# Debugging Python Code in Natlink

Debugging Python Code in Natlink can be tricky. It is helpful to attach a debugger such as Visual Studio Code, but there are several configuration steps to make it work.

## [Debug Adapter Protocol](https://microsoft.github.io/debug-adapter-protocol/) (DAP) compatible debuggers

A module pydebug is used to implement DAP in Python. Unless you find another tool supporting DAP, your only choice is Visual Studio Code for debugging. Or you might get some mileage out of [DAP Mode (emacs-lsp.github.io)](https://emacs-lsp.github.io/dap-mode/).

### Configuring Natlink

#### To Enable Attach the Debugger with a Working Unimacro

This is the simplest, if you have or can set up a working Unimacro installation. If you need to debug before Unimacro is up and running, see below.

You don’t need to set environment variables. The default port of 7474 will be used, unless there is an environment variable ‘NatlinkPyDebugPort’ which will override the port.

Enable the grammar “debug\_natlink” which is in \_debug\_natlink.py.

|  |  |  |
| --- | --- | --- |
| Goal | Utterance | Result |
| Enable debugger attachment. | Debug Code Start” | Natlink listens for DAP on the default port. |
| Find Natlink’s DAP status. | Debug Code Info | Returns information about DAP (port number, whether it is running) |
|  |  |  |

There are a number of compatible debuggers. Visual Studio Code is readily available. Kombodo and Pycharm (as of 2020-11-24) do not support DAP. So you if you need to debug, it is easiest to install Visual Studio Code. A module called ‘pydebug’ is used by natlink to allow the debugger to connect via DAP.

Natlink (via DAP) will listen on a socket for debugger connections. You need to be aware of any network security implications of having a program listen on a socket. For example, you could configure a firewall to allow debugging across the internet.

There are three methods to enable the DAP and listen for debug connections.

* Start and attach the debugger when natlinkmain.py is loaded.
* Start and attach the debugger anytime you want after natlink is started
* If Unimacro is working in your install, enable the debugger to attach to natlink (by registrying ‘pydebug’ debugging through a voice command, then attach the debugger when you want.

#### To Enable Attaching the Debugger without Unimacro

You would choose this mainly if Unimacro were not installed on your system or not working. When natlink starts, it will be listening to a socket immediately whether or not a debugger attaches.

Create the environment variable ‘NatlinkPyDebugPort’ and set it to the desired port number; the recommended port is 7474.

Create the environment variable ‘NatLinkPython’ and set to point the default Python Executable for the python environment running natlink. i.e. “C:\Python38-32\Python.exe” if that is where the python environment for natlink lives. This needs to have access to the same libraries in site-packages as natlink; if you are using a virtual environment for natlink, set the variable to the full path of the Python.exe in the virtual environment. Natlink, via debugpy, will launch the python executable as part of the debug connection process. It the python executable doesn’t launch or doesn’t match natlink’s python you will see python traceback errors on the natlink window.

**Only if you want the natlink** to wait for the debugger at launch, and break as soon as the debugger is attached: Create the following environment variable ‘NatlinkPyDebugStartup’ and set it to ‘Y’ or ‘y’ or ‘T’ or ‘t’. any other value will be ignored, so set it to ‘n’ to disable debugging at startup.

#### To Enable Attach the Debugger with a Working Unimacro

This is the simplest, if you have a working Unimacro installation.

You don’t need to set environment variables. The default port of 7474 will be used, unless there is an environment variable ‘NatlinkPyDebugPort’ which will override the port.

#### Configuring the Debugger

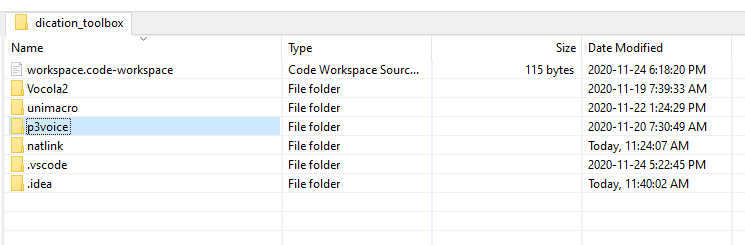
Instructions are only provided for Visual Studio Code. If your debugger supports DAP and debugpy and you get it working, consider adding instructions to this document.

##### Visual Studio Code

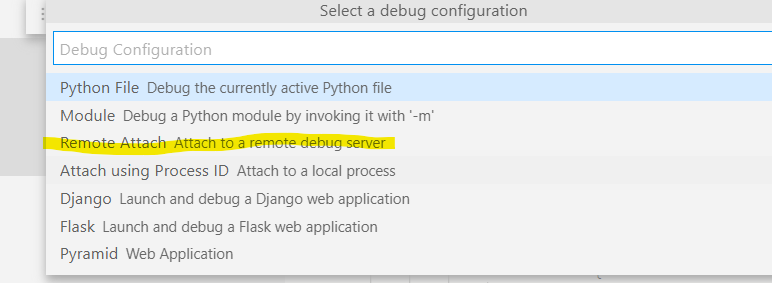
This configuration has worked. You might find other configurations that work ok.

Create a workspace at the root of your dictation-toolbox code. It is a good idea to put your Unimacro, Natlink, and Vocola user folders below this root as well, so that you can debug them. If they are below your Visual Studio Code project root, you may not be able to set breakpoints.

In this example they are all in a subdirectory p3voice. DO NOT point your user folders to your Vocoal2 or Unimacro source code directories.



Ensure the python path for your workspace is set to you python install used with natlink.

Add a debug configuration and select python remote attach configuration . 

This will create or amend a launch.json file in your project. Change the name if you want, as shown below – it does not matter. The host should remain localhost and the port set to 7474 or whatever port you decide to use for debugging.

The path mappings should be as shown below. The defaults will not work correctly; if you don’t change them as shown, your debugger will connect but your breakpoints wont’ work.

{

*// Use IntelliSense to learn about possible attributes.*

*// Hover to view descriptions of existing attributes.*

*// For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387*

    "version": "0.2.0",

    "configurations": [

        {

            "name": "Python: Natlink Attach",

            "type": "python",

            "request": "attach",

            "connect": {

                "host": "localhost",

                "port": 7474

            },

            "pathMappings": [

                {

                    "localRoot": "${workspaceFolder}",

                    "remoteRoot": "${workspaceFolder}"

                }

            ]

        }

    ]

}

## The Code

The code for enabling the debug support is in ../macrosystem/core/debug\_natlink.py. If you know how to add support for other debugging mechanisms, please go ahead and add them, and update the unimacro grammar too.

The unimacro grammar is in \_debug\_natlink.py