MERK Plugin Development Guide

Version .022

Summary

MERK plugins are Python 3 classes that inherit from a base class, named “Plugin”, that is imported from the MERK application. They are loaded from a subdirectory in the main directory where MERK stores settings and scripts, **`.merk/plugins`**, located in the user’s home directory.

A basic plugin looks something like this:

```
from merk import *

PACKAGE = "Dan’s Plugins"

class DumpPlugin(Plugin):
     NAME = "Dump Plugin"
     VERSION = "1.0"
     DESCRIPTION = "Displays all IRC network traffic"

     def line_in(self,data):
          print("<- "+data)

     def line_out(self,data):
          print("-> "+data)
```

Writing MERK plugins

Plugin requirements and optional features

MERK plugins are Python 3 classes that must meet four basic requirements:

- The class must inherit from the base class **`Plugin`**
- The class must have a **`NAME`** attribute string
- The class must have a **`VERSION`** attribute string
- The class must have at least one event method

There are three optional features that you can add to a plugin:

- A class attribute string named **`DESCRIPTION`**
- A string in the “root” of the plugin’s module named **`PACKAGE`**
- A 48x48px PNG[1] image file with the same file name as the module
- A 25x25px PNG image with the same name as the plugin class name with the file extension “.png”

To get access to the first requirement, simply import **`Plugin`** from **`merk`**. You can either explicitly import Plugin:

```
from merk import Plugin
```

1 https://en.wikipedia.org/wiki/Portable_Network_Graphics

or use a “splat” and import it implicitly:

```
from merk import *
```

The second two requirements, the **NAME** and **VERSION** attributes, are used by MERK to display the plugin in the client. These should be class attributes, rather than instance attributes[2]. **NAME** should be set to the name of the plugin, a short descriptive string; it must contain at least one character that is not whitespace. **VERSION** should be the version number of the plugin; if no version number is required, this attribute can be set to a blank string, but it *must* exist. A third attribute, **DESCRIPTION**, is optional; this should be a short string that describes what the plugin does.

The last requirement is for the plugin to have at least one event method. An event method is a plugin class method that MERK will execute when a specific event occurs.

If you add a string named **PACKAGE** to the plugin module, this string value will be used to display the plugin in MERK’s “Plugins” menu.

If a 48x48 pixel PNG image is in the same directory as the module, and has the same name as the module (with the exception of the file extension), this image will be used as the module’s “icon” in MERK’s “Plugins” menu.

If a 25x25 pixel PNG image is in the same directory as the module, and has the same name as the plugin *class* (with the file extension “.png”), this image will be used as the plugin’s icon in MERK’s “Plugins” menu.

> For example, say we have a plugin. It’s in a module named “bleep.py”, and the **Plugin** class in the module is named **MyPlugin**. If you want to display a module icon, you would name your 48x48px PNG image “bleep.png”. If you want to display a plugin icon, you would name your 25x25px PNG image “MyPlugin.png”. Remember, file names are case-sensitive. Place both of these images in the same directory as **bleep.py**, and you’re done!

For convenience, you can put your plugin in its own directory in **.merk/plugins**, named whatever you wish. Just like with normal Python modules, sub directories need a **__init__.py** file in order to be detected by MERK.

For best practice, each plugin should exist either as a single Python file, or in a single directory, with no subdirectories. This is not enforced in any way, however.

Plugin class and instance features

The Plugin class contains some built-in methods and the **irc** attribute to make interacting with the MERK client, any connected IRC servers, and even other **Plugins** easy.

2 https://www.geeksforgeeks.org/class-instance-attributes-python/

The `irc` attribute

The **`irc`** attribute is part of the **`Plugin`** class, and is the way plugins interact with IRC servers. It is integrated as an instance attribute[3] of the class. This attribute/object is the instance of the Twisted IRC client[4] that MERK is using for communication with the IRC server. Anything you can normally do with the Twisted IRC client, you can use **`irc`** for. Whenever an event method is triggered, the **`irc`** attribute is set to the Twisted instance that is connected to the server event that triggered the method. So, for example, if you wanted to write a method that forwards all private messages to another user with the nickname “OtherNick”, you could write:

```
def private(self,user,message):
     self.irc.msg("OtherNick",user+": "+message)
```

If the **`irc`** attribute is unavailable (due to a disconnection, error, or other reason), the attribute’s value is set to **`None`**. For help on how to use the **`irc`** attribute, take a look at the documentation for the Twisted IRC Client[5].

One additional feature MERK built into the **`irc`** attribute is a new attribute, **`client_id`**. The **`client_id`** attribute is a string that is unique for each connection. This way, plugins can tell the difference between **`irc`** attributes that may be connected to the same IRC server.

3 https://docs.python.org/3/tutorial/classes.html#class-and-instance-variables
4 twisted.words.protocols.irc.IRCClient
5 https://twistedmatrix.com/documents/current/api/twisted.words.protocols.irc.IRCClient.html

Event methods

MERK executes the event method of every plugin that contains that event method when a specific event occurs (like the reception of a public message, a private message, or a notice message). In short, MERK plugins are event-driven[6].

load

Arguments None

Description Triggered every time MERK loads the plugin.

unload

Arguments None

Description Triggered every time MERK unloads the plugin (that is, when MERK closes).

line_in

Arguments data (string)

Description Triggered every time MERK receives data from an IRC server. data contains the data sent to the client.

line_out

Arguments data (string)

Description Triggered every time MERK sends data to an IRC server. data contains the data sent to the server.

6 https://en.wikipedia.org/wiki/Event-driven_programming