Using PLAXIS Remote scripting with the Python wrapper

29 APRIL 2015

Latest version of this document available online:
https://www.plaxis.com/support/tips-and-tricks/using-plaxis-remote-scripting-with-the-python-wrapper/

Applications: 2D 2015, 2D 2016, 3D AE

The PLAXIS software provides a HTTP based API (REST HTTP API), for which a special Python wrapper was developed to for an easy to use scripting API. Both PLAXIS Input and PLAXIS Output support this usage of a Remote Scripting server.

Prerequisites

In order to use this

- a PLAXIS VIP licence is required
- a recent version of Python. This can be downloaded from http://python.org/download/ . The Python installer comes with a code editor called IDLE. Many alternatives are available, however in Plaxis documentation we will assume IDLE as the editor of choice.
- At least a very rudimentary knowledge of the Python language is recommended. The Plaxis documentation makes no attempt to teach it. Good resources for this purpose are e.g.:
    - Dive into Python 3, http://www.diveintopython3.net/
    - After hours programming http://www.afterhoursprogramming.com/tutorial/Python/Overview/
    - Think Python: How to Think Like a Computer Scientist, http://greenteapress.com/thinkpython/html/index.html
- The firewall should not block the PLAXIS application from accessing the internet, nor must it block other applications (in particular the python.exe executable) from communicating with the remote scripting server inside the PLAXIS application.

Activate scripting server

In order to start using this Remote Scripting/REST HTTP API, the remote scripting server inside PLAXIS needs to be activated. This can be done by opening the menu item *Expert* > *Configure remote scripting server*. The corresponding window will pop up, see Figure 1:

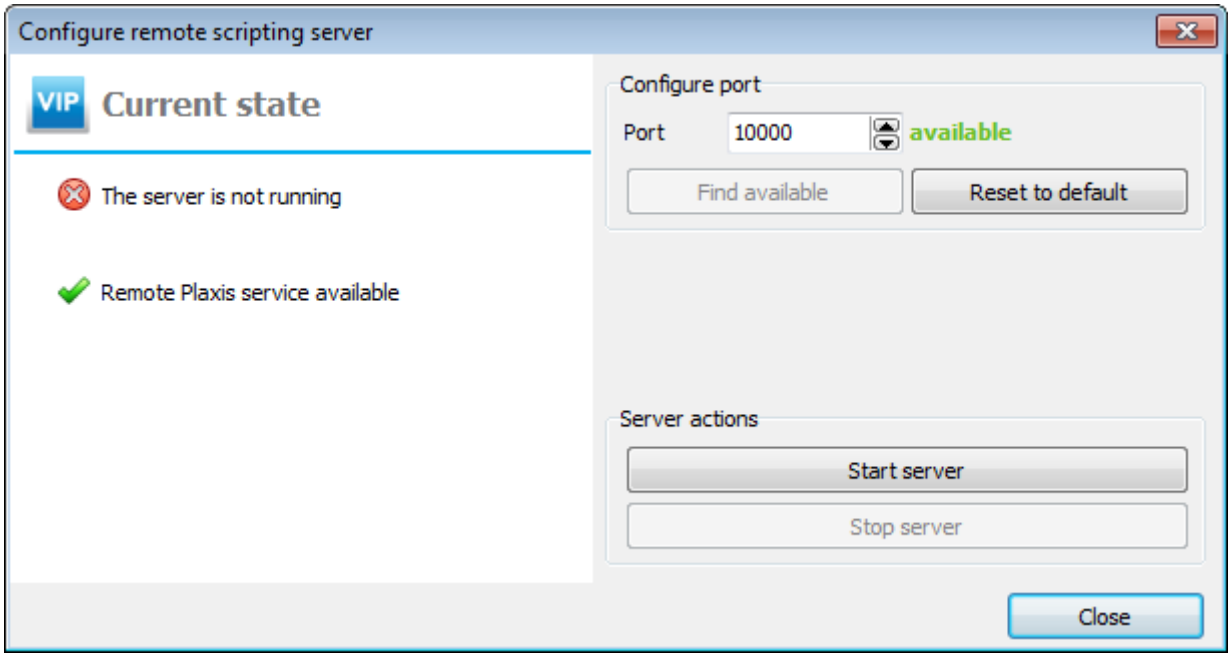

Figure 1. Configure remote scripting server window

In order for the Remote Scripting server to run, you would need:

1. A valid PLAXIS VIP licence;
2. A working HTTP connection to the Plaxis Remote scripting Authorization site (i.e. a working internet connection);
3. The local connection port needs to be configured and the server activated.

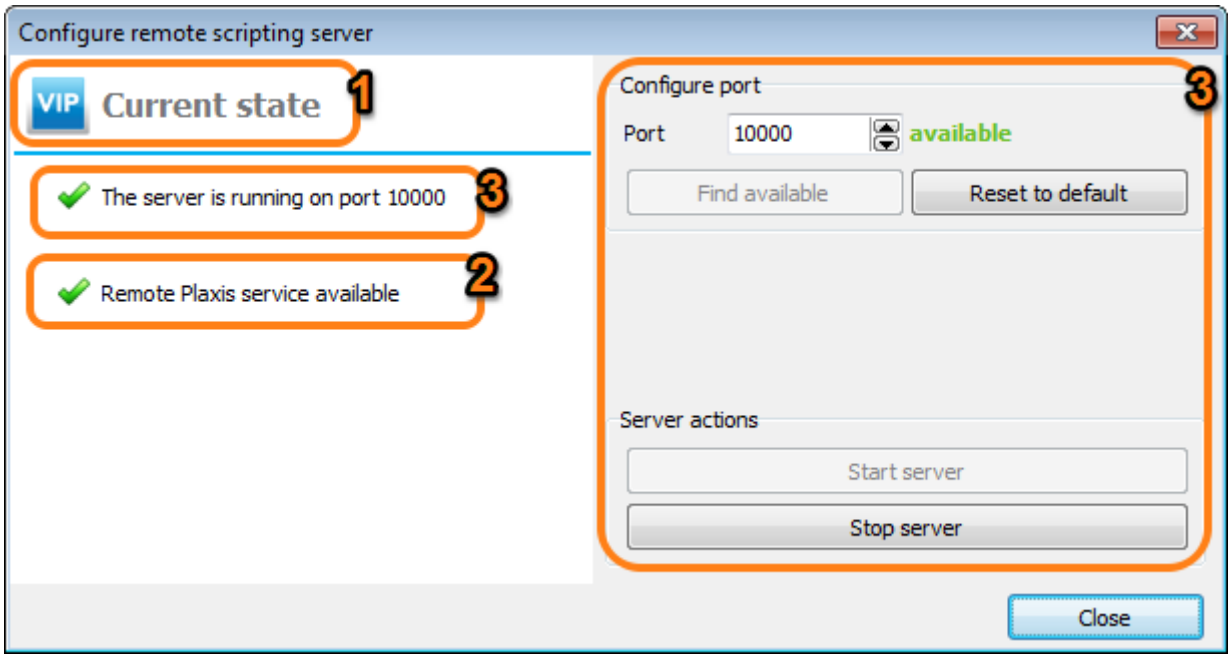

Figure 2. Run the remote scripting server requirements:
1) valid licence, 2) connection with the Remote Plaxis service website
and 3) a local connection port

Boilerplate code

In order to start your first Python script, you must make sure that your Python script can communicate with the PLAXIS application. For this, the Plaxis scripting library wrapper needs to be imported as a module and the correct port needs to be set. For a locally running PLAXIS application this can easily be achieved using the following boilerplate Python code to start your Python script with (assuming the scripting server port was set to 10000, see above):

```
localhostport = 10000
plaxis_path = r'C:\Program Files (x86)\Plaxis\PLAXIS 2D' #no trailing backslash!

import imp
found_module = imp.find_module('plxscripting', [plaxis_path])
plxscripting = imp.load_module('plxscripting', *found_module)
from plxscripting.easy import *

s_i, g_i = new_server('localhost', localhostport)
```

In this, we need to set the correct localhost port using the same number as set in the Configure remote scripting server window. Next we need the location where the PLAXIS application is installed: plaxis_path. From this location we will import the plxscripting module to allow for the communication with the Plaxis application.

Now we have two new variables:

- **s_i**: this is bound to an object representing the PLAXIS Input Application (the Remote scripting ***server***).
- **g_i**: this variable is bound to the ***global object*** of the current open Plaxis model in Input. This should be used when you want to make changes to the model (e.g. adding a borehole or a point, generating the mesh and start the calculation).

PLAXIS Input and PLAXIS Output

Note, if you want to use both PLAXIS Input and PLAXIS Output in your Python script, you will need to setup the connection with Input and Output separately by binding the server and the global object for Input and Output separately using each their own port number:

```
#[...]
localhostport_input = 10000
localhostport_output = 10001
s_i, g_i = new_server('localhost', localhostport_input )
s_o, g_o = new_server('localhost', localhostport_output )
```

Trouble shooting

If your script gives an error, please check the following:

- Is the PLAXIS application running?
- Is the Remote Scripting server inside PLAXIS activated with the same port number as used in the Python script?
- Is the firewall blocking communications with the PLAXIS application?

For more details on this Python wrapper and some code examples, please see the appendix on the Python HTTP REST API wrapper in the PLAXIS Reference manual and the related links.