

# **The ShockValueBot 2 Specification**

Michael Gohde

Initial Draft 2019-07-10

# Table of Contents

- The ShockValueBot 2 Specification.....1
- 1. Introduction.....3
  - 1.1 Purpose.....3
- 2. The Plugin API.....3
  - 2.1 Definition of a ShockValueBot 2 plugin.....3
  - 2.2 Definition of a ShockValueBot 2 plugin class.....3
- 3. Example Plugin.....5

# 1. Introduction

## 1.1 Purpose

This document exists to describe the interface by which ShockValueBot 2 plugins are implemented. In it, there exists a detailed explanation of how plugins are loaded, instantiated, and called throughout the execution of a ShockValueBot instance.

## 2. The Plugin API

### 2.1 Definition of a ShockValueBot 2 plugin

A ShockValueBot 2 plugin consists of a Python 3 module implementing the following:

1. A class conforming to the specification outlined in the following sections
2. A global variable named INST pointing to the class defined in part 1.

### 2.2 Definition of a ShockValueBot 2 plugin class

ShockValueBot plugins are instantiated and treated as Python objects. As such, they must feature a set of common methods and data structures meeting the following specification:

Common Methods	
<code>__init__(self, <b>logger</b>, <b>client</b>, <b>pluginSet</b>)</code>	<p>A plugin class constructor must accept the following arguments:</p> <ol style="list-style-type: none"><li>1. <b>logger</b> – a function reference to be used when logging events locally. The logger function has the following signature:  <code>logger(<b>name</b>, <b>message</b>)</code> Where <b>name</b> is the name of the plugin, and <b>message</b> is the data to log.</li><li>2. <b>client</b> – a discord.py client object corresponding to the currently running client in use by ShockValueBot.</li><li>3. <b>pluginSet</b> – an internally defined instance of the Plugins class. This represents the set of correctly loaded and instantiated plugins currently available to ShockValueBot.</li></ol>
<code>canHandle(self, <b>cmd</b>, <b>fullmessage</b>)</code>	<p>This method is used to determine whether a given plugin is either:</p> <ol style="list-style-type: none"><li>1. Capable of handling a given command passed</li></ol>

	<p>to ShockValueBot</p> <p>or</p> <p>2. Able to provide a response to some pertinent information within a given message.</p> <p>It is expected that the method will return True or False corresponding to whether or not the plugin in question can meet the above criteria given the following arguments:</p> <ol style="list-style-type: none"> <li>1. <b>cmd</b> – a lowercase, whitespace-stripped token representing the first “word” in a given message. Standalone commands are generally of the form <i>!command</i></li> <li>2. <b>fullmessage</b> – The complete, unaltered contents of a received message.</li> </ol>
<p>async handle(self, <b>message</b>, <b>channel</b>, <b>args</b>, <b>cmd</b>)</p>	<p>This method is used to directly respond to a given message assuming that the conditions provided to canHandle() are met and it returns True.</p> <p>The following arguments have been provided to enable a reasonable degree of flexibility in terms of plugin implementation:</p> <ol style="list-style-type: none"> <li>1. <b>message</b> – a discord.py message object corresponding to the message containing the command.</li> <li>2. <b>channel</b> – a discord.py channel object corresponding to the channel in which the message was sent.</li> <li>3. <b>args</b> – The contents of the received message after having the command passed to canHandle() left stripped. Ie. the remainder of the message after the command token.</li> <li>4. <b>cmd</b> – The command token, a lowercase whitespace-stripped token representing the first “word” in a given message.</li> </ol>
<p>getHelp(self)</p>	<p>It is expected that this method will return a string containing help information regarding the correct usage of the plugin. For example, if the plugin</p>

	<p>implements a given command, the string returned should follow the form:</p> <p><i>!command help related to the command or its arguments</i></p>
async update(self)	<p>This optional method allows a plugin to perform regularly scheduled computation or updates. As presently implemented, ShockValueBot will call this method in every plugin in which it is implemented approximately once every two seconds.</p>

Common Data	
<b>NAME</b>	Each ShockValueBot plugin class must contain a field representing its name after initialization.

### 3. Example Plugin

The following is a listing shows a plugin capable of greeting a user when they send a message starting with “!hello”. It does not implement the optional update() method as it is not necessary to do so for this specific use case.

```
import discord
import asyncio

class Hello:
    def __init__(self, logger, client, pluginSet):
        self.NAME="hello"
        logger(self.NAME, "Initializing hello plugin")

    def canHandle(self, cmdStr, fullMessage):
        return cmdStr.lower()=="!hello"

    async def handle(self, message, channel, args, cmd):
        await channel.send('Hello {0.author.mention}'.format(message))
        return True

    def getHelp(self):
        return "!hello greets a user."

INST=Hello
```