *(string)*

SortOrder: Order of the section in the report (lowest first). *(int)*

6.7.4 Public Functions

public abstract void FormOpened(): Called by *FeedbackForm.OnFormOpened*. Inherited from *FormElement*.

public abstract void FormSubmitted(): Called by *FeedbackForm.OnFormSubmitted*. Inherited from *FormElement*.

public abstract void FormClosed(): Called by *FeedbackForm.OnFormClosed*. Inherited from *FormElement*.

6.8 Markdown

6.8.1 Namespace

EasyFeedback

6.8.2 Description

Static class provided with EF in order to help you add Markdown formatting to your reports, without having to learn the **'Markdown'_** format.

6.8.3 Variables

HR: String denoting a horzontal rule or line. (public) *(const string)*

LINE_BREAK: String denoting a new paragraph. (public) *(const string)*

6.8.4 Public Functions

public static string Header(string text, Markdown.HeaderLevel level = Markdown.HeaderLevel.H1): Creates a header from the provided text, with the provided level.

public static string H1(string text): Creates a first-level header from the provided text.

public static string H2(string text): Creates a second-level header from the provided text.

public static string H3(string text): Creates a third-level header from the provided text.

public static string H4(string text): Creates a fourth-level header from the provided text.

public static string H5(string text): Creates a fifth-level header from the provided text.

public static string H6(string text): Creates a sixth-level header from the provided text.

public static string Em(string text): Formats the text with emphasis/italics.

public static string Strong(string text): Emboldens the text.

public static string Strike(string text): Strikes through the text.

public static string UnorderedList(string[] items): Creates an unordered (bulleted) list from the array of items.

public static string OrderedList(string[] items): Creates an ordered (numbered) list from the array of items.

public static string Hyperlink(string text, string url): Creates an inline hyperlink.

public static string Image(string url, string alt=""); Creates an inline image.

public static string Code(string text): Creates an inline span of preformatted text.

public static string CodeBlock(string text, string language=""); Creates a block of preformatted text. You may optionally provide a langugage for syntax highlighting, where supported.

public static string Blockquote(string text): Creates a block of quoted text.