# Viewmodels

The viewmodels are located within the `Settings.UI.Library` project.

## Components

- Each viewmodel takes in the general `settingsRepository`, the `moduleSettingsRepository` if it exists and the delegates for IPC communication.
- The general `settingsRepository` contains the general configurations of all powertoys whereas the `moduleSettingsRepository` is specific to the module. This is to ensure that the configuration details are shared amongst the viewmodels without having to re-open the `settings.json` file.
- Whenever there is a change in the UI, the `OnPropertyChanged` event is invoked and the viewmodel sends a corresponding IPC message to the runner which would perform the designated action such as dispatching the change to the modules or enabling/disabling the powertoy etc.

#### Difference between viewmodels

- The `GeneralViewModel` is different from the rest of the view models with regard to the IPC communication wherein it sends special IPC messages to the runner to check for updates and to restart as admin.
- Each of the powerToy viewmodels have two types of IPC communications, one for the general status of the powerToy and the other for communication powerToy specific change in properties to the runner.

## `SettingsRepository`

- The `SettingsRepository` is a generic singleton which contains the configurations for each viewmodel.
- As it is a generic singleton, there can only be one instance of the settings repository of a particular type. This ensures that all the viewmodels are modifying a common object and a change made in one locations reflects everywhere.
- The singleton implementation is thread-safe. Unit tests have been added for the same.

### Settings viewmodel anomalies

- The reason behind using the `SettingsRepository` is to ensure that the settings process does not try to access the `settings.json` files directly but rather does it through this class which encapsulates all the file operations from the viewmodels.
- However, this could not be expanded to all the viewmodels directly for the following reasons. Some refactoring must be done to unify these cases and to bring them under the same model:
    - The PowerRename viewmodel does not save the settings configurations in the same format as the rest of the powertoys, ie. {name, version, properties}. However, it only stores the properties directly.
    - Some viewmodels expect the runner to create the file instead of creating the file themselves, like in keyboard manager.
    - The colorpicker powertoy creates the `settings.json` within the module. This must be taken care of when encapsulated within the settingsRepository.
- Currently, all modules use the `SettingsRepository` to access the General Settings config.
- However, only Fancyzones, ShortcutGuide and PowerPreview use the `SettingsRepository` to access the module properties.