

# Changelog

## 3.1.0 - 2023-08-01

### Changed

- Default to first list on board if none specified before report submitted
- Reset `FeedbackForm.CurrentReport` when form is opened

### Fixed

- Multiple log and screenshot attachments in report if form is opened and closed repeatedly before submitting a report

## 3.0.0 - 2023-06-26

### Added

- Input system support
- Create feedback prefab from settings menu
- Documentation, help email, and review links in settings menu

### Changed

- Move asset to Packages
- Settings asset stored in `Assets/Settings` by default
- Allow settings asset to be moved
- Refactor form input handling
- Update to match AeLa current code style guides
- Move cursor visibility management to `ShowHideMouse` component

### Fixed

- Deprecation warning for `UnityWebRequest.isHttpError` and `UnityWebRequest.isNetworkError` in 2020.3+
- Deprecation warning in 2021+ for `Texture.Resize`

## 2.2.0 - 2023-01-09

### Added

- Option to use Legacy screenshot capture mode.

### Fixed

- "A Native Collection has not been disposed" error.

## 2.1.0 - 2021-09-26

### Added

- Option to resize screenshots larger than 1080p.
- [Editor] Warning about Trello's attachment filesize limits.

### Changed

- Capture screenshot to memory instead of local file.
- Use attachment API to upload screenshot.

### Fixed

- All attachment uploads fail if screenshot upload fails.
- Screenshots not captured in WebGL builds.
- Screenshots sometimes left behind on filesystem.

- Crash on Switch on form opened.
- [Editor] Setup buttons on Feedback Form component don't do anything.

## 2.0.0 - 2021-06-02

### Added

- `AeLa.EasyFeedback`, `AeLa.EasyFeedback.Editor`, and `AeLa.EasyFeedback.Demo` assembly definitions
- Toast system for sending messages to the player
- Order field for label (priority) order in dropdown
- Email field on default Feedback prefab
- Button to open current feedback board in settings

### Changed

- Updated namespaces for new assemblies
- Replaced submitting/submitted/error popup with toasts to improve submission UX
- Configuration moved to Project Settings
- Minor settings UI changes
- Moved docs to DocFX

### Removed

- Dropped support for Unity 2019.3 and older

### Fixed

- Trello authentication fails due to whitespace in token
- Form gets stuck on screen during submission

## 1.5.0 - 2021-02-12

### Added

- Support for multiple labels on report

### Fixed

- Minor bug fixes

## 1.4.1 - 2021-01-11

### Fixed

- Form doesn't open in Editor when platform is set to Android

## 1.4.0 - 2020-12-08

### Added

- TMP version of Feedback prefab

### Fixed

- Suppress CS0618 warnings

## 1.3.1 - 2020-10-08

### Fixed

- `IOException` in build during screenshot capture

## 1.3.0 - 2019-11-18

### Changed

- Moved config menu location

### Removed

- Support for Unity 2017.3 and older

### Fixed

- Submission fails with vague error when summary field removed from form
- Support for Unity 2019+

## 1.2.0 - 2019-05-26

### Changed

- Moved asset to Plugins folder

### Fixed

- Slashes in board name break board dropdown menu in configuration

## 1.1.5 - 2018-10-12

### Fixed

- Compiler errors in 2017.2+

## 1.1.4 - 2018-10-06

### Fixed

- Invalid editor window errors
- Form doesn't open on Android

## 1.1.3 - 2018-05-13

### Fixed

- Boards fail to load after authentication

## 1.1.2 - 2018-03-18

### Changed

- Improve Trello API request timeout handling

### Fixed

- Use editor web window for authentication in Unity 2017

## 1.1.1 - 2018-01-09

### Fixed

- Deprecated Unity API calls in 2017.3

## 1.1.0 - 2017-11-26

### Added

- Markdown formatting helper

## 1.0.5 - 2017-11-12

### Fixed

- Use correct screenshot API for Unity 2017+

## 1.0.4 - 2017-09-26

### Fixed

- "Get Trello API Token" button sometimes focuses Unity Cloud Services window

## 1.0.3 - 2017-08-06

### Fixed

- Screenshots not captured on iOS

## 1.0.2 - 2017-07-14

### Fixed

- Trello authentication sometimes fails

## 1.0.1 - 2017-05-15

### Added

- Documentation PDF
- Demo scene

### Changed

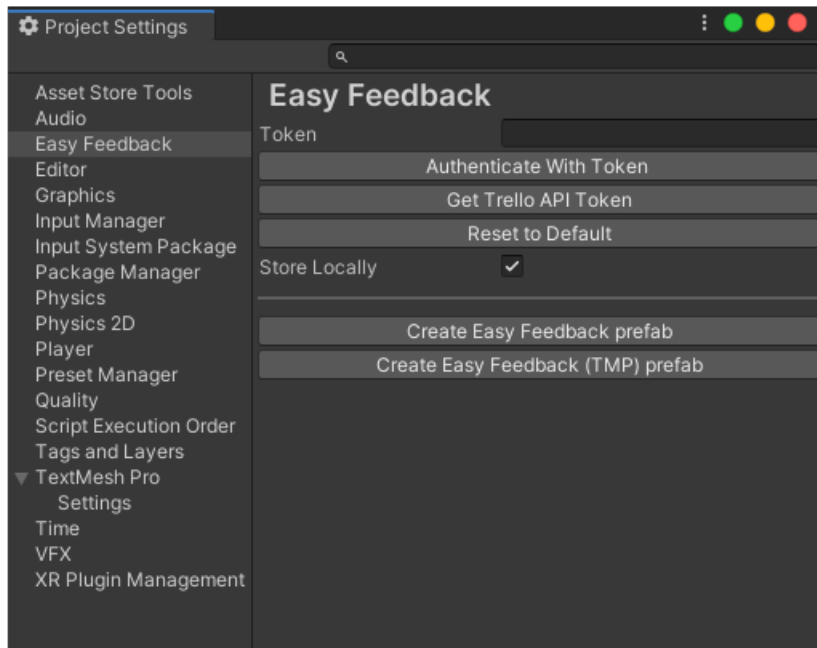
- Minor semantic changes

## 1.0.0 - 2017-04-28

- Initial release!  

# Getting started with Easy Feedback

## 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 settings at `Edit > Project Settings > Easy Feedback` in the toolbar, and click “Get Trello API Token.”

### Warning

It is highly recommended that you create a unique 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.

Authorize Trello

# Log in to Trello


Email (or username)

Password

**Log In** [Forgot your password? Reset it.](#)

[Log in with SSO](#)

**Did you sign up with your Google Account?**

 **Log In with Google**

Don't have an account? [Create a Trello account.](#)

[Tour](#) [Pricing](#) [Apps](#) [Jobs](#) [Blog](#) [Developers](#) [About](#) [Help](#) [Legal](#)

© Copyright 2017, Trello, Inc. All rights reserved.

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.

## 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.”

EasyFeedback.

Board Name

Create 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!

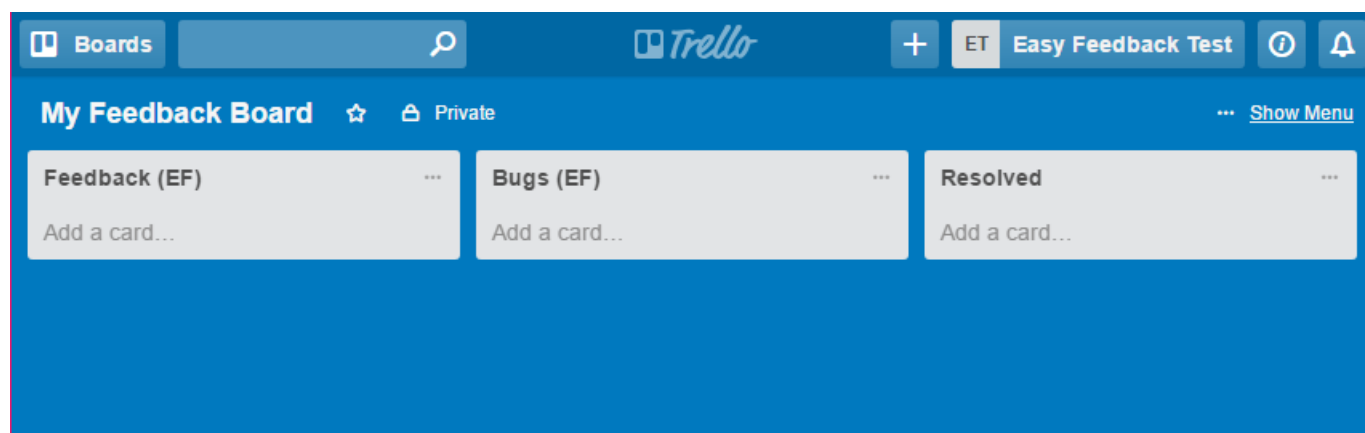


## Personal Boards

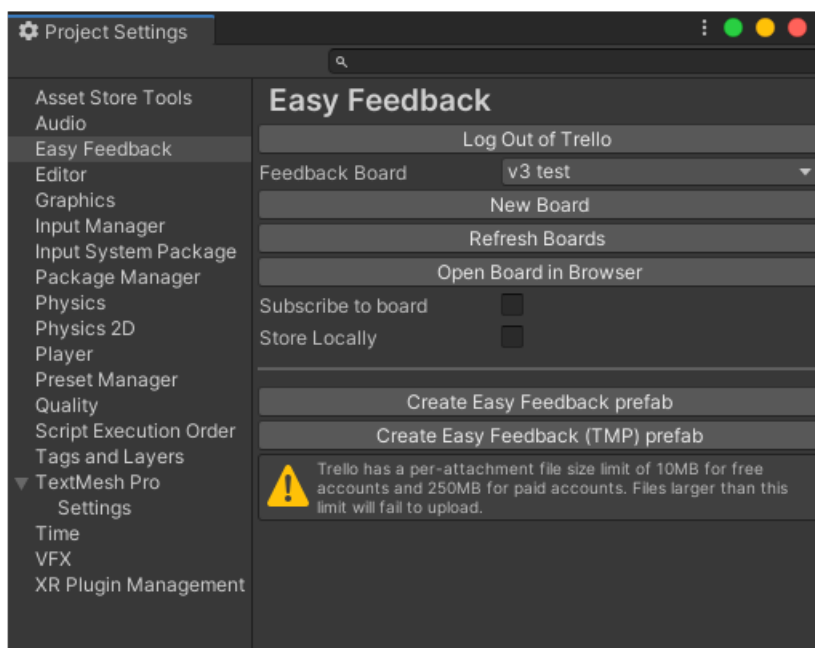
My Feedback Board

Create new board...

[Create a new team...](#)



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



## Creating the feedback prefab

Easy Feedback comes with a feedback form template implemented using either Unity UI text or TextMesh Pro.

To create a new copy of either of these templates, click either the "Create Easy Feedback prefab" or "Create Easy Feedback (TMP)" button on the Easy Feedback settings page, then select a location in your project to save the prefab.

## Adding the form to a scene

To add the form to a scene in your game, 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!

## Using the Input System

By default, Easy Feedback uses legacy Unity input but comes with support for the Unity Input system package. See [Input System support](#) to use the Input System with Easy Feedback.



# Configuring Easy Feedback

The Easy Feedback settings can be opened from `Edit > Project Settings > Easy Feedback` . Before authenticating with Trello, it will be mostly empty. For help authenticating with Trello, see [Authorizing with Trello](#).

## Log out of Trello

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

## New Board

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

## Refresh Boards

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

## Feedback Board

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

## 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.

## 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 when not authenticated with Trello. All others require authentication. See [Authorizing with Trello](#) for more help.

## Create Prefab buttons

The **Create Easy Feedback prefab** and **Create Easy Feedback (TMP) prefab** buttons help you quickly create an Easy Feedback Form prefab from the prefab templates in `Packages/Easy Feedback Form/Prefabs` . You should only need to use these the first time you set up your Easy Feedback Form prefab.

# 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.

## Anatomy of a Feedback Board

### 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 in the Easy Feedback settings.

### 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 [EasyFeedbackConfig](#) asset.

### 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.

## 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.

### 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 settings from [Edit -> Project Settings -> Easy Feedback](#) and click “Refresh Boards” to update the category name on your form.

### 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](#) and click “Refresh Boards” to update the categories on your form.

### 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](#) and click “Refresh Boards” to update the categories 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.

### 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` and click “Refresh Boards” to update the priority name on your form.

### 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` and click “Refresh Boards” to update the priorities on your form.

### 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` 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.

### Rearranging priorities

The order of the priorities in the dropdown can be changed by setting the `order` property of the label(s) in the `EasyFeedbackConfig.asset` file.

# 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.

## 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.

### Config

A reference to the auto-generated `EasyFeedbackConfig.asset`. This generally should not be changed.

### Include screenshot

Whether or not to include a screenshot with the report.

**Default:** *checked*

### Resize large screenshots

Trello has a [per-attachment file size limit of 10MB for free accounts and 250MB for paid accounts](#). This option resizes screenshots larger than 1080p to avoid the image size restriction. You can safely disable this if your account allows 250MB attachments.

**Default:** *checked*

### Form

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

### Events

See [FeedbackForm](#) for more detail on the events listed on the component.

## 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.

### Order of Priority Options

To change the order of the options in the priority dropdown, set the `order` property of the Labels in [EasyFeedbackConfig.asset](#). Lower values will appear higher in the list.

### 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.

### [?](#) Note

See also: [Report](#)

### 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 variables 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).

## 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` prefab.

## 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.

## Toasts

By default, Easy Feedback will send submission status messages via the `Toaster` attached to the Easy Feedback prefab. `Toaster.Toast(string)` is added as a callback on each of the submission [events](#).

### Customizing the Toast

You can customize the toast popup to your liking by modifying the `Toast` prefab.

# Input

Easy Feedback uses the legacy input API to detect input by default.

The asset also comes with out of the box support for Unity's [Input System](#) package. To configure Easy Feedback to use the Input System, see [Input System Support](#)

## Show/hide the form on input

By default, the feedback form is configured to be shown/hidden on keypress using the **Show Feedback Form Input** component. Feel free to reconfigure, remove and/or replace this component depending on your needs.

## Tab Next

The **Tab Next** component on some fields allows the player to use the tab key to jump to the next input field. Feel free to reconfigure,

remove and/or replace this component depending on your needs.

# Writing Custom Form Fields

Because every game is different, you may want to write a custom [FormField](#) to include 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](#).

## [?](#) Note

If you override the Awake method in [FormField](#), be sure to call `base.Awake()` so that the event listeners are properly registered.

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 `FeedbackForm` 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 `:ref: 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!");
```

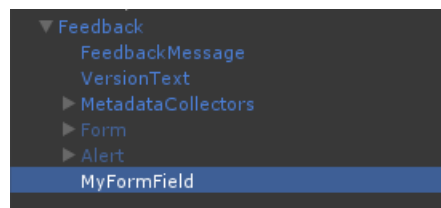
**Sections** on the report are referenced by name via the report's indexer. Here, we're getting the section we just added to the report, and setting its text contents to the string "Hello world!"

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 hierarchy, 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...](#)

# 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 necessary 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.

See the [API Documentation](#) for scripting reference.

See [Writing Custom Form Fields](#) for a quick guide to getting started writing our own custom fields.

# Input System Support

## Introduction

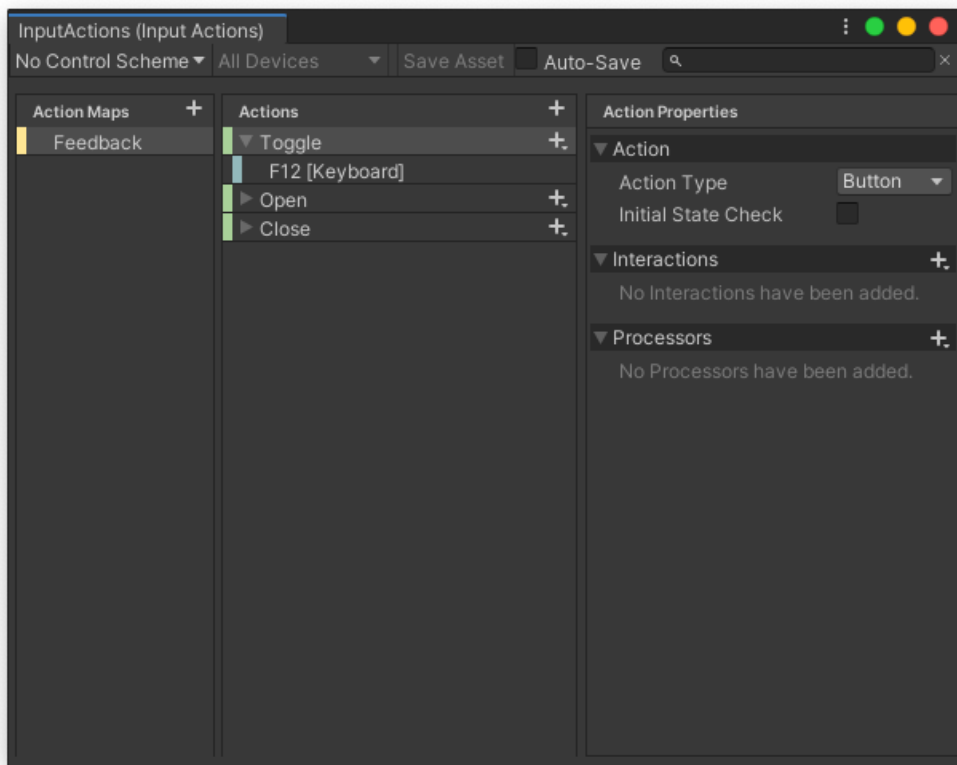
Easy Feedback comes with out of the box support for Unity's [Input System](#) package. By default, Easy Feedback uses legacy input via the **Show Feedback Form Input** and **Tab Next** components. These components must be replaced to use the Input System with Easy Feedback. The Input System support assemblies provide Input System versions of the default input components.

A wizard is also provided to make it easier to migrate a feedback prefab to the Input System.

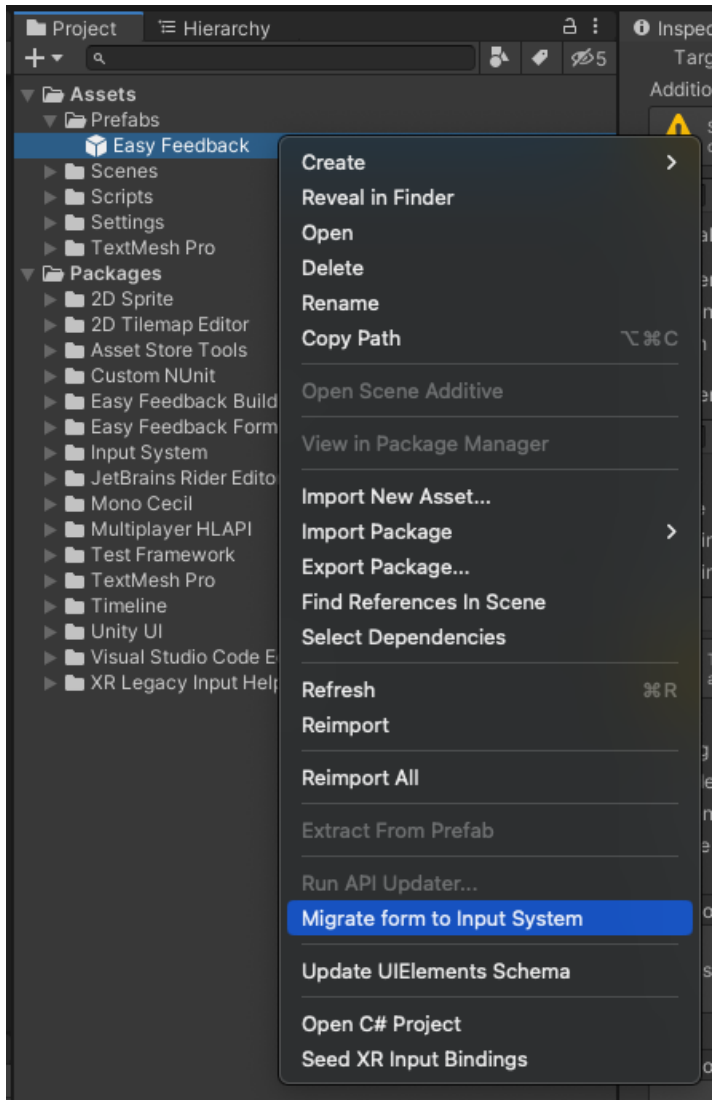
Input System support is automatically enabled by the `INPUT_SYSTEM_SUPPORT` preprocessor define when the Input System is enabled in your project. The Input System support assemblies can be found in `Packages/Easy Feedback Form/InputSystemSupport`.

## Setup

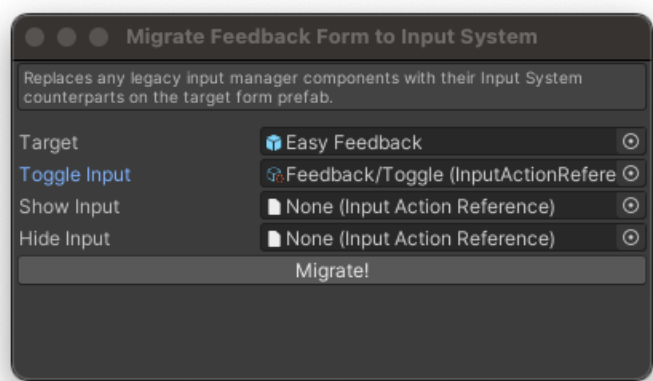
Be sure the Input System is installed and enabled in your project. See the [Input System installation guide](#) for more information.



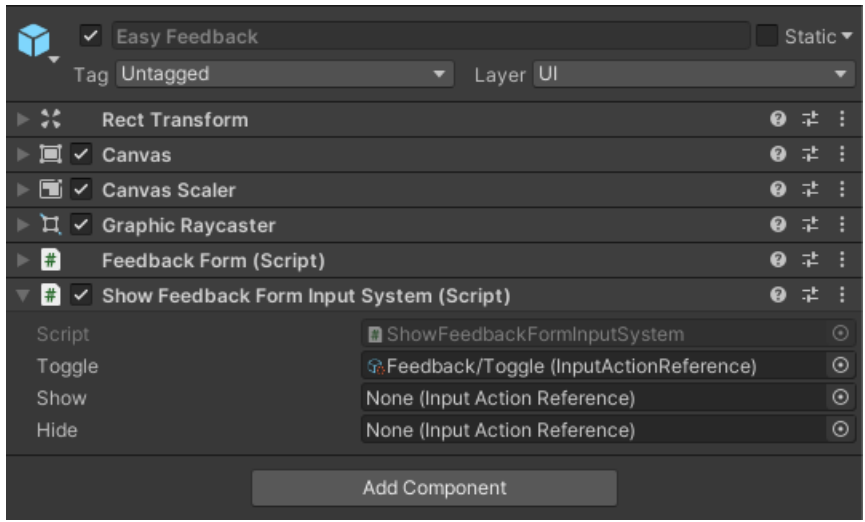
Set up Input Action(s) for the feedback form. You will need at least an input action for toggling the form or an action for showing and hiding the form respectively. How you set up your inputs is up to you! It is recommended that you set at least **hide** and **toggle** or **show**. See the [Input System documentation](#) for more detail on setting up actions.





Right click your feedback prefab and select "Migrate form to Input System." Select your input action(s) in the wizard window.



Click "Migrate!"



Select your form prefab and confirm that the **Show Feedback Form** component has been replaced with the **Show Feedback Form Input System** component.

That's it! Your feedback form should now work with the Input System.  

## Advanced use cases

The Input System components are intended to cover the most generic use-cases for integrating Easy Feedback. If your project has more advanced use-cases for the Input System and Easy Feedback, feel free to remove or replace the input components with your own code.

# Namespace AeLa.EasyFeedback

## Classes

### [EFConfig](#)

Configuration information for Easy Feedback

### [FeedbackBoard](#)

### [FeedbackForm](#)

### [FeedbackForm.SubmissionMessageEvent](#)

A submission event including a message

### [FeedbackText](#)

### [FormElement](#)

Parent class for any element that responds to the basic [FeedbackForm](#) events.

### [FormField](#)

Manages a field on the [FeedbackForm](#)

### [Report](#)

### [ReportSection](#)

## Enums

### [ScreenshotMode](#)

# Class EFConfig

Configuration information for Easy Feedback

Inheritance

System.Object

EFConfig

Namespace: [AeLa.EasyFeedback](#)

Assembly: cs.temp.dll.dll

Syntax

```
public class EFConfig : ScriptableObject
```

Constructors

EFConfig()

Declaration

```
public EFConfig()
```

Fields

Board

Declaration

```
public FeedbackBoard Board
```

Field Value

TYPE	DESCRIPTION
<a href="#">FeedbackBoard</a>	

StoreLocal

Declaration

```
public bool StoreLocal
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

Token

Declaration

```
public string Token
```

Field Value

TYPE	DESCRIPTION
System.String	

# Class FeedbackBoard

Inheritance

System.Object

FeedbackBoard

Namespace: [AeLa.EasyFeedback](#)

Assembly: cs.temp.dll.dll

Syntax

```
public class FeedbackBoard
```

## Fields

### CategoryIds

Declaration

```
public string[] CategoryIds
```

Field Value

TYPE	DESCRIPTION
System.String[]	

### CategoryNames

Declaration

```
public string[] CategoryNames
```

Field Value

TYPE	DESCRIPTION
System.String[]	

## Id

Declaration

```
public string Id
```

Field Value

TYPE	DESCRIPTION
System.String	

## Labels

Declaration

```
public Label[] Labels
```

Field Value

TYPE	DESCRIPTION
Label[]	



ListIds

Declaration

```
public string[] ListIds
```

Field Value

TYPE	DESCRIPTION
System.String[]	

ListNames

Declaration

```
public string[] ListNames
```

Field Value

TYPE	DESCRIPTION
System.String[]	

# Class FeedbackForm

Inheritance

System.Object

FeedbackForm

Namespace: [AeLa.EasyFeedback](#)

Assembly: cs.temp.dll.dll

Syntax

```
public class FeedbackForm : MonoBehaviour
```

## Fields

## Config

Declaration

```
public EFConfig Config
```

Field Value

TYPE	DESCRIPTION
<a href="#">EFConfig</a>	

## CurrentReport

The current report being built. Will be sent as next report

Declaration

```
public Report CurrentReport
```

Field Value

TYPE	DESCRIPTION
<a href="#">Report</a>	

## Form

Declaration

```
public Transform Form
```

Field Value

TYPE	DESCRIPTION
Transform	

## IncludeScreenshot

Declaration

```
public bool IncludeScreenshot
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

**OnFormClosed**

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

**Declaration**

```
public UnityEvent OnFormClosed
```

**Field Value**

TYPE	DESCRIPTION
UnityEvent	

**OnFormOpened**

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

**Declaration**

```
public UnityEvent OnFormOpened
```

**Field Value**

TYPE	DESCRIPTION
UnityEvent	

**OnFormSubmitted**

Called right before the report is sent to Trello, so additional information may be added.

**Declaration**

```
public UnityEvent OnFormSubmitted
```

**Field Value**

TYPE	DESCRIPTION
UnityEvent	

**OnSubmissionError**

Called to notify of any errors during submission

**Declaration**

```
public FeedbackForm.SubmissionMessageEvent OnSubmissionError
```

**Field Value**

TYPE	DESCRIPTION
FeedbackForm.SubmissionMessageEvent	

OnSubmissionFailed

Called if the submission fails

Declaration

public UnityEvent OnSubmissionFailed

Field Value

TYPE	DESCRIPTION
UnityEvent	

OnSubmissionSucceeded

Called when the submission has successfully completed

Declaration

public UnityEvent OnSubmissionSucceeded

Field Value

TYPE	DESCRIPTION
UnityEvent	

ResizeLargeScreenshots

Resizes screenshots larger than 1080p to help with Trello's filesize limit.

Declaration

public bool ResizeLargeScreenshots

Field Value

TYPE	DESCRIPTION
System.Boolean	

Remarks

Not supported in Legacy screenshot capture mode.

ScreenshotCaptureMode

Method used to capture the screenshot.

Declaration

public ScreenshotMode ScreenshotCaptureMode

Field Value

TYPE	DESCRIPTION
ScreenshotMode	

Properties

IsOpen

Whether or not the form is currently being displayed

Declaration

```
public bool IsOpen { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Methods

Awake()

Declaration

```
public void Awake()
```

DisableForm()

Disables all the Selectable elements on the form.

Declaration

```
public void DisableForm()
```

EnableForm()

Enables all the Selectable elements on the form.

Declaration

```
public void EnableForm()
```

Hide()

Hides the form, called by the Close button.

Declaration

```
public void Hide()
```

InitTrelloAPI()

Declaration

```
public void InitTrelloAPI()
```

Show()

Takes a screenshot, then opens the form

Declaration

```
public void Show()
```

Submit()

Called by the submit button, submits the form.

Declaration

```
public void Submit()
```

Toggle()

Toggles the open state of the form

Declaration

```
public void Toggle()
```

# Class FeedbackForm.SubmissionMessageEvent

A submission event including a message

## Inheritance

System.Object

FeedbackForm.SubmissionMessageEvent

Namespace: [AeLa.EasyFeedback](#)

Assembly: cs.temp.dll.dll

## Syntax

```
public class SubmissionMessageEvent : UnityEvent<string>
```

# Class FeedbackText

Inheritance

System.Object

FeedbackText

Namespace: [AeLa.EasyFeedback](#)

Assembly: cs.temp.dll.dll

Syntax

```
public class FeedbackText : MonoBehaviour
```

## Fields

## Form

Declaration

```
public FeedbackForm Form
```

Field Value

TYPE	DESCRIPTION
<a href="#">FeedbackForm</a>	

## Message

Declaration

```
public string Message
```

Field Value

TYPE	DESCRIPTION
System.String	



# Class FormElement

Parent class for any element that responds to the basic [FeedbackForm](#) events.

Inheritance

System.Object

FormElement

[CategoryDropdown](#)

[PriorityDropdown](#)

[ReportTitle](#)

[FormField](#)

Namespace: [AeLa.EasyFeedback](#)

Assembly: cs.temp.dll.dll

Syntax

```
public abstract class FormElement : MonoBehaviour
```

Fields

Form

The feedback form this component is a part of

Declaration

```
protected FeedbackForm Form
```

Field Value

TYPE	DESCRIPTION
<a href="#">FeedbackForm</a>	

Methods

Awake()

Declaration

```
public virtual void Awake()
```

FormClosed()

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

Declaration

```
protected abstract void FormClosed()
```

FormOpened()

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

Declaration

```
protected abstract void FormOpened()
```

FormSubmitted()

Called right before the report is sent to Trello

Declaration

protected abstract void FormSubmitted()

Remarks

Add user-provided data to your report here

# Class FormField

Manages a field on the [FeedbackForm](#)

For more help with FormFields, see [Custom Form Fields](#).

Inheritance

System.Object

[FormElement](#)

FormField

Inherited Members

[FormElement.Form](#)

[FormElement.FormOpened\(\)](#)

[FormElement.FormSubmitted\(\)](#)

[FormElement.FormClosed\(\)](#)

[FormElement.Awake\(\)](#)

Namespace: [AeLa.EasyFeedback](#)

Assembly: cs.temp.dll.dll

Syntax

```
public abstract class FormField : FormElement
```

Fields

SectionTitle

The title of this field's section on the report

Declaration

```
public string SectionTitle
```

Field Value

TYPE	DESCRIPTION
System.String	

SortOrder

Order of the section in the report (lowest first)

Declaration

```
public int SortOrder
```

Field Value

TYPE	DESCRIPTION
System.Int32	

# Class Report

Inheritance

System.Object

Report

Namespace: [AeLa.EasyFeedback](#)

Assembly: cs.temp.dll.dll

Syntax

```
public class Report
```

## Constructors

### Report()

Declaration

```
public Report()
```

## Fields

### Labels

Labels to add to the card on Trello

Declaration

```
public readonly List<Label> Labels
```

Field Value

TYPE	DESCRIPTION
List<Label>	

### List

Trello list this report will be added to

Declaration

```
public List List
```

Field Value

TYPE	DESCRIPTION
List	

### Title

The title of the card on Trello

Declaration

```
public string Title
```

Field Value

TYPE	DESCRIPTION
System.String	

Properties

Attachments

Additional files attached to this report

Declaration

```
public List<FileAttachment> Attachments { get; }
```

Property Value

TYPE	DESCRIPTION
List<FileAttachment>	

Remarks

Private to enforce Trello attachment limit (100)

Item[String]

Returns a section in the report by title

Declaration

```
public ReportSection this[string sectionTitle] { get; set; }
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	sectionTitle	

Property Value

TYPE	DESCRIPTION
<a href="#">ReportSection</a>	

Methods

AddLabel(Label)

Adds a label to the report.

Declaration

```
public void AddLabel(Label label)
```

Parameters

TYPE	NAME	DESCRIPTION
Label	label	

AddSection(ReportSection)

Adds a new section to the report

Declaration

```
public void AddSection(ReportSection section)
```

Parameters

TYPE	NAME	DESCRIPTION
ReportSection	section	

AddSection(String, Int32)

Adds a new empty section to the report

Declaration

```
public void AddSection(string title, int sortOrder = 0)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	title	The title of the section
System.Int32	sortOrder	The order of the section on the report (lowest first)

AttachFile(FileAttachment)

Attach a file to the report

Declaration

```
public void AttachFile(FileAttachment file)
```

Parameters

TYPE	NAME	DESCRIPTION
FileAttachment	file	

AttachFile(String, Byte[])

Attach a file to the report

Declaration

```
public void AttachFile(string name, byte[] data)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	The name of the file

TYPE	NAME	DESCRIPTION
Byte[]	data	The file data

**AttachFile(String, String)**

Attach a file to the report

**Declaration**

```
public void AttachFile(string name, string filePath)
```

**Parameters**

TYPE	NAME	DESCRIPTION
System.String	name	The name of the file
System.String	filePath	The path to the file

**GetLocalFileText()**

**Declaration**

```
public string GetLocalFileText()
```

**Returns**

TYPE	DESCRIPTION
System.String	

**HasLabel(Label)**

Checks if the report already has a label.

**Declaration**

```
public bool HasLabel(Label label)
```

**Parameters**

TYPE	NAME	DESCRIPTION
Label	label	

**Returns**

TYPE	DESCRIPTION
System.Boolean	

**HasSection(String)**

Checks whether the report already has a section

**Declaration**

public bool HasSection(string title)

Parameters

TYPE	NAME	DESCRIPTION
System.String	title	

Returns

TYPE	DESCRIPTION
System.Boolean	

RemoveSection(String)

Declaration

public void RemoveSection(string title)

Parameters

TYPE	NAME	DESCRIPTION
System.String	title	

ToString()

Returns the report formatted in markdown for Trello

Declaration

public override string ToString()

Returns

TYPE	DESCRIPTION
System.String	

Overrides

System.Object.ToString()



# Class ReportSection

Inheritance

System.Object

ReportSection

Namespace: [AeLa.EasyFeedback](#)

Assembly: cs.temp.dll.dll

Syntax

```
public class ReportSection
```

## Constructors

### ReportSection(String, Int32)

Creates a new report section with the specified title and sort order

Declaration

```
public ReportSection(string title, int sortOrder = 0)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	title	
System.Int32	sortOrder	

### ReportSection(String, String)

Creates a new report section with the specified title and text

Declaration

```
public ReportSection(string title, string text)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	title	
System.String	text	

## Fields

### SortOrder

The order of this element in the report (lowest first)

Declaration

```
public int SortOrder
```

Field Value

TYPE	DESCRIPTION
System.Int32	

**Title**

The title of this section

**Declaration**

```
public string Title
```

**Field Value**

TYPE	DESCRIPTION
System.String	

**Methods**

**Append(String)**

Appends text to the section text

**Declaration**

```
public void Append(string text)
```

**Parameters**

TYPE	NAME	DESCRIPTION
System.String	text	

**AppendLine(String)**

Appends a line to the section text

**Declaration**

```
public void AppendLine(string line)
```

**Parameters**

TYPE	NAME	DESCRIPTION
System.String	line	

**SetText(String)**

Replaces the existing section text with specified text

**Declaration**

```
public void SetText(string text)
```

**Parameters**

TYPE	NAME	DESCRIPTION
System.String	text	

**ToString()**

Returns the section in markdown formatting for Trello

**Declaration**

```
public override string ToString()
```

**Returns**

TYPE	DESCRIPTION
System.String	

**Overrides**

System.Object.ToString()

# Enum ScreenshotMode

Namespace: [AeLa.EasyFeedback](#)

Assembly: cs.temp.dll.dll

## Syntax

```
public enum ScreenshotMode
```

## Fields

NAME	DESCRIPTION
Legacy	Captures the screen using ScreenCapture.CaptureScreenshot. Use if screenshots are not captured correctly by <a href="#">Texture</a> mode.
Texture	Captures the screen to memory using Texture2D.ReadPixels.

# Namespace AeLa.EasyFeedback.APIs

## Classes

[AddCardResponse](#)

[Badges](#)

[CardLabel](#)

[Descdata](#)

[Emoji](#)

[Trello](#)

## Structs

[Board](#)

Board data returned from Trello API

[BoardCollection](#)

[Label](#)

[LabelCollection](#)

[LabelNames](#)

[List](#)

[ListCollection](#)

[Prefs](#)

Board preferences

[Subscribed](#)

Object for GETting the subscribed value Trello has an underscore on value here, annoying

## Enums

[AccessibilityLevel](#)

[CardAgeMode](#)

[Invitations](#)

[PermissionLevel](#)

# Enum AccessibilityLevel

Namespace: [AeLa.EasyFeedback.APIs](#)

Assembly: cs.temp.dll.dll

## Syntax

```
public enum AccessibilityLevel
```

## Fields

NAME	DESCRIPTION
disabled	
members	
observers	
org	
public	

# Class AddCardResponse

Inheritance

System.Object

AddCardResponse

Namespace: [AeLa.EasyFeedback.APIs](#)

Assembly: cs.temp.dll.dll

Syntax

```
public class AddCardResponse
```

## Fields

### badges

Declaration

```
public Badges badges
```

Field Value

TYPE	DESCRIPTION
<a href="#">Badges</a>	

### checkItemStates

Declaration

```
public bool[] checkItemStates
```

Field Value

TYPE	DESCRIPTION
System.Boolean[]	

### closed

Declaration

```
public bool closed
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

### dateLastActivity

Declaration

```
public DateTime dateLastActivity
```

Field Value

TYPE	DESCRIPTION
DateTime	

desc

Declaration

```
public string desc
```

Field Value

TYPE	DESCRIPTION
System.String	

descData

Declaration

```
public Descdata descData
```

Field Value

TYPE	DESCRIPTION
Descdata	

due

Declaration

```
public string due
```

Field Value

TYPE	DESCRIPTION
System.String	

dueComplete

Declaration

```
public bool dueComplete
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

email

Declaration

```
public string email
```

Field Value

TYPE	DESCRIPTION
System.String	

id



Declaration

```
public string id
```

Field Value

TYPE	DESCRIPTION
System.String	

idAttachmentCover

Declaration

```
public string idAttachmentCover
```

Field Value

TYPE	DESCRIPTION
System.String	

idBoard

Declaration

```
public string idBoard
```

Field Value

TYPE	DESCRIPTION
System.String	

idChecklists

Declaration

```
public string[] idChecklists
```

Field Value

TYPE	DESCRIPTION
System.String[]	

idLabels

Declaration

```
public string[] idLabels
```

Field Value

TYPE	DESCRIPTION
System.String[]	

idList

Declaration

```
public string idList
```

#### Field Value

TYPE	DESCRIPTION
System.String	

### idMembers

#### Declaration

```
public string[] idMembers
```

#### Field Value

TYPE	DESCRIPTION
System.String[]	

### idShort

#### Declaration

```
public int idShort
```

#### Field Value

TYPE	DESCRIPTION
System.Int32	

### labels

#### Declaration

```
public CardLabel[] labels
```

#### Field Value

TYPE	DESCRIPTION
CardLabel[]	

### manualCoverAttachment

#### Declaration

```
public bool manualCoverAttachment
```

#### Field Value

TYPE	DESCRIPTION
System.Boolean	

### name

#### Declaration

```
public string name
```

Field Value

TYPE	DESCRIPTION
System.String	

pos

Declaration

```
public int pos
```

Field Value

TYPE	DESCRIPTION
System.Int32	

shortUrl

Declaration

```
public string shortUrl
```

Field Value

TYPE	DESCRIPTION
System.String	

stickers

Declaration

```
public string[] stickers
```

Field Value

TYPE	DESCRIPTION
System.String[]	

url

Declaration

```
public string url
```

Field Value

TYPE	DESCRIPTION
System.String	

# Class Badges

Inheritance

System.Object

Badges

Namespace: [AeLa.EasyFeedback.APIs](#)

Assembly: cs.temp.dll.dll

Syntax

```
public class Badges
```

## Fields

### attachments

Declaration

```
public int attachments
```

Field Value

TYPE	DESCRIPTION
System.Int32	

### checkItems

Declaration

```
public int checkItems
```

Field Value

TYPE	DESCRIPTION
System.Int32	

### checkItemsChecked

Declaration

```
public int checkItemsChecked
```

Field Value

TYPE	DESCRIPTION
System.Int32	

### comments

Declaration

```
public int comments
```

Field Value

TYPE	DESCRIPTION
System.Int32	

description

Declaration

public bool description

Field Value

TYPE	DESCRIPTION
System.Boolean	

due

Declaration

public string due

Field Value

TYPE	DESCRIPTION
System.String	

dueComplete

Declaration

public bool dueComplete

Field Value

TYPE	DESCRIPTION
System.Boolean	

fogbugz

Declaration

public string fogbugz

Field Value

TYPE	DESCRIPTION
System.String	

subscribed

Declaration

public bool subscribed

Field Value

TYPE	DESCRIPTION
System.Boolean	

viewingMemberVoted

Declaration

public bool viewingMemberVoted

Field Value

TYPE	DESCRIPTION
System.Boolean	

votes

Declaration

public int votes

Field Value

TYPE	DESCRIPTION
System.Int32	

# Struct Board

Board data returned from Trello API

Namespace: [AeLa.EasyFeedback.APIs](#)

Assembly: cs.temp.dll.dll

## Syntax

```
public struct Board
```

## Fields

### closed

#### Declaration

```
public bool closed
```

#### Field Value

TYPE	DESCRIPTION
System.Boolean	

### desc

#### Declaration

```
public string desc
```

#### Field Value

TYPE	DESCRIPTION
System.String	

### descData

#### Declaration

```
public object descData
```

#### Field Value

TYPE	DESCRIPTION
System.Object	

### id

#### Declaration

```
public string id
```

#### Field Value

TYPE	DESCRIPTION
System.String	

### idOrganization

Declaration

public string idOrganization

Field Value

TYPE	DESCRIPTION
System.String	

labelNames

Declaration

public LabelNames labelNames

Field Value

TYPE	DESCRIPTION
LabelNames	

name

Declaration

public string name

Field Value

TYPE	DESCRIPTION
System.String	

pinned

Declaration

public bool pinned

Field Value

TYPE	DESCRIPTION
System.Boolean	

prefs

Declaration

public Prefs prefs

Field Value

TYPE	DESCRIPTION
Prefs	

shortUrl

Declaration



public string shortUrl

Field Value

TYPE	DESCRIPTION
System.String	

url

Declaration

public string url

Field Value

TYPE	DESCRIPTION
System.String	

# Struct BoardCollection

Namespace: [AeLa.EasyFeedback.APIs](#)

Assembly: cs.temp.dll.dll

## Syntax

```
public struct BoardCollection
```

## Fields

### boards

## Declaration

```
public Board[] boards
```

## Field Value

TYPE	DESCRIPTION
<a href="#">Board[]</a>	

# Enum CardAgeMode

Namespace: [AeLa.EasyFeedback.APIs](#)

Assembly: cs.temp.dll.dll

## Syntax

```
public enum CardAgeMode
```

## Fields

NAME	DESCRIPTION
pirate	
regular	

# Class CardLabel

Inheritance

System.Object

CardLabel

Namespace: [AeLa.EasyFeedback.APIs](#)

Assembly: cs.temp.dll.dll

Syntax

```
public class CardLabel
```

## Fields

### color

Declaration

```
public string color
```

Field Value

TYPE	DESCRIPTION
System.String	

### id

Declaration

```
public string id
```

Field Value

TYPE	DESCRIPTION
System.String	

### idBoard

Declaration

```
public string idBoard
```

Field Value

TYPE	DESCRIPTION
System.String	

### name

Declaration

```
public string name
```

Field Value

TYPE	DESCRIPTION
System.String	

uses

Declaration

```
public int uses
```

Field Value

TYPE	DESCRIPTION
System.Int32	

# Class Descdata

Inheritance

System.Object

Descdata

Namespace: [AeLa.EasyFeedback.APIs](#)

Assembly: cs.temp.dll.dll

Syntax

```
public class Descdata
```

## Fields

emoji

Declaration

```
public Emoji emoji
```

Field Value

TYPE	DESCRIPTION
<a href="#">Emoji</a>	

# Class Emoji

Inheritance

System.Object

Emoji

Namespace: [AeLa.EasyFeedback.APIs](#)

Assembly: cs.temp.dll.dll

Syntax

```
public class Emoji
```

# Enum Invitations

Namespace: [AeLa.EasyFeedback.APIs](#)

Assembly: cs.temp.dll.dll

## Syntax

```
public enum Invitations
```

## Fields

NAME	DESCRIPTION
admins	
members	



# Struct Label

Namespace: [AeLa.EasyFeedback.APIs](#)

Assembly: cs.temp.dll.dll

## Syntax

```
public struct Label
```

## Constructors

**Label**(String, String, String, String, Int32, Int32)

## Declaration

```
public Label(string id = null, string idBoard = null, string name = null, string color = null, int uses = 0, int order = 0)
```

## Parameters

TYPE	NAME	DESCRIPTION
System.String	id	
System.String	idBoard	
System.String	name	
System.String	color	
System.Int32	uses	
System.Int32	order	

## Fields

### color

## Declaration

```
public string color
```

## Field Value

TYPE	DESCRIPTION
System.String	

### id

## Declaration

```
public string id
```

## Field Value

TYPE	DESCRIPTION
System.String	

### idBoard

Declaration

public string idBoard

Field Value

TYPE	DESCRIPTION
System.String	

name

Declaration

public string name

Field Value

TYPE	DESCRIPTION
System.String	

order

Declaration

public int order

Field Value

TYPE	DESCRIPTION
System.Int32	

uses

Declaration

public int uses

Field Value

TYPE	DESCRIPTION
System.Int32	

# Struct LabelCollection

Namespace: [AeLa.EasyFeedback.APIs](#)

Assembly: cs.temp.dll.dll

## Syntax

```
public struct LabelCollection
```

## Fields

### labels

## Declaration

```
public Label[] labels
```

## Field Value

TYPE	DESCRIPTION
<a href="#">Label[]</a>	

# Struct LabelNames

Namespace: [AeLa.EasyFeedback.APIs](#)

Assembly: cs.temp.dll.dll

## Syntax

```
public struct LabelNames
```

## Fields

### blue

#### Declaration

```
public string blue
```

#### Field Value

TYPE	DESCRIPTION
System.String	

### green

#### Declaration

```
public string green
```

#### Field Value

TYPE	DESCRIPTION
System.String	

### lime

#### Declaration

```
public string lime
```

#### Field Value

TYPE	DESCRIPTION
System.String	

### orange

#### Declaration

```
public string orange
```

#### Field Value

TYPE	DESCRIPTION
System.String	

### purple

#### Declaration

public string purple

Field Value

TYPE	DESCRIPTION
System.String	

red

Declaration

public string red

Field Value

TYPE	DESCRIPTION
System.String	

sky

Declaration

public string sky

Field Value

TYPE	DESCRIPTION
System.String	

yellow

Declaration

public string yellow

Field Value

TYPE	DESCRIPTION
System.String	

# Struct List

Namespace: [AeLa.EasyFeedback.APIs](#)

Assembly: cs.temp.dll.dll

## Syntax

```
public struct List
```

## Fields

### closed

#### Declaration

```
public bool closed
```

#### Field Value

TYPE	DESCRIPTION
System.Boolean	

### id

#### Declaration

```
public string id
```

#### Field Value

TYPE	DESCRIPTION
System.String	

### idBoard

#### Declaration

```
public string idBoard
```

#### Field Value

TYPE	DESCRIPTION
System.String	

### name

#### Declaration

```
public string name
```

#### Field Value

TYPE	DESCRIPTION
System.String	

### pos

#### Declaration

public float pos

Field Value

TYPE	DESCRIPTION
System.Single	

subscribed

Declaration

public bool subscribed

Field Value

TYPE	DESCRIPTION
System.Boolean	

# Struct ListCollection

Namespace: [AeLa.EasyFeedback.APIs](#)

Assembly: cs.temp.dll.dll

## Syntax

```
public struct ListCollection
```

## Fields

### lists

## Declaration

```
public List[] lists
```

## Field Value

TYPE	DESCRIPTION
<a href="#">List[]</a>	



# Enum PermissionLevel

Namespace: [AeLa.EasyFeedback.APIs](#)

Assembly: cs.temp.dll.dll

## Syntax

```
public enum PermissionLevel
```

## Fields

NAME	DESCRIPTION
org	
private	
public	

# Struct Prefs

Board preferences

Namespace: [AeLa.EasyFeedback.APIs](#)

Assembly: cs.temp.dll.dll

Syntax

```
public struct Prefs
```

## Fields

### background

Declaration

```
public string background
```

Field Value

TYPE	DESCRIPTION
System.String	

### backgroundBrightness

Declaration

```
public string backgroundBrightness
```

Field Value

TYPE	DESCRIPTION
System.String	

### backgroundColor

Declaration

```
public string backgroundColor
```

Field Value

TYPE	DESCRIPTION
System.String	

### backgroundImage

Declaration

```
public object backgroundImage
```

Field Value

TYPE	DESCRIPTION
System.Object	

### backgroundImageScaled

Declaration

public object backgroundImageScaled

Field Value

TYPE	DESCRIPTION
System.Object	

backgroundTile

Declaration

public bool? backgroundTile

Field Value

TYPE	DESCRIPTION
System.Nullable<System.Boolean>	

calendarFeedEnabled

Declaration

public bool? calendarFeedEnabled

Field Value

TYPE	DESCRIPTION
System.Nullable<System.Boolean>	

canBeOrg

Declaration

public bool? canBeOrg

Field Value

TYPE	DESCRIPTION
System.Nullable<System.Boolean>	

canBePrivate

Declaration

public bool? canBePrivate

Field Value

TYPE	DESCRIPTION
System.Nullable<System.Boolean>	

canBePublic

Declaration

public bool? canBePublic

Field Value

TYPE	DESCRIPTION
System.Nullable<System.Boolean>	

canInvite

Declaration

public bool? canInvite

Field Value

TYPE	DESCRIPTION
System.Nullable<System.Boolean>	

cardAging

Declaration

public CardAgeMode? cardAging

Field Value

TYPE	DESCRIPTION
System.Nullable<CardAgeMode>	

cardCovers

Declaration

public bool? cardCovers

Field Value

TYPE	DESCRIPTION
System.Nullable<System.Boolean>	

comments

Declaration

public AccessibilityLevel? comments

Field Value

TYPE	DESCRIPTION
System.Nullable<AccessibilityLevel>	

invitations

Declaration

public Invitations? invitations

Field Value

TYPE	DESCRIPTION
System.Nullable<Invitations>	

permissionLevel

Declaration

public PermissionLevel? permissionLevel
---

Field Value

TYPE	DESCRIPTION
System.Nullable<PermissionLevel>	

selfJoin

Declaration

public bool? selfJoin
-----------------------

Field Value

TYPE	DESCRIPTION
System.Nullable<System.Boolean>	

voting

Declaration

public AccessibilityLevel? voting
-----------------------------------

Field Value

TYPE	DESCRIPTION
System.Nullable<AccessibilityLevel>	

# Struct Subscribed

Object for GETting the subscribed value Trello has an underscore on value here, annoying

Namespace: [AeLa.EasyFeedback.APIs](#)

Assembly: cs.temp.dll.dll

## Syntax

```
public struct Subscribed
```

## Fields

**\_value**

## Declaration

```
public bool _value
```

## Field Value

TYPE	DESCRIPTION
System.Boolean	

# Class Trello

Inheritance

System.Object

Trello

Namespace: [AeLa.EasyFeedback.APIs](#)

Assembly: cs.temp.dll.dll

Syntax

```
public class Trello
```

## Constructors

### Trello(String)

Declaration

```
public Trello(string token)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	token	

## Fields

### ApiUri

Declaration

```
public const string ApiUri = "https://trello.com/1"
```

Field Value

TYPE	DESCRIPTION
System.String	

### AppKey

Declaration

```
public const string AppKey = "9babe077311b8a24fddaebb73de1df6a"
```

Field Value

TYPE	DESCRIPTION
System.String	

### CategoryTag

Declaration

```
public const string CategoryTag = "(EF)"
```

Field Value

TYPE	DESCRIPTION
System.String	

### ErrorMessage

Declaration

public string ErrorMessage
----------------------------

Field Value

TYPE	DESCRIPTION
System.String	

### IsDoneUploading

Declaration

public bool IsDoneUploading
-----------------------------

Field Value

TYPE	DESCRIPTION
System.Boolean	

### LastAddCardResponse

Declaration

public AddCardResponse LastAddCardResponse
--

Field Value

TYPE	DESCRIPTION
AddCardResponse	

### LastRequest

Declaration

public UnityWebRequest LastRequest
------------------------------------

Field Value

TYPE	DESCRIPTION
UnityWebRequest	

### MaxCharLength

Declaration

public const int MaxCharLength = 16384
--

Field Value



TYPE	DESCRIPTION
System.Int32	

TemplateBoardID

Declaration

```
public const string TemplateBoardID = "589d1b02a4856195b7cc31c9"
```

Field Value

TYPE	DESCRIPTION
System.String	

UploadError

Declaration

```
public bool UploadError
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

UploadException

Declaration

```
public Exception UploadException
```

Field Value

TYPE	DESCRIPTION
Exception	

Properties

AuthURL

Declaration

```
public static string AuthURL { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Methods

AddAttachmentAsync(String, Byte[], String, String, String)

Declaration

```
public IEnumerator AddAttachmentAsync(string cardID, byte[] file = null, string url = null, string name = null, string mimeType = null)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	cardID	
Byte[]	file	
System.String	url	
System.String	name	
System.String	contentType	

Returns

TYPE	DESCRIPTION
IEnumerator	

AddBoard(String, Boolean, Boolean, String, String, String, String, String, Nullable<Prefs>)

Editor-safe method for adding a board

Declaration

```
public Board AddBoard(string name, bool defaultLabels = true, bool defaultLists = true, string desc = null, string idOrganization = null, string idBoardSource = null, string keepFromSource = "all", string powerUps = "all", Prefs? prefs = default(Prefs? ))
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	
System.Boolean	defaultLabels	
System.Boolean	defaultLists	
System.String	desc	
System.String	idOrganization	
System.String	idBoardSource	
System.String	keepFromSource	
System.String	powerUps	
System.Nullable<Prefs>	prefs	

Returns

TYPE	DESCRIPTION
Board	

AddCard(String, String, IEnumerable<Label>, String, Byte[])

Adds a card to a board

Declaration

```
public IEnumerator AddCard(string name, string description, IEnumerable<Label> labels, string list, byte[] fileSource = null)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	Title of the card
System.String	description	Description of the card
IEnumerable<Label>	labels	Any labels on the card
System.String	list	The list the card belongs to
Byte[]	fileSource	File data to attach to the card

Returns

TYPE	DESCRIPTION
IEnumerator	

GetBoards()

Editor-safe method for getting the boards on the authorized Trello account

Declaration

```
public Board[] GetBoards()
```

Returns

TYPE	DESCRIPTION
Board[]	

GetBoardsAsync(Action<Board[]>)

Declaration

```
public IEnumerator GetBoardsAsync(Action<Board[]> onFinished)
```

Parameters

TYPE	NAME	DESCRIPTION
Action<Board[]>	onFinished	

Returns

TYPE	DESCRIPTION
IEnumerator	

GetLabels(String)

Editor-safe method for getting labels from a board

Declaration

```
public Label[] GetLabels(string boardID)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	boardID	

Returns

TYPE	DESCRIPTION
Label[]	

GetLabelsAsync(String, Action<Label[]>)

Declaration

```
public IEnumerator GetLabelsAsync(string boardID, Action<Label[]> onFinished)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	boardID	
Action<Label[]>	onFinished	

Returns

TYPE	DESCRIPTION
IEnumerator	

GetLists(String)

Editor-safe method for getting the lists on a board

Declaration

```
public List[] GetLists(string boardID)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	boardID	

Returns

TYPE	DESCRIPTION
List[]	

**GetListsAsync(String, Action<List[]>)**

**Declaration**

```
public IEnumerator GetListsAsync(string boardID, Action<List[]> onFinished)
```

**Parameters**

TYPE	NAME	DESCRIPTION
System.String	boardID	
Action<List[]>	onFinished	

**Returns**

TYPE	DESCRIPTION
IEnumerator	

**GetSubscribed(String)**

Returns whether or not the authenticated user is subscribed to a board

**Declaration**

```
public bool GetSubscribed(string boardID)
```

**Parameters**

TYPE	NAME	DESCRIPTION
System.String	boardID	The board

**Returns**

TYPE	DESCRIPTION
System.Boolean	Whether or not the authenticated user is subscribed to the board

**GetURI(String)**

Returns a fully formed and authenticated request URI for the Trello API path provided

**Declaration**

```
public string GetURI(string apiPath)
```

**Parameters**

TYPE	NAME	DESCRIPTION
System.String	apiPath	The Trello API endpoint path (starting with /)

Returns

TYPE	DESCRIPTION
System.String	

IsValidToken(String, Boolean)

Checks if a token is valid

Declaration

```
public static bool IsValidToken(string token, bool silent = false)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	token	
System.Boolean	silent	

Returns

TYPE	DESCRIPTION
System.Boolean	

PutSubscribed(String, Boolean)

Sets a user's subscribed state for a board

Declaration

```
public void PutSubscribed(string boardID, bool value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	boardID	The board
System.Boolean	value	The subscribed state

# Namespace AeLa.EasyFeedback.FormElements

## Classes

[CategoryDropdown](#)

[PriorityDropdown](#)

[ReportTitle](#)

# Class CategoryDropdown

Inheritance

System.Object

[FormElement](#)

CategoryDropdown

Inherited Members

[FormElement.Form](#)

Namespace: [AeLa.EasyFeedback.FormElements](#)

Assembly: cs.temp.dll.dll

Syntax

```
public class CategoryDropdown : FormElement
```

## Methods

### Awake()

Declaration

```
public override void Awake()
```

Overrides

[FormElement.Awake\(\)](#)

### FormClosed()

Declaration

```
protected override void FormClosed()
```

Overrides

[FormElement.FormClosed\(\)](#)

### FormOpened()

Declaration

```
protected override void FormOpened()
```

Overrides

[FormElement.FormOpened\(\)](#)

### FormSubmitted()

Declaration

```
protected override void FormSubmitted()
```

Overrides

[FormElement.FormSubmitted\(\)](#)



# Class PriorityDropdown

Inheritance

System.Object

[FormElement](#)

PriorityDropdown

Inherited Members

[FormElement.Form](#)

Namespace: [AeLa.EasyFeedback.FormElements](#)

Assembly: cs.temp.dll.dll

Syntax

```
public class PriorityDropdown : FormElement
```

## Methods

### Awake()

Declaration

```
public override void Awake()
```

Overrides

[FormElement.Awake\(\)](#)

### FormClosed()

Declaration

```
protected override void FormClosed()
```

Overrides

[FormElement.FormClosed\(\)](#)

### FormOpened()

Declaration

```
protected override void FormOpened()
```

Overrides

[FormElement.FormOpened\(\)](#)

### FormSubmitted()

Declaration

```
protected override void FormSubmitted()
```

Overrides

[FormElement.FormSubmitted\(\)](#)

# Class ReportTitle

Inheritance

System.Object

[FormElement](#)

ReportTitle

Inherited Members

[FormElement.Form](#)

[FormElement.Awake\(\)](#)

Namespace: [AeLa.EasyFeedback.FormElements](#)

Assembly: cs.temp.dll.dll

Syntax

```
public class ReportTitle : FormElement
```

Methods

FormClosed()

Declaration

```
protected override void FormClosed()
```

Overrides

[FormElement.FormClosed\(\)](#)

FormOpened()

Declaration

```
protected override void FormOpened()
```

Overrides

[FormElement.FormOpened\(\)](#)

FormSubmitted()

Declaration

```
protected override void FormSubmitted()
```

Overrides

[FormElement.FormSubmitted\(\)](#)

# Namespace AeLa.EasyFeedback.FormInput

## Classes

[ShowFeedbackFormInput](#)

[TabNext](#)

[TabNextBase](#)

## Interfaces

[IToggleFormInput](#)

# Interface IToggleFormInput

Namespace: [AeLa.EasyFeedback.FormInput](#)

Assembly: cs.temp.dll.dll

## Syntax

```
public interface IToggleFormInput
```

## Properties

### Descriptor

User-readable description of the input

### Declaration

```
string Descriptor { get; }
```

### Property Value

TYPE	DESCRIPTION
System.String	

# Class ShowFeedbackFormInput

Inheritance

System.Object  
ShowFeedbackFormInput

Implements

[IToggleFormInput](#)

Namespace: **AeLa.EasyFeedback.FormInput**

Assembly: cs.temp.dll.dll

Syntax

```
public class ShowFeedbackFormInput : MonoBehaviour, IToggleFormInput
```

Fields

HideKey

Key used to hide the feedback form

Declaration

```
public KeyCode HideKey
```

Field Value

TYPE	DESCRIPTION
KeyCode	

ShowKey

Key used to show the feedback form

Declaration

```
public KeyCode ShowKey
```

Field Value

TYPE	DESCRIPTION
KeyCode	

ToggleKey

Key used to toggle the feedback form

Declaration

```
public KeyCode ToggleKey
```

Field Value

TYPE	DESCRIPTION
KeyCode	

Properties

Descriptor

Declaration

```
public string Descriptor { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Implements

[IToggleFormInput](#)

# Class TabNext

Inheritance

System.Object

TabNextBase

TabNext

Inherited Members

TabNextBase.Next

TabNextBase.Previous

TabNextBase.input

TabNextBase.nextInput

TabNextBase.previousInput

TabNextBase.Start()

TabNextBase.Select(Selectable)

TabNextBase.TryGetInputField(Selectable, IInputField)

TabNextBase.GetInputField(Selectable)

TabNextBase.Copy(TabNextBase)

Namespace: AeLa.EasyFeedback.FormInput

Assembly: cs.temp.dll.dll

Syntax

```
public class TabNext : TabNextBase
```

# Class TabNextBase

## Inheritance

System.Object

TabNextBase

[TabNext](#)

Namespace: [AeLa.EasyFeedback.FormInput](#)

Assembly: cs.temp.dll.dll

## Syntax

```
public abstract class TabNextBase : MonoBehaviour
```

## Fields

### input

Attached InputField (TMP or Unity)

## Declaration

```
protected IInputField input
```

## Field Value

TYPE	DESCRIPTION
IInputField	

## Next

## Declaration

```
public Selectable Next
```

## Field Value

TYPE	DESCRIPTION
Selectable	

### nextInput

## Declaration

```
protected IInputField nextInput
```

## Field Value

TYPE	DESCRIPTION
IInputField	

## Previous

## Declaration

```
public Selectable Previous
```

## Field Value



TYPE	DESCRIPTION
Selectable	

previousInput

Declaration

<code>protected</code> <code>InputField</code> previousInput
--

Field Value

TYPE	DESCRIPTION
InputField	

Methods

Copy(TabNextBase)

Copies properties from `other` to this instance

Declaration

<code>public virtual void</code> Copy(TabNextBase other)
--

Parameters

TYPE	NAME	DESCRIPTION
<code>TabNextBase</code>	other	

GetInputField(Selectable)

Declaration

<code>protected</code> <code>InputField</code> GetInputField(Selectable selectable)
---

Parameters

TYPE	NAME	DESCRIPTION
Selectable	selectable	

Returns

TYPE	DESCRIPTION
InputField	

Select(Selectable)

Declaration

<code>protected virtual void</code> Select(Selectable selectable)
---

Parameters

TYPE	NAME	DESCRIPTION
Selectable	selectable	

**Start()**

**Declaration**

```
protected virtual void Start()
```

**TryGetInputField(Selectable, out IInputField)**

**Declaration**

```
protected bool TryGetInputField(Selectable selectable, out IInputField field)
```

**Parameters**

TYPE	NAME	DESCRIPTION
Selectable	selectable	
IInputField	field	

**Returns**

TYPE	DESCRIPTION
System.Boolean	

# Namespace AeLa.EasyFeedback.InputSystemSupport

## Classes

[ShowFeedbackFormInputSystem](#)

[TabNextInputSystem](#)

A drop-in replacement for the component using the Input System.

# Class ShowFeedbackFormInputSystem

Inheritance

System.Object  
ShowFeedbackFormInputSystem

Implements

IToggleFormInput

Namespace: **AeLa.EasyFeedback.InputSystemSupport**

Assembly: cs.temp.dll.dll

Syntax

```
public class ShowFeedbackFormInputSystem : MonoBehaviour, IToggleFormInput
```

Fields

Hide

Input action used to hide the feedback form

Declaration

```
public InputActionReference Hide
```

Field Value

TYPE	DESCRIPTION
InputActionReference	

Show

Input action used to show the feedback form

Declaration

```
public InputActionReference Show
```

Field Value

TYPE	DESCRIPTION
InputActionReference	

Toggle

Input action used to toggle the feedback form

Declaration

```
public InputActionReference Toggle
```

Field Value

TYPE	DESCRIPTION
InputActionReference	

Properties

Descriptor

Declaration

```
public string Descriptor { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Implements

IToggleFormInput

# Class TabNextInputSystem

A drop-in replacement for the `TabNextBase` component using the Input System.

## Inheritance

System.Object

TabNextInputSystem

Namespace: [AeLa.EasyFeedback.InputSystemSupport](#)

Assembly: cs.temp.dll.dll

## Syntax

```
public class TabNextInputSystem : TabNextBase
```

## Remarks

This is just a simple helper component with hard-coded input bindings for tab/shift. You may use this as an example if you need something more bespoke. Always feel free to reach out to our support email if you'd like help extending Easy Feedback!

# Namespace AeLa.EasyFeedback.InputSystemSupport.Editor

## Classes

[InputSystemMigration](#)

[MigrateFeedbackFormWizard](#)

[MigrationMenu](#)

# Class InputSystemMigration

Inheritance

System.Object

InputSystemMigration

Namespace: [AeLa.EasyFeedback.InputSystemSupport.Editor](#)

Assembly: cs.temp.dll.dll

Syntax

```
public static class InputSystemMigration
```

Methods

MigrateTarget(GameObject, (InputActionReference, InputActionReference, InputActionReference))

Migrates relevant components on the target GameObject and all of its children to the new input system versions.

Declaration

```
public static void MigrateTarget(GameObject target, (InputActionReference, InputActionReference, InputActionReference) showFormInputActions)
```

Parameters

TYPE	NAME	DESCRIPTION
GameObject	target	
<a href="#">ValueTuple</a> <InputActionReference, InputActionReference, InputActionReference>	showFormInputActions	InputActionReferences for the <a href="#">ShowFeedbackFormInputSystem</a> component



# Class MigrateFeedbackFormWizard

Inheritance

System.Object

MigrateFeedbackFormWizard

Namespace: [AeLa.EasyFeedback.InputSystemSupport.Editor](#)

Assembly: cs.temp.dll.dll

Syntax

```
public class MigrateFeedbackFormWizard : EditorWindow
```

Fields

Target

The target feedback prefab

Declaration

```
public GameObject Target
```

Field Value

TYPE	DESCRIPTION
GameObject	

Methods

GetWindow()

Declaration

```
public static MigrateFeedbackFormWizard GetWindow()
```

Returns

TYPE	DESCRIPTION
<a href="#">MigrateFeedbackFormWizard</a>	

# Class MigrationMenu

Inheritance

System.Object

MigrationMenu

Namespace: [AeLa.EasyFeedback.InputSystemSupport.Editor](#)

Assembly: cs.temp.dll.dll

Syntax

```
public static class MigrationMenu
```

# Namespace AeLa.EasyFeedback.UI.Toaster

## Classes

[Toast](#)

[Toaster](#)

Displays [Toast\(String\)](#).

## Enums

[Toaster.PopoutDirection](#)

[Toaster.ToastAnchor](#)

# Class Toast

Inheritance

System.Object

Toast

Namespace: [AeLa.EasyFeedback.UI.Toaster](#)

Assembly: cs.temp.dll.dll

Syntax

```
public class Toast : MonoBehaviour
```

## Fields

## Text

Declaration

```
protected GameObject Text
```

Field Value

TYPE	DESCRIPTION
GameObject	

## Properties

## Message

Declaration

```
public string Message { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.String	

## RectTransform

Declaration

```
public RectTransform RectTransform { get; }
```

Property Value

TYPE	DESCRIPTION
RectTransform	

# Class Toaster

Displays [Toast\(String\)](#).

By default, Easy Feedback will send submission status messages via the Toaster attached to the Easy Feedback prefab. `Toaster.Toast(string)` is added as a callback on each of the submission [events](#).

## Customizing the Toast

You can customize the toast popup to your liking by modifying the `Toast` prefab.

### Inheritance

System.Object

Toaster

Namespace: [AeLa.EasyFeedback.UI.Toaster](#)

Assembly: cs.temp.dll.dll

### Syntax

```
public class Toaster : MonoBehaviour
```

### Fields

#### AnimationDuration

How long (seconds) the slide in/out animation takes

#### Declaration

```
protected float AnimationDuration
```

#### Field Value

TYPE	DESCRIPTION
System.Single	

#### Duration

How long (seconds) a message remains on screen

#### Declaration

```
protected float Duration
```

#### Field Value

TYPE	DESCRIPTION
System.Single	

#### PopupDirection

Direction the toast will move when it appears

#### Declaration

```
protected Toaster.PopoutDirection PopupDirection
```

#### Field Value

TYPE	DESCRIPTION
<a href="#">Toaster.PopoutDirection</a>	

### ToastPrefab

The toast prefab object

#### Declaration

<b>protected</b> Toast ToastPrefab
------------------------------------

#### Field Value

TYPE	DESCRIPTION
<a href="#">Toast</a>	

### ViewportAnchor

Where the toast will appear on screen

#### Declaration

<b>protected</b> Toaster.ToastAnchor ViewportAnchor
---

#### Field Value

TYPE	DESCRIPTION
<a href="#">Toaster.ToastAnchor</a>	

### Methods

#### Toast(String)

Displays a toast with the provided message

#### Declaration

<b>public void</b> Toast( <b>string</b> message)
--

#### Parameters

TYPE	NAME	DESCRIPTION
System.String	message	

# Enum Toaster.PopoutDirection

Namespace: [AeLa.EasyFeedback.UI.Toaster](#)

Assembly: cs.temp.dll.dll

## Syntax

```
public enum PopoutDirection
```

## Fields

NAME	DESCRIPTION
Down	
Left	
Right	
Up	

# Enum Toaster.ToastAnchor

Namespace: [AeLa.EasyFeedback.UI.Toaster](#)

Assembly: cs.temp.dll.dll

## Syntax

```
public enum ToastAnchor
```

## Fields

NAME	DESCRIPTION
BottomLeft	
BottomRight	
TopLeft	
TopRight	



# Namespace AeLa.EasyFeedback.Utility

## Classes

[FileAttachment](#)

[Markdown](#)

[ScreenshotUtil](#)

[SetSelectedOnOpen](#)

[SetVersionText](#)

[ShowHideMouse](#)

Shows or hides the mouse when the feedback form is opened or closed. Remove this component from your form if you do not want the mouse to be automatically managed.

## Enums

[Markdown.HeaderLevel](#)

# Class FileAttachment

Inheritance

System.Object

FileAttachment

Namespace: [AeLa.EasyFeedback.Utility](#)

Assembly: cs.temp.dll.dll

Syntax

```
public class FileAttachment
```

## Constructors

### FileAttachment(String, Byte[], String)

Creates a new instance of the FileAttachment object

Declaration

```
public FileAttachment(string name, byte[] data, string mimeType = null)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	The name of the attachment
<a href="#">Byte[]</a>	data	The file data
System.String	mimeType	The MIME type of the file

### FileAttachment(String, String)

Creates a new instance of the FileAttachment object

Declaration

```
public FileAttachment(string filePath, string mimeType = null)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	filePath	The path to the file
System.String	mimeType	The MIME type of the file

### FileAttachment(String, String, String)

Creates a new instance of the FileAttachment object

Declaration

```
public FileAttachment(string name, string filePath, string mimeType = null)
```

**Parameters**

TYPE	NAME	DESCRIPTION
System.String	name	The name of the attachment
System.String	filePath	The path to the file
System.String	mimeType	The MIME type of the file

**Properties**

**Data**

Attached file data

**Declaration**

```
public byte[] Data { get; set; }
```

**Property Value**

TYPE	DESCRIPTION
Byte[]	

**MimeType**

The MIME type for this file

**Declaration**

```
public string MimeType { get; set; }
```

**Property Value**

TYPE	DESCRIPTION
System.String	

**Name**

The name of the file attachment (0 to 256 characters).

**Declaration**

```
public string Name { get; set; }
```

**Property Value**

TYPE	DESCRIPTION
System.String	

# Class Markdown

Inheritance

System.Object

Markdown

Namespace: [AeLa.EasyFeedback.Utility](#)

Assembly: cs.temp.dll.dll

Syntax

```
public static class Markdown
```

## Fields

### HR

Creates a horizontal rule or line

Declaration

```
public const string HR = "----"
```

Field Value

TYPE	DESCRIPTION
System.String	

### LINE\_BREAK

Creates a new paragraph

Declaration

```
public const string LINE_BREAK = "\n\n"
```

Field Value

TYPE	DESCRIPTION
System.String	

## Methods

### Blockquote(String)

Creates a block of quoted text

Declaration

```
public static string Blockquote(string text)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	text	The text

Returns

TYPE	DESCRIPTION
System.String	

**Code(String)**

Creates an inline span of preformatted text

Declaration

```
public static string Code(string text)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	text	The text

Returns

TYPE	DESCRIPTION
System.String	

**CodeBlock(String, String)**

Creates a block of preformatted text

Declaration

```
public static string CodeBlock(string text, string language = "")
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	text	The text
System.String	language	The language for syntax highlighting (where supported)

Returns

TYPE	DESCRIPTION
System.String	

**Em(String)**

Formats the text with emphasis/italics

Declaration

```
public static string Em(string text)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	text	The text to be emphasized

#### Returns

TYPE	DESCRIPTION
System.String	

### H1(String)

Creates a first-level header from the specified text

#### Declaration

```
public static string H1(string text)
```

#### Parameters

TYPE	NAME	DESCRIPTION
System.String	text	The header text

#### Returns

TYPE	DESCRIPTION
System.String	

### H2(String)

Creates a second-level header from the specified text

#### Declaration

```
public static string H2(string text)
```

#### Parameters

TYPE	NAME	DESCRIPTION
System.String	text	The header text

#### Returns

TYPE	DESCRIPTION
System.String	

### H3(String)

Creates a third-level header from the specified text

#### Declaration

```
public static string H3(string text)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	text	The header text

Returns

TYPE	DESCRIPTION
System.String	

H4(String)

Creates a fourth-level header from the specified text

Declaration

```
public static string H4(string text)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	text	The header text

Returns

TYPE	DESCRIPTION
System.String	

H5(String)

Creates a fifth-level header from the specified text

Declaration

```
public static string H5(string text)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	text	The header text

Returns

TYPE	DESCRIPTION
System.String	

H6(String)

Creates a sixth-level header from the specified text

Declaration

```
public static string H6(string text)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	text	The header text

Returns

TYPE	DESCRIPTION
System.String	

Header(String, Markdown.HeaderLevel)

Creates a header from the specified text, with the specified level

Declaration

```
public static string Header(string text, Markdown.HeaderLevel level = Markdown.HeaderLevel.H1)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	text	The header text
Markdown.HeaderLevel	level	The header level

Returns

TYPE	DESCRIPTION
System.String	

Hyperlink(String, String)

Creates an inline link

Declaration

```
public static string Hyperlink(string text, string url)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	text	The link text



TYPE	NAME	DESCRIPTION
System.String	url	The link url

Returns

TYPE	DESCRIPTION
System.String	

Image(String, String)

Creates an inline image

Declaration

```
public static string Image(string url, string alt = "")
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	url	The url of the image
System.String	alt	The alt-text for the image

Returns

TYPE	DESCRIPTION
System.String	

OrderedList(String[])

Creates an ordered (numbered) list from an array of items

Declaration

```
public static string OrderedList(string[] items)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String[]	items	The items of the list

Returns

TYPE	DESCRIPTION
System.String	

Strike(String)

Strikes through the text

Declaration

```
public static string Strike(string text)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	text	The text

Returns

TYPE	DESCRIPTION
System.String	

Strong(String)

Emboldens the text

Declaration

```
public static string Strong(string text)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	text	The text to be emboldened

Returns

TYPE	DESCRIPTION
System.String	

UnorderedList(String[])

Creates an unordered (bulleted) list from an array of items

Declaration

```
public static string UnorderedList(string[] items)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String[]	items	The items of the list

Returns

TYPE	DESCRIPTION
System.String	

# Enum Markdown.HeaderLevel

Namespace: [AeLa.EasyFeedback.Utility](#)

Assembly: cs.temp.dll.dll

## Syntax

```
public enum HeaderLevel
```

## Fields

NAME	DESCRIPTION
H1	
H2	
H3	
H4	
H5	
H6	

# Class ScreenshotUtil

Inheritance

System.Object

ScreenshotUtil

Namespace: [AeLa.EasyFeedback.Utility](#)

Assembly: cs.temp.dll.dll

Syntax

```
public static class ScreenshotUtil
```

## Methods

CaptureScreenshot(ScreenshotMode, Boolean, Action<Byte[]>, Action<String>)

Declaration

```
public static IEnumerable CaptureScreenshot(ScreenshotMode mode, bool resizeLargeScreenshots, Action<byte[]> onCapturedCallback, Action<string> on errorCallback)
```

Parameters

TYPE	NAME	DESCRIPTION
<a href="#">ScreenshotMode</a>	mode	
System.Boolean	resizeLargeScreenshots	
Action< <a href="#">Byte</a> []>	onCapturedCallback	
Action<System.String>	on errorCallback	

Returns

TYPE	DESCRIPTION
IEnumerable	

# Class SetSelectedOnOpen

Inheritance

System.Object

SetSelectedOnOpen

Namespace: [AeLa.EasyFeedback.Utility](#)

Assembly: cs.temp.dll.dll

Syntax

```
public class SetSelectedOnOpen : MonoBehaviour
```

# Class SetVersionText

Inheritance

System.Object

SetVersionText

Namespace: [AeLa.EasyFeedback.Utility](#)

Assembly: cs.temp.dll.dll

Syntax

```
public class SetVersionText : MonoBehaviour
```

## Fields

### Prefix

Declaration

```
public string Prefix
```

Field Value

TYPE	DESCRIPTION
System.String	

### Suffix

Declaration

```
public string Suffix
```

Field Value

TYPE	DESCRIPTION
System.String	

### VersionNumber

Declaration

```
public string VersionNumber
```

Field Value

TYPE	DESCRIPTION
System.String	

# Class ShowHideMouse

Shows or hides the mouse when the feedback form is opened or closed. Remove this component from your form if you do not want the mouse to be automatically managed.

Inheritance

System.Object

ShowHideMouse

Namespace: [AeLa.EasyFeedback.Utility](#)

Assembly: cs.temp.dll.dll

Syntax

```
public class ShowHideMouse : MonoBehaviour
```