



|  |  |
| --- | --- |
|  | **2017** |
|  |  |

**ROBOT FRAMEWORK FOR WINDOWS**

**INSTALLATION INSTRUCTIONS**

**VERSION 2.5 JANUARY 27, 2017**

This document describes the installation of Robot Framework on the Windows platform. The document includes installation of certain packages that are required to execute the scripts created using Robot Framework and the RIDE IDE (Integrated Development Environment).

# Table of Contents

[Table of Contents 1](#_Toc17059)

[Installation of Robot Framework 3](#_Toc17060)

[Preconditions 3](#_Toc17061)

[Python 3](#_Toc17062)

[wxPython 4](#_Toc17063)

[Robot Framework 5](#_Toc17064)

[Installation of PIP 5](#_Toc17065)

[RIDE 7](#_Toc17066)

[Selenium2Library 7](#_Toc17067)

[AutoItLibrary 7](#_Toc17068)

[Web Drivers 9](#_Toc17069)

[Download CSV files automatically in Firefox using Selenium WebDriver: 9](#_Toc17070)

[Start-Up 12](#_Toc17071)

[Other Libraries 13](#_Toc17072)

[Other Keywords added to Selenium2Library 13](#_Toc17073)

[Updated Built In keyword in Selenium2Library 15](#_Toc17074)

# **Installation of Robot Framework**

## Preconditions

### Python

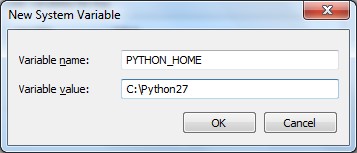
RIDE runs only on the regular Python, not on Jython or IronPython. **Python 2.6** is the minimum version. These instructions refer to Python 2.7. Notice that similarly as Robot Framework, RIDE does not yet support Python 3.

Download and install Python 2.7.\* from [http://python.org.](http://python.org/) Go to the Python download page and choose a Windows x86 MSI Installer. Choose 32 bit Python (“x86”), not 64 bit, to be compatible with the AutoItLibrary. Note that 32 bit Python will run properly on a 64 bit system.

A quick reference link is find S.No1(In excel sheet): <https://www.python.org/downloads/> Choose defaults when you run python-2.7.6.msi.

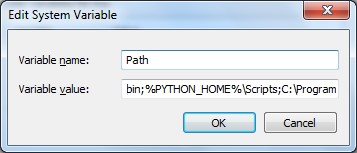
#### Setting Environment Variables

1. Navigate to *Computer* ‣ *Properties* ‣ *Advanced system settings*‣ *Environment Variables*
2. Click on “new” to add an environment System variable **PYTHON\_HOME** and set the value as “**C:\Python27**” (refer to below screenshot)



1. Add “**%PYTHON\_HOME%;**” and “**%PYTHON\_HOME%\Scripts;**” to your system

‘**Path’**. Ensure that there is a “**;**” at the beginning and end of the new entries in the path (unless it is the first or last entry). Refer to the screenshot below.



To verify if Python is correctly installed on your machine, run the below command in a command (cmd) window.

Python --version

If you get the response as shown in the snapshot below, it means Python is installed in your system. (This install was for Python 2.7.5.)



### wxPython

RIDE's GUI is implemented using the [wxPython](http://wxpython.org/) toolkit. Only **wxPython** with Unicode support is officially supported. (Recent versions of wxPython have Unicode support.) The ANSI version of wxPython is not supported. The recommended wxPython version is 2.8.12.1.

|  |
| --- |
| wxPython2.8-win32-unicode-2.8.12.1-py27.exe |

Download an appropriate installer (such as ) from the SourceForge project page at, A quick reference link find S.No2(In excel sheet):https://sourceforge.net/projects/wxpython/files/wxPython/2.8.12.1/wxPython2.8-win32-unicode-2.8.12.1-py27.exe/downloadand run the installer using default choices.

### Robot Framework

There are multiple ways to install Robot Framework. You can install Robot Framework from source, via Windows installers or through Python package managers. The following installation steps are via the Python package manager PIP.

## Installation of PIP

**pip** requires [setuptools,](https://pypi.python.org/pypi/setuptools) so setuptools has to be installed first, before pip can be installed.

To install setuptools from scratch:

1. Download ez\_setup.py. Please follow the download instructions below:
   1. There is a location of ez\_setup.py at , A quick reference link find S.No3(In excel sheet): https://bitbucket.org/pypa/setuptools/downloads/ez\_setup.py [.](https://bitbucket.org/pypa/setuptools/raw/bootstrap/ez_setup.py)
   2. Copy the code and save it in a file called ez\_setup.py. We recommend putting the file in a directory such as C:\Python27\Scripts.
   3. Navigate to the directory where you have placed ez\_setup.py.
2. Then run the following (which may require administrator access) from the folder that contains ez\_setup.py using a command window:

python ez\_setup.py

1. Verify if the ‘setuptools’ are correctly installed. Once the above py file is run, at the bottom of the command prompt, look for the below message. The version file may be more recent.

Output:



1. Download get-pip.py. Please follow the download instructions below:
   1. Navigate to, A quick reference link find S.No4(In excel sheet): https://bootstrap.pypa.io/
   2. Right-click on [get-pip.py](https://bitbucket.org/pypa/setuptools/raw/bootstrap/ez_setup.py) and click on ‘Save link as’
   3. Download the file. We recommend putting the file in a directory such as C:\Python27\Scripts.
2. Then run the following (which may require administrator access) from the folder that contains get-pip.py using a command window, to install the latest version of pip:

python get-pip.py

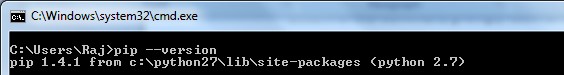
1. To verify if PIP is correctly installed in your machine, run the below command. You should see the output as shown below, although perhaps with a newer version number.

pip

--

version

Output:



1. Then get the latest install of [setuptools,](https://pypi.python.org/pypi/setuptools) by running the below command

pip install --upgrade setuptools

1. Now, install Robot Framework by running the below command

pip install robotframework

If you experience a problem with pip use “easy\_install robotframework”.

1. After a successful installation of Robot Framework you should be able to execute the following command on the command prompt. Your version number may be newer.

pybot

--

version

Output:



## RIDE

RIDE can be installed with [pip](http://www.pip-installer.org/) using the below command:

pip install robotframework-ride

You can also create a desktop shortcut for Ride. I got the desktop icon by again going through the steps for installing Ride, this time using the robotframework-ride-

1.2.1.win32.exe file from the robotframework-ride project in code.google.com. I selected the option to get the icon. The install returned an error. But the Ride icon appeared on the desktop. I then modified the Target for the Ride icon (in the properties) by replacing “C:\Users\<my user name>\” by “C:\Python27\”. This approach is a bit round-about, but it does the job of getting the RIDE icon on the dashboard.

## Selenium2Library

Selenium2Library can be installed using pip. The advantage of installing it via pip is that it will automatically install its dependencies.

You can install Selenium2Library using pip, with the following command

pip install robotframework-selenium2library

## AutoItLibrary

Tests can use the native Windows GUI testing library AutoItLibrary to manipulate the mouse. These tests will not be available for Mac or Linux.

AutoItLibrary documentation mentions a dependency on Python for Windows Extension, pywin32. That project is in sourceforge. Please follow the below instructions to download and install ‘pywin32’.

1. Go to[,](http://sourceforge.net/projects/pywin32/files/pywin32/) A quick reference link find S.No5(In excel sheet): <http://sourceforge.net/projects/pywin32/files/pywin32/>
2. Choose the folder for Build 218 or later.
3. Click on ‘README.txt’ to understand your system configuration. Note that since we installed 32 bit Python, the 32 bit version of Python for Windows extension has to be used.
4. Click on the link in the web page to download the appropriate *pywin32* when you have *32-bit Python27* installed:



1. Open a command window using the “Run as Administrator” option. Browse to the download location of the file, and run the .exe file to install pywin32. You could run the file by typing .\pywin32-218.win32-py2.7.exe.

a. Reference: [http://stackoverflow.com/questions/11106281/error-installing-andrunning-pywin32-2-7](http://stackoverflow.com/questions/11106281/error-installing-and-running-pywin32-2-7)

After installing the dependency libraries, download and install ‘AutoItLibrary’ by following the below instructions

1. Go to, A quick reference link find S.No6(In excel sheet) <https://code.google.com/p/robotframework-autoitlibrary/>
2. Click on ‘Downloads’ and follow the links to download the [‘AutoItLibrary-1.1.zip’](https://code.google.com/p/robotframework-autoitlibrary/downloads/detail?name=AutoItLibrary-1.1.zip&can=2&q=)
3. To install AutoItLibrary, unzip the downloaded source release file into a directory on your PC, open a command window in that directory, and type:

python setup.py install

**NOTE**

:

In Windows Vista and Windows 7 you must do thi

s installation as Administrator.

The installation creates a folder, either C:\RobotFramework\Extensions\AutoItLibrary or C:\Python27\Lib\site-packages\AutoItLibrary, on your PC and puts various files into this directory folder.

## Web Drivers

The Robot Framework scripts can be executed on a Firefox browser by default without specifically adding a web driver. However, executing scripts on ***Internet Explorer*** and ***Google Chrome*** requires web drivers to be downloaded and placed in the Path environment variable.

 Download IE driver server. Use the 32 bit version (Win32) as it is more stable. Get the server from [http://selenium-release.storage.googleapis.com/index.html.](http://selenium-release.storage.googleapis.com/index.html) Use the directory with the latest version number. The file’s name will be

“IEDriverServer\_Win32\_2.40.0.zip” or a later version. Extract the file to get “IEDriverServer.exe”.

 Download the Chrome driver. I suggest using the 32 bit version in case there are compatibility issues. You can get the server from the directory [http://chromedriver.storage.googleapis.com/index.html?path=2.9/.](http://chromedriver.storage.googleapis.com/index.html?path=2.9/) The zip file name is “chromedriver\_win32.zip”. Extract the file to get “chromedriver.exe”.

 Place the two drivers in the “C:\Python27” folder or in some other folder in your Path.

I recommend that you put the two drivers into a new folder called “C:\DriverServers” and then add “C:\DriverServers” to the Path environmental variable.

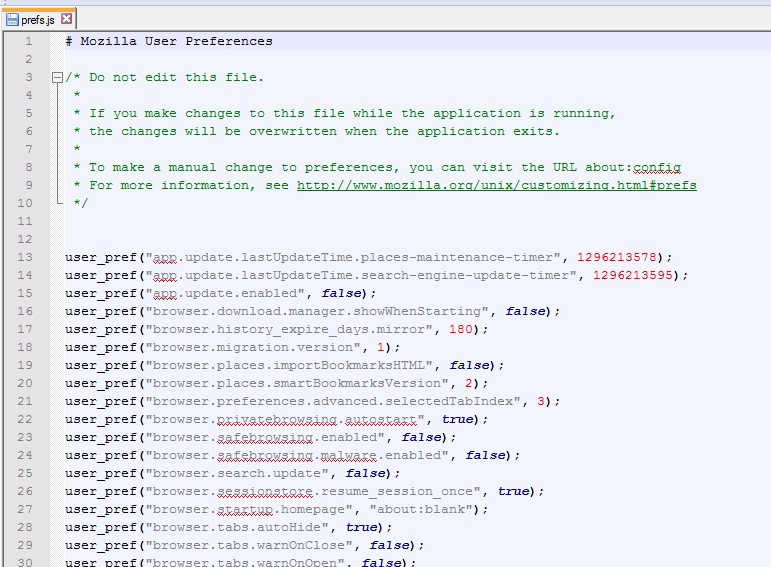
 If you have not done so, install Chrome and Firefox. (Internet Explorer comes with Windows.)

## Download CSV files automatically in Firefox using Selenium WebDriver:

As Selenium itself doesn't interact with system-level dialogs, in order to download CSV as part of the browser automation process, it requires the help from either additional frameworks or an approach that handles downloading automatically.

Firefox's download manager preferences are controlled by some properties defined in pref.js file, which can be set programmatically while instantiating FirefoxDriver using Selenium WebDriver.

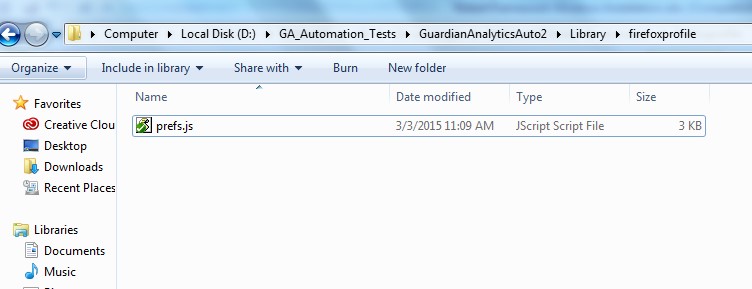
browser.helperApps.neverAsk.saveToDisk stores a comma-separated list of MIME types to save to disk without asking what to use to open the file. Default Pref.js file



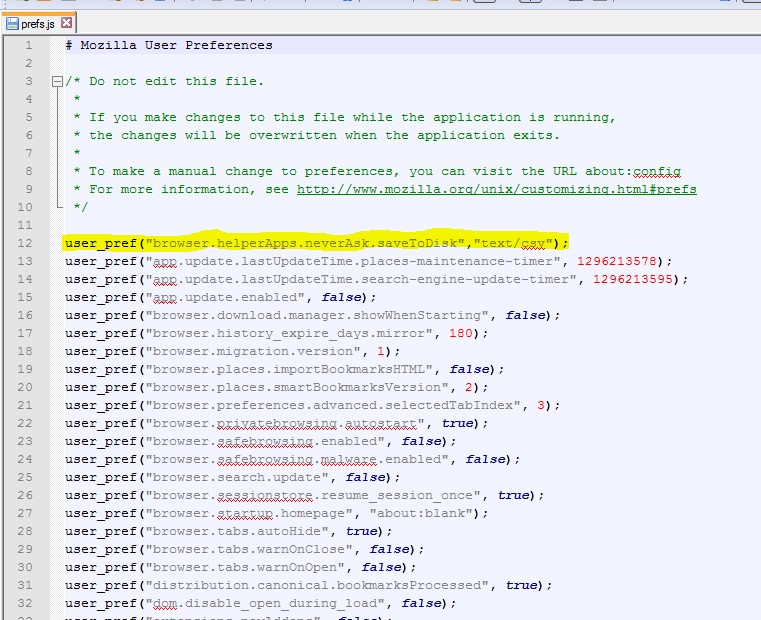
Default Directory: C:\Python27\Lib\site-packages\Selenium2Library\resources\firefoxprofile

user\_pref("browser.helperApps.neverAsk.saveToDisk","text/csv");

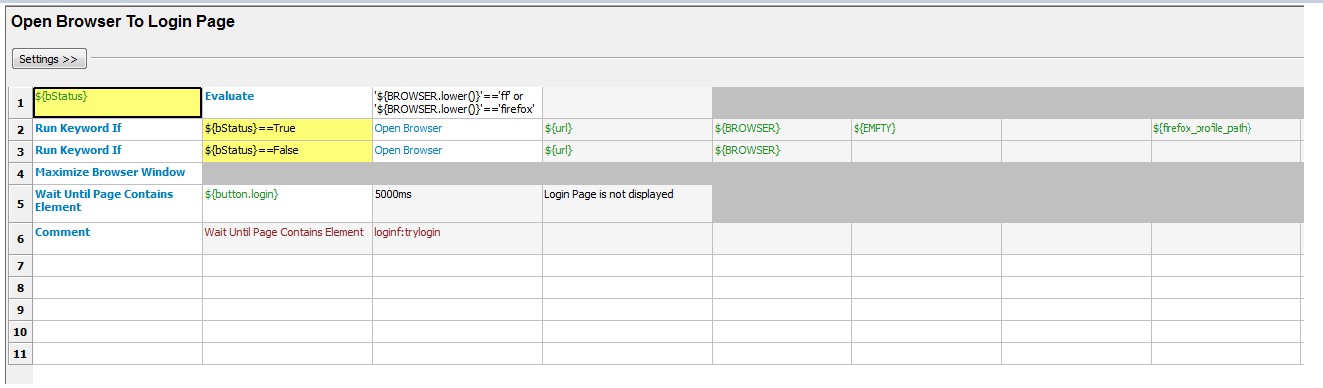
We have added the above command in pref.js file and placed this file in Library\firfoxprofile folder within project directory.



Updated Pref.js file:



Now test is lunching the browser using “Selenium2Library.Open Browser” keyword by passing Firefox profile directory as ${firefox\_profile\_path}.



${firefox\_profile\_path} = ${EXECDIR}\\Library\\firefoxprofile

**Note:** ${EXECDIR} = project directory

## Start-Up

|  |
| --- |
| ride.py |

After a successful installation RIDE can be started from the command line by running . If a small window appears, simply resize it. If you installed the RIDE icon on the desktop, click (or double click) on it to start RIDE.

## Other Libraries

There are several libraries that can be useful for test development:

* xlrd: To manipulate Excel files: o pip install xlrd
* pytz: To work with time zones o We had problems with pip, so you may have to use “easy\_install pytz”
* python-dateutil: to get extensions to the standard Python datetime module.
  + pip install python-dateutil
* Other library files may be needed for your installation.
  + pip install pyPdf pip install pyPdf2

 for tests that manipulate PDF files

## Other Keywords added to Selenium2Library

There is a file called “\_element.py” which may require additional keywords to be added. For example, if the test suite uses the keyword “Mouse Scroll” then the “Mouse Scroll” keyword listed below should be added to the “\_element.py” file. The simplest approach is to copy each of these keywords into \_element.py, since you may not know which keywords are used.

“\_element.py” is typically located in

“C:\Python27\Lib\site-packages\Selenium2Library\keywords\\_element.py”.

The lines to add are:

def mouse\_click(self):

"""Simulates moving mouse away from the element specified by `locator`.

Key attributes for arbitrary elements are `id` and `name`. See `introduction` for details about locating elements.

"""

ActionChains(self.\_current\_browser()).move\_by\_offset(1042, 840).click()

def mouse\_scroll(self, locator):

"""Simulates releasing the left mouse button on the element specified by `locator`.

Key attributes for arbitrary elements are `id` and `name`. See `introduction` for details about locating elements.

"""

self.\_info("Simulating Mouse Up on element '%s'" % locator) element = self.\_element\_find(locator, True, False) if element is None: raise AssertionError("ERROR: Element %s not found." % (locator)) size = element.size offsetx = (size['width'] / 2) + 1 offsety = (size['height'] / 2) + 1

# the following is one long line indented the same as the above lines

ActionChains(self.\_current\_browser()).move\_to\_element(element).click\_and\_hold(elem

ent).move\_by\_offset(offsetx,offsety).release(element).perform()

# the above is one long line indented the same as the other lines

def mouse\_release(self, locator):

"""Simulates releasing the left mouse button on the element specified by `locator`.

Key attributes for arbitrary elements are `id` and `name`. See `introduction` for details about locating elements.

"""

#self.\_info("Simulating Mouse Up on element '%s'" % locator) element = self.\_element\_find(locator, True, False) if element is None:

raise AssertionError("ERROR: Element %s not found." % (locator)) ActionChains(self.\_current\_browser()).release(element)

def key\_hold(self,key):

"""Simulates user pressing control key """ if key.startswith('\\') and len(key) > 1:

key = self.\_map\_ascii\_key\_code\_to\_key(int(key[1:])) #if len(key) > 1:

# raise ValueError("Key value '%s' is invalid.", key) ActionChains(self.\_current\_browser()).key\_down(key).perform()

## Updated Built In keyword in Selenium2Library

Most of the test scripts fail due to window resizing (flickering) issues. We have identified the reason for failure is “while capturing screen shots window is resizing on IE browser” , So we have updated the “capture\_page\_screenshot” built in Selenium2Library keyword based browser name variable

“${BROWSER}” and screenshot file name. This keyword will not capture the screen shots in IE browser case, if you want capture screenshot on IE browser we need to send “file name” as an argument for this keyword.

Updated the keyword “capture\_page\_screenshot” to not to capture screen shots in IE Browser at location C:\Python27\Lib\site-packages\Selenium2Library\keywords\\_screenshot.py file.

**Note:** These updates will not effect on the test execution in Firefox browser.

Import Statement:

from robot.libraries.BuiltIn import BuiltIn

Updated Keyword:

def capture\_page\_screenshot(self, filename=None):

"""Takes a screenshot of the current page and embeds it into the log.

`filename` argument specifies the name of the file to write the screenshot into. If no `filename` is given, the screenshot is saved into file

`selenium-screenshot-<counter>.png` under the directory where

the Robot Framework log file is written into. The `filename` is also considered relative to the same directory, if it is not given in absolute format.

`css` can be used to modify how the screenshot is taken. By default the bakground color is changed to avoid possible problems with background leaking when the page layout is somehow broken.

"""

fpathStatus = (filename!=None) print "fpathStatus=" + str(fpathStatus) print "Browser= "+str(BuiltIn().get\_variable\_value("${BROWSER1}")) browserStatus = (str(BuiltIn().get\_variable\_value("${BROWSER1}"))=="ie") print "browserStatus="+str(browserStatus) path, link = self.\_get\_screenshot\_paths(filename)

if hasattr(self.\_current\_browser(), 'get\_screenshot\_as\_file'): print "\_current\_browser().get\_screenshot\_as\_file updated" if fpathStatus==True and browserStatus==True:

self.\_current\_browser().get\_screenshot\_as\_file(path) elif fpathStatus==False and browserStatus==True:

print "no file name" elif browserStatus==False: self.\_current\_browser().get\_screenshot\_as\_file(path)

else:

print " conditions not matched "

else:

print "\_current\_browser().save\_screenshot(path) updated" if fpathStatus==True and browserStatus==True: self.\_current\_browser().save\_screenshot(path) elif fpathStatus==False and browserStatus==True:

print "no file name" elif browserStatus==False:

self.\_current\_browser().save\_screenshot(path)

else:

print "conditions not matched "

#self.\_current\_browser().save\_screenshot(path) # Commented by me

# Image is shown on its own row and thus prev row is closed on purpose self.\_html('</td></tr><tr><td colspan="3"><a href="%s">'

'<img src="%s" width="800px"></a>' % (link, link))

