**Action Executer**

The main function of the action executer is the carrying out of effects of a positively evaluated rule. The action executer is what carries out the actual actions that are associated with a rule.

The action executer object is created by the rules engine each time a rule has been evaluated at true. Many action executers can exist at one time is multiple rules have been triggered. The rules engine passes the context of the current state of the application as well as an array list of the rEffect objects from the rule. Once the object has been constructed, the action executor will loop through all the effects in the array list. In each loop, the “type” of the effect is enumerated and evaluated as the argument in a switch statement. This is more efficient than using many if statements. The individual case that is evaluated would create a new action object and execute the action.

Each rule type has a unique action class that inherits from AsyncTask. AsyncTasks are Android’s native method of threading and allows for the parallelization of actions being executed. Each thread will be made of an action that can be executed. For instance, a separate thread is created to execute a “toast” (Android’s most simple notification to the user). The individual action classes are passed both the context of the action and the parameters needed to execute that action in the form of a string. The constructor of the action will parse the parameters to make sense of one long string before executing the action. Each action type has a unique method for parsing the parameters because different rules require different parameters (some rules do not have any parameters). The “execute” function of the action class contains source code that will carry out the individual task, using both the application context and the parsed parameters.

The specifications of the action executer require it to be very lightweight and adaptable. The possibility of several action executors existing in memory and executing effects at one time is very probable. The action executer must also be able to execute any type of action. It must be universal enough to create an action object of any type needed and pass the necessary application context and parameters.

**Thread Diagram**

Action Executor

Action

Action

Action

*Context, Effects list*

*Context, Parameters list*

*Context, Parameters list*

*Context, Parameters list*

*execute()*

*execute()*

*execute()*

Each action object stemming from the action executer is executed on its own thread.