# Contributing

Thanks you for your interest in contributing to Xamarin.Essentials! In this document we'll outline what you need to know about contributing and how to get started.

## Code of Conduct

Please see our [Code of Conduct](CODE\_OF\_CONDUCT.md).

## Prerequisite

You will need to complete a Contribution License Agreement before any pull request can be accepted. Complete the CLA at https://cla.dotnetfoundation.org/.

## Contributing Code - Best Practices

### Enums

\* Always use `Unknown` at index 0 for return types that may have a value that is not known

\* Always use `Default` at index 0 for option types that can use the system default option

\* Follow naming guidelines for tense... `SensorSpeed` not `SensorSpeeds`

\* Assign values (0,1,2,3) for all enums

### Property Names

\* Include units only if one of the platforms includes it in their implementation. For instance HeadingMagneticNorth implies degrees on all platforms, but PressureInHectopascals is needed since platforms don't provide a consistent API for this.

### Units

\* Use the standard units and most well accepted units when possible. For instance Hectopascals are used on UWP/Android and iOS uses Kilopascals so we have chosen Hectopascals.

### Style

\* Prefer using `==` when checking for null instead of `is`

### Exceptions

We currently have different ways of indicating that nothing can be done:

- do nothing

- throw `FeatureNotSupportedException`

- throw `PlatformNotSupportedException`

- throw `FeatureNotEnabledException`

One case where we do nothing is in Android's energy saver API: if we are not Lollipop, then we just fall through:

https://github.com/xamarin/Essentials/blob/1.0.0/Xamarin.Essentials/Battery/Battery.android.cs#L12-L48

One case where we throw `FeatureNotSupportedException` is with the sensors: if there is no sensor X, then we throw.

One case (and the only case so far) where we throw `PlatformNotSupportedException` is in Android's text-to-speech API: if we try and speak, but we couldn't initialize, then we throw.

So far, I was able to determine that we throw `FeatureNotSupportedException` for:

- the sensors on all platforms if we aren't able to access the hardware

- we throw in the start and the stop (this one may be overkill, we can probably first check to see if it is started, and if not then just do nothing)

- the Android external browser if there was no browser installed

- the email API

- Android: if there is no `message/rfc822` intent handler

- iOS: (if the mail VC can't send, or if the `mailto:` doesn't have an app, or if trying to send HTML over the `mailto:` protocol

- UWP: if the `EmailManager` is not available, or if trying to send HTML

- the flashlight API on all platforms if there is no camera flash hardware

- the phone dialler

- Android / iOS: if the OS can't handle the `tel:` protocol

- UWP: the `PhoneCallManager` is missing

- the sms API

- Android: if there is no `smsto:` intent handler

- iOS: (if the message VC can't send

- UWP: if the `ChatMessageManager` is not available

- the vibration API on UWP if the `VibrationDevice` is not available or if no hardware was found

We throw a `PlatformNotSupportedException` for:

- Android when we aren't able to initialize the text-to-speech engine

We throw a `FeatureNotEnabledException` for:

- Geolocation if no providers are found

We do "nothing":

- the Vibration API on iOS and android never actually checks, it just starts it

- the Map API on Android and UWP just starts the URI, assuming that something will be there

- the Geolocation API always assumes that there is a GPS and throws a `FeatureNotEnabledException` if there was no way to get the hardware

- the KeepScreenOn feature just assumes the window flag will be honoured (probably is, but is there an api level/hardware limit?)

- the energy saver API on android pre-Lollipop

## Documentation - mdoc

This project uses [mdoc](http://www.mono-project.com/docs/tools+libraries/tools/monodoc/generating-documentation/) to document types, members, and to add small code snippets and examples. mdoc files are simple xml files and there is an msbuild target you can invoke to help generate the xml placeholders.

Read the [Documenting your code with mdoc wiki page](https://github.com/xamarin/Essentials/wiki/Documenting-your-code-with-mdoc) for more information on this process.

Every pull request which affects public types or members should include corresponding mdoc xml file changes.

### Bug Fixes

If you're looking for something to fix, please browse [open issues](https://github.com/xamarin/Essentials/issues).

Follow the style used by the [.NET Foundation](https://github.com/dotnet/runtime/blob/master/docs/coding-guidelines/coding-style.md), with two primary exceptions:

- We do not use the `private` keyword as it is the default accessibility level in C#.

- We will \*\*not\*\* use `\_` or `s\_` as a prefix for internal or private field names

- We will use `camelCaseFieldName` for naming internal or private fields in both instance and static implementations

Read and follow our [Pull Request template](https://github.com/xamarin/Essentials/blob/master/.github/PULL\_REQUEST\_TEMPLATE.md)

### Proposals

To propose a change or new feature, review the guidance below and then [open an issue using this template](https://github.com/xamarin/Essentials/issues/new).

#### Non-Starter Topics

The following topics should generally not be proposed for discussion as they are non-starters:

\* Large renames of APIs

\* Large non-backward-compatible breaking changes

\* Platform-Specifics which can be accomplished without changing Xamarin.Essentials

\* Avoid clutter posts like "+1" which do not serve to further the conversation

#### Guiding Principles for New Features

Any proposals for new feature work and new APIs should follow the spirit of these principles:

\* APIs should be simple, direct, and generally implemented with static classes and methods whenever practical

\* Usage of interfaces is to be strictly avoided - APIs should be simple and performant

\* Custom UI should be entirely avoided

\* UI code is only allowable in cases where the platform provides an implementation (eg: Browser, Email Composer, Phone Dialer, etc)

\* New features should have native APIs available to allow implementation on a reasonable subset of the supported platforms, especially (iOS, Android, UWP)

\* No new external dependencies should be added to support implementation of new feature work (there can be exceptions but they must be thoroughly considered for the value being added)

#### Proposal States

##### Open

Open proposals are still under discussion. Please leave your concrete, constructive feedback on this proposal. +1s and other clutter posts which do not add to the discussion will be removed.

##### Accepted

Accepted proposals are proposals that both the community and core Xamarin.Essentials agree should be a part of Xamarin.Essentials. These proposals are ready for implementation, but do not yet have a developer actively working on them. These proposals are available for anyone to work on, both community and the core Xamarin.Essentials team.

If you wish to start working on an accepted proposal, please reply to the thread so we can mark you as the implementor and change the title to In Progress. This helps to avoid multiple people working on the same thing. If you decide to work on this proposal publicly, feel free to post a link to the branch as well for folks to follow along.

###### What "Accepted" does mean

\* Any community member is welcome to work on the idea.

\* The core Xamarin.Essentials team \_may\_ consider working on this idea on their own, but has not done so until it is marked "In Progress" with a team member assigned as the implementor.

\* Any pull request implementing the proposal will be welcomed with an API and code review.

###### What "Accepted" does not mean

\* The proposal will ever be implemented, either by a community member or by the core Xamarin.Essentials team.

\* The core Xamarin.Essentials team is committing to implementing a proposal, even if nobody else does. Accepted proposals simply mean that the core Xamarin.Essentials team and the community agree that this proposal should be a part of Xamarin.Essentials.

##### In Progress

Once a developer has begun work on a proposal, either from the core Xamarin.Essentials team or a community member, the proposal is marked as in progress with the implementors name and (possibly) a link to a development branch to follow along with progress.

#### Rejected

Rejected proposals will not be implemented or merged into Xamarin.Essentials. Once a proposal is rejected, the thread will be closed and the conversation is considered completed, pending considerable new information or changes.