# Contributing to UI for Universal Windows Platform

## Before You Start

Anyone wishing to contribute to the UI for Universal Windows Platform project MUST read & sign the [UI For Universal Windows Platform Contribution License Agreement](https://docs.google.com/forms/d/e/1FAIpQLScXkQkYKKZZKXV9aTkhvCT3Q1cACosyhKUILPdOl7Ly9PIL\_g/viewform). The UI for UWP team cannot accept pull requests from users who have not signed the CLA first.

This project has adopted the code of conduct defined by the Contributor Covenant to clarify expected behavior in our community.

For more information see the [.NET Foundation Code of Conduct](https://dotnetfoundation.org/code-of-conduct).

## Introduction

These guidelines are here to facilitate your contribution and streamline the process of getting changes merged into this project and released. Any contributions you can make will help tremendously, even if only in the form of an issue report.

Following these guidelines will help to streamline the pull request, change submission process and serve only one purpose - simplicity.

## Submit a Pull Request

See [Github's documentation for pull requests](https://help.github.com/articles/using-pull-requests).

Pull requests are the preferred way to contribute to UI for UWP. Any time you can send us a pull request with the changes that you want, we will have an easier time seeing what you are trying to do. But a pull request itself is not usually sufficient. Each pull request should bring some context and purpose with it, and it should be done against specific branch.

### A good pull request

Every contribution has to come with:

\* Before starting coding, you should open an issue on the [UI for UWP Issues page](https://github.com/telerik/UI-For-UWP/issues) and start discussing with the community to see if your idea/feature is interesting enough.

\* A good documentation should be provided for every added API - do not expect that your API is so well designed that it needs no documentation. The documentation has a separate repository that could be found [here](https://github.com/telerik/uwp-docs). Once validated your documentation will be visible [here](http://docs.telerik.com/devtools/universal-windows-platform/introduction-uwp)

\* Adding a sample application in the [SDKExamples app](https://github.com/telerik/UI-For-UWP/tree/master/SDKExamples.UWP) that demonstrates new functionality is a sign for good pull request as well. The sample should be applicable and to demonstrate a real case scenarios.

\* Test your code at least with SDK 10586, SDK 14393 and SDK 15063.

\* PR has to target development branch. However, the work you are doing for your pull request should not be done in the development branch of your forked repository. Create a topic branch for your work - always create a branch for your work from the "development" branch. This allows you to isolate the work you are doing from other changes that may be happening.

\* It is very important to provide a meaningful description with your pull requests that alter any code. A good format for these descriptions will include:

\* Why: The problem you are facing (in as much detail as is necessary to describe the problem to someone who doesn't know anything about the system you're building)

\* What: A summary of the proposed solution

\* How: A description of how this solution solves the problem, in more detail than item #2

\* Any additional discussion on possible problems this might introduce, questions that you have related to the changes, etc.

\* (optional) When you've completed your work on a topic branch, you may squash your work down into fewer commits to make the merge process easier. For information on squashing via an interactive rebase, see [the rebase documentation on GitHub](https://help.github.com/articles/interactive-rebase)

PR has to be validated by at least two core members before being merged.

### Quality insurance for pull requests

We encourage developers to comply the following guidances when submitting pull requests for controls:

\* If new control is added it must be usable and efficient with keyboard only - that enable users to accomplish basic app scenarios by using only the keyboard.

\* Tab order must be logical and in accordance to set tab indexes - we recommend the use of arrow keys as keyboard shortcuts for navigating among child elements, in cases where the child elements have a special relationship to each other.

\* Focused controls must be visible - an indicator should be visualized when the control gains focus.

\* For UI elements that can be invoked, implement keyboard event handlers for the Spacebar and Enter keys.

\* Do not use custom colors but instead rely on theme colors so high contrasts themes can be used with your control - to be considered accessible, visible text must have a minimum luminosity contrast ratio of 4.5:1 against the background. Exceptions include logos and incidental text, such as text that is part of an inactive UI component. Please, check [this](https://www.w3.org/TR/WCAG20-TECHS/G18.html) for some more detailed information.

\* Add AutomationProperties.Name on all new controls to define what the controls purpose is.

\* Test the screen reader experience of your UI with the Narrator (Launch Narrator [WinKey+Enter], then CTRL+F12). Check if the information is sufficient, meaningful and helps the user navigate and understand your control

You can find more information about accessibility [here](https://blogs.msdn.microsoft.com/winuiautomation/2015/07/14/building-accessible-windows-universal-apps-introduction)

## Reporting Bugs

When reporting a bug we encourage the following guidelines to be followed:

\* Always update to the most recent master release; the bug may already be resolved.

\* Search for similar issues in the [issues list](https://github.com/telerik/UI-For-UWP/issues) for this repo -- it may already be an identified problem.

\* Make sure you can reproduce your problem locally in an isolated project with no external dependencies.

\* If this is a bug or problem that is clear, simple, and is unlikely to require any discussion -- it is OK to open an issue on GitHub with a reproduction of the bug including workflows, screenshots, links to examples. If you'd rather take matters into your own hands, fix the bug yourself - create a pull request, so that we can help and provide feedback.

## General rules and Code Style

All code contributed to this project should adhere to a consistent style, so please keep these in mind before you submit your Pull Requests:

\* DO NOT require that users perform any extensive initialization before they can start programming basic scenarios.

\* DO provide good defaults for all values associated with parameters, options, etc.

\* DO ensure that APIs are intuitive and can be successfully used in basic scenarios without referring to the reference documentation.

\* DO communicate incorrect usage of APIs as soon as possible.

\* DO design an API by writing code samples for the main scenarios. Only then, you define the object model that supports those code samples.

\* DO NOT use regions. DO use partial classes instead.

\* DO declare static dependency properties at the top of their file.

\* DO NOT seal controls.

\* DO use extension methods over static methods where possible.

\* DO NOT return true or false to give sucess status. Throw exceptions if there was a failure.

\* DO use verbs like GET.

\* DO NOT use verbs that are not already used like fetch.

\* DO space indentation, size of 4

\* DO use braces for one-line blocks (if, for, while, do).

\* DO place braces, "else", "catch", and "finally" on new line.

\* DO qualify member access with this.

## Naming conventions

\* We are following the coding guidelines of [.NET Core Foundational libraries](https://github.com/dotnet/corefx/blob/master/Documentation/coding-guidelines/coding-style.md).

## Documentation

When adding a documentation for a new API use:

\* Readable and self-documenting identifier names.

\* Consistent naming and terminology

\* Provide strongly typed APIs.

\* Use verbose identifier names.

The official documentation for UI for UWP is available [here](http://docs.telerik.com/devtools/universal-windows-platform/Introduction-uwp) and here is the git repo for it [Telerik UI for UWP documentaton repo](https://github.com/telerik/uwp-docs) wher you could find a detailed [contribution guide](https://github.com/telerik/uwp-docs#contributing).

## Files and folders

\* Only a single class should be added per filer.

\* If a feature consist of more than one class use folders to group them.

## Asking for Help

The UI for UWP team does \*not\* provide formal support for the product, except to those customers who have purchased a [commercial license for UI for UWP](http://www.telerik.com/universal-windows-platform-ui) or a support-only package from [Telerik.com](https://www.telerik.com). Please do not create support requests for this project in the issues list for this repo, as these will be immediately closed and you'll be directed to post your question on a community forum.