**SELENIUM-PYTHON FOR TEST AUTOMATION**

This documentation is specific to using **Selenium with the Python Language**. The following documentation applies to using the Windows Operating System (Windows 10). Some of the implementation details may work with MAC OS but has only been tested on the Windows 10 OS as per this documentation. *MAC OS driver implementation* can be found at the end of this document.

The purpose of using Selenium via the Python Language is to automate the web browser to simulate user interactions through the web browser for testing purposes.

**To install Selenium for Python**, you must make sure that Python is first installed on your computer. Supported versions of Python for Selenium are Python 2.7 and 3.4+. You can check that Python is installed by entering this code into the command line (terminal window): python –version.

**SELENIUM INSTALLATION**

To use selenium (for Python) you must install the Selenium Python bindings (i.e.: the API used to access selenium webdrivers for use with specific browsers –i.e.: Chromedriver, Firefox gecko driver).

There are two ways to install Selenium Python Bindings

1. You can download the source file from PyPI (<https://pypi.org/project/selenium/#files>) specific for your platform. Once it has been downloaded, decompress the file (extract it) and run the following command in your terminal window: **python setup.py install**
2. You can use ‘pip’ to install (or upgrade) the Python bindings with the following command:

**pip install -U selenium (Note that you must have ‘pip’ installed to use this command**. Pip comes with the Python versions mentioned above**)**

**Selenium Drivers**

Selenium web drivers are used to control the web browser. The two web drivers that will be described for use here are Chromedriver for the Google Chrome browser and Gecko driver for the Firefox browser.

**Prerequisites**

Ensure that Chrome and Firefox are already installed on your computer in their default locations (as per the installation).

**Chromedriver**

Download the chromedriver executable from this link below (**Note** it is important that you know what **version** of Google Chrome you have so that you can download the correct release of Chromedriver):

<http://chromedriver.chromium.org/downloads>

Once the chromedriver executable file is downloaded it must be placed where the webdriver can locate it. This can be accomplished in two ways:

1. Provide the location of Chromedriver to your ‘Path Environment Variable’ on your computer.

Windows OS: Go to ‘Advanced System Settings’ on your computer (i.e.: right-click on ‘This PC’, click on ‘properties’, and then Advanced System Settings). Next, click on the ‘Environment Variables’ button, then in ‘System Variables’ section, scroll down to ‘Path,’ select it, **click ‘Edit’** **and then click ‘New’** and paste or type it in the location that you placed the chromedriver executable file (Note you are just providing the path to the folder you placed it in---do not type in the name of the chromedriver.exe file itself).

**IMPORTANT**: When using **Python**, the **chrome executable file** must be placed in the **Python directory (where the name of your browser folder is located)**. After you place the executable in the Python directory, add this location to your system environmental path (see below example):

**Place the location of your chrome.exe file within the Python directory in the system environment path on your computer (see below):**

C:\Users\Owner\AppData\Local\Programs\Python\Python35-32\Lib\site-packages\selenium\webdriver\**chrome**\

(The **chorme.exe** file should be placed inside the chrome folder located in the above path). –**Note**: It is important that you place the backslash character at the end of the path statement above or your code will not work.

To verify that Chromedriver can be located and is working just go to the command prompt in your terminal window and type in **chromedriver** and hit return. You should see the chromedriver server start.

1. Another way that Webdriver can locate chromedriver is to include its path location in your Python script during instantiation: driver = webdriver.Chrome(‘place path to chromedriver here’)

EXAMPLE: driver = webdriver.Chrome(‘C:\Users\Owner\AppData\..\Chrome\’)

Note the **Capital ‘C’** in ‘webdriver.Chrome().

**Precaution**: Google Chrome automatically updates itself which can create compatibility issues with chromedriver. If you try to run your automation tests and they no longer work, you may receive this type of error: selenium.common.exceptions.SessionNotCreatedException: session not created: Chrome version must be between 70 and 73.

Google will usually provide updated releases of chromedriver during the same time frame that the Chrome Browser updates itself. To resolve the compatibility issue, you can do one of two things:

**Quick-fix:** download the latest version of Google Chromedriver (here: <http://chromedriver.chromium.org/downloads>) and place the executable file in the same directory folder that you save and run your Python scripts from.

**Alternative fix:** Replace the current chromedriver with the newest release by deleting the current one from the original folder location and putting the new one in that location (i.e.: it is the path you indicated in your ‘system environment variables’).

**Launching Firefox via Gecko Driver:**

You must download Gecko Driver and, just like above with chromedriver, and set the location of its path so that the webdriver can find it.

**Note**: You must know what version (32-bit vs. 64-bit) of Firefox you have (this can be looked up in the browser’s ‘help’--->’About Firefox’ menu ( ‘About Mozilla Firefox’ pop-up window will display the 64-bit or 32-bit version info).

Download the most recent version of Gecko Driver at the below GitHub link (Here, you must download the correct version based on the version of Firefox you have ---32 bit or 64 bit):

<https://github.com/mozilla/geckodriver/releases>

The version discussed henceforth will be for the most current release for Firefox 64-bit: (**geckodriver-v0.24.0-win64.zip**)

There are two ways to make Geckodriver accessible to webdriver:

1. Provide the location of Geckodriver to your system ‘Path Environment Variable’ on your computer. **Windows 10 OS**: Go to ‘Advanced System Settings’ on your computer (i.e.: right-click on ‘This PC’, click on ‘properties’, and then Advanced System Settings). Next, click on the ‘Environment Variables’ button, then in ‘System Variables’ section, scroll down to ‘Path,’ select it, click ‘Edit’ and then click ‘New’ and paste or type in the location that you placed the Geckodriver executable file (Note you are just providing the path to the folder you placed it in---you do not need to type in the name of the ‘geckodriver.exe’ file itself).

**EXAMPLE:** C:\Users\Owner\firefox\GeckoDriver\

**IMPORTANT**: When using **Python**, the **gecko executable file** must be placed in the **Python directory (where the name of your browser folder is located)**. After you place the executable in the Python directory, add this location to your system environment path (see below example):

**Place the location of your geckodriver.exe file within the Python directory in the system environment path on your computer:**

C:\Users\Owner\AppData\Local\Programs\Python\Python35-32\Lib\site-packages\selenium\webdriver\**firefox**\

(The **geckodriver.exe** file should be placed inside the Firefox folder located in the above path)

To verify that the gecko driver is now working place the following code in a text file and run it at the command prompt via python or run python at the command prompt (i.e.: once your terminal window is open be sure to type ‘python’ at the command prompt so that you are in the python code editor) and paste the following code there (A browser window should open up the google search page in your Firefox browser if gecko driver is working correctly):

from selenium import webdriver

#--skipped line--

driver= webdriver.Firefox()

driver.implicitly\_wait(10)

driver.get('http://google.com')

**MAC OS DRIVER IMPLEMENTATION**

**HOW TO MAKE GECKO DRIVER OR CHROME DRIVER ACCESSIBLE ON A MAC:**

OPEN a terminal window on your **MAC** and follow the below steps:

1. Look up your current path by typing in ‘echo $PATH’ at the command prompt (**without** the quotes)
2. To modify the path to include the directory that you placed geckodriver.exe or chromedriver.exe in, type the

following command at the prompt (this assumes that your file is still in the downloads folder):

export PATH=$PATH:~/Downloads/

1. Check that the path was created by typing in echo $PATH
2. You should see the ‘Downloads’ folder now at the end of the path
3. To confirm that your driver can be found on the path, TYPE geckodriver or chromedriver (whichever one you

downloaded) at the command prompt in your terminal window.

**Note:** You should be able to see that the server has started, as well as the local port that your driver is listening on. If for some reason there is an issue with the driver being located, place the executable file either in the Python directory or one of these two directories: /usr/local/bin or /usr/bin. \***Remember** to export it to the PATH.

To begin running tests, just press **Ctrl + C** and this will get you back to an active command prompt.