# Feature Switching Design

The main idea behind the design is to maintain an in-memory store (FeatureSwitchRuleStore) matching features with the switch rules set up for them.

The store must be populated on application start. During the lifetime of the application, the rules for each of the features can be updated. One way of implementing this is by subscribing to a server-specific topic in a message queue. In this case, each time a feature flag setup is changed in the dashboard, a message with the updated information must be created and broadcasted to the topics for all the application servers.

Each feature switch is described by the type of the applied rule (FeatureSwitchRuleType), the specific rule configuration for the feature encapsulated in implementations of IFeatureSwitchRule interface, desired on/off state set when the rule conditions are satisfied (SwitchOn). If the rule conditions are not met, then the feature enabled/disabled state is taken as the opposite of SwitchOn attribute value.

The main entry point is the FeatureStateProvider class. When we need to check if a specific feature is enabled for a current request, we call the IsFeatureEnabled method passing the feature’s Guid to it. FeatureStateProvider gets an instance of IFeatureSwitchRuleHandler interface from FeatureSwitchRuleHandlerFactory and uses it to determine the state of the feature.

# Example

// constructor injection

public HelloWorldController(FeatureStateProvider featureStateProvider)

// specific controller action

public string Index()

{

if (\_featureStateProvider.IsFeatureEnabled(Features.Greeting))

{

...

}

...

}