# API Review

## The Importance of API

Having a consistent and functional API is critical in a software component.  There is a famous saying: *"API is forever".*  This is true.  Once a class, interface or method is public or protected, it cannot be deleted and is expected to function until the end of time, unless deprecated for-removal in one release and then removed in a later release (which is a very rare process).

## Steps for API Review

Because API is so important, there are a few additional requirements  over and above what is needed for a bug fix. The process is described on the code review process page.

## Config Files, CSS, Properties and More are API

Usually we think of API in terms of classes and methods, however, things like config files and CSS are also API.  Essentially, any set of characters or keywords that are documented to cause a well-specified behavior can be considered to be API.  When in doubt, it is best to ask.

## Behavioral Changes are API

Changes in behavior can break the users of the toolkit. Major changes in behavior such as threading should be treated like API and require the same steps as an API review. It is critical that the Project Leads and Reviewers are aware of how the toolkit behaves and understand how it is used.

While some API changes such as new classes and methods are easy to see, behavioral changes can be fuzzy.  When in doubt, asking for clarification is the best policy here.

## FXML Considerations

When defining API, it's often easy to forget to take into account how the API will be accessed from FXML.  New events and properties are easily consumed by FXML.  In constructors, use of @NamedArg() allows FXML to construct objects, often to be assigned to properties.

When defining an API, if applicable, ensure that the API can be consumed from FXML and provide test cases that show how it is done.