
Natlink Documentation

Release 5.5.5

Quintijn Hoogenboom

Sep 23, 2023

CONTENTS:

- 1 Introduction** **3**
- 2 Using a Published Grammar** **5**
- 3 Installing a Published Grammar** **7**
- 4 Install instructions for Natlink** **9**
- 5 Configure Natlink** **11**
 - 5.1 Location of 'natlink.ini' 11
 - 5.2 Adding directories to the configuration 11
 - 5.3 User Directories 12
 - 5.4 Set the log_level 12
 - 5.5 Manual editing natlink.ini 12
 - 5.6 Restart Dragon 12
- 6 Modules** **13**
- 7 Developers instructions** **15**
 - 7.1 Setup Visual Studio Code environment 15
 - 7.2 Further instructions 16
 - 7.3 Invalid options Visual Studio 16
 - 7.4 Fix 16
- 8 Project** **19**
 - 8.1 Documentation 19
- 9 Indices and tables** **21**

Natlink is an OpenSource extension module for the speech recognition program [Dragon](<https://www.nuance.com/dragon/business-solutions/dragon-professional-individual.html>).

Natlink with Python 3 and Dragon 13, 14, 15, or 16 is now in a beta phase for:

- Dragonfly
- Vocola and
- User defined grammars.

Unimacro is not yet working, a beta release for this state will be released pretty soon. [Version for Python 2.7](<https://qh.antenna.nl/unimacro/installation/installationstableversionpython2.html>).

INTRODUCTION

Natlink has been used to extend Dragon Naturally Speaking (Dragon) with Python modules. Many developers of extensions for Dragon find it more practical to develop and share extensions using Python than the built in Visual Basic macros of Dragon.

Similarly, many end users find it easier to use Dragon extensions written in Natlink than to locate and install scripts using the built in Dragon scripting. However, an end user may have to edit or rename the occasional text file, or run a command line program *pip* to install a Python package.

End users interact with an extension with a Natlink grammar. A grammar specifies what will happen when certain words are dictated with grammar rules.

For example, a hypothetical grammar could have a very simple rule “american date today” which prints the current date out in a format mm/dd/yyyy.

Grammars can be used to insert text boilerplate, operate the menus of programs, or otherwise control a computer. They can even be used to help write computer programs.

USING A PUBLISHED GRAMMAR

INSTALLING A PUBLISHED GRAMMAR

Follow the instructions for the specific grammar or extension. If the author has published the package `xyz` to PyPI <https://pypi.org/> you will be able to install it with a command line ‘`pip install xyz`’ in a command shell or powershell prompt and add the necessary lines to `natlink.ini` (something like `xyz=xyz`).

They might provide other instructions on installing the package with `pip`, or just provide a python file for you to figure out what to deal with. In the case where they just provide you python files, the python files module must be placed in a folder listed in the `directories` section of `natlink.ini`.

As an end user of Natlink, you may never need to write your own grammar, and you won’t need programming skills. You will need to Python packages and grammars and perhaps do some small amount of configuration on your computer.

Grammars are implemented as Python modules. A Python module is a single Python file which is identified with the “.py” extension. Natlink needs to know to load these Python modules. This is done through the `TODO` insert hyperlink configuration file `natlink.ini`.

`natlink.ini` has a `[directories]` setting that lists the directories (file folders) natlink loads grammars from. You can add as many as you need.

A line may look like `name = directory`

The name on the left of the equals is just a one word descriptor, and natlink doesn’t do much with the name. Sometimes confusingly they may be the same or similar as the directory on the right of the equals.

The directory can be: - the name of Python package you have installed. Generally this is the easiest if someone has published a grammar as a Python package. - a folder on your file system where you have placed the Python file OR where an extension will place Python files.

In any case, natlink will load all the python code in all the directories listed in the `natlink.ini` config file.

It can be confusing because you might see two directories with similar names. For example, `Vocola` uses two directories in `natlink.ini`. `Natlink.ini` will have two lines for `Vocola`:

```
[directories] vocoladirectory = vocola2 vocolagrammarsdirectory = natlink_userdirvocolagrammars
```

This is because `vocola` has a bunch of Python that needs to be loaded from the `vocola2` Python package and a bunch more Python that `vocola` itself creates at runtime and places in `vocolagrammarsdirectory`. Some other natlink extensions do the same thing.

INSTALL INSTRUCTIONS FOR NATLINK

- Install Dragon NaturallySpeaking
- Optional: install Python 3.10 (32 bit required) for your user and do not add to path. This will also be done, if necessary, when you run the installer program, but the path where python is installed may be less convenient.
- Download and run the latest [Natlink installer](<https://github.com/dictation-toolbox/natlink/releases>): choose the most recent release, at the top of the page, and then scroll down and choose the file `natlink-?.?.?-py3.10-32-setup.exe`.
- As last step of the install procedure, the program `natlinkconfig_gui` is started. When you want to run this program later (again), type the program name at the Windows start program prompt.
- When using ‘dragonfly’, ‘vocola’ and/or ‘unimacro’, these packages are installed via `pip` with the config program. When you need other packages, you can install these with `pip`.
- Then start Dragon, the ‘Messages from Natlink’ window should show up.
- See configure Natlink for more details.
- Also see [installation instructions](<https://github.com/dictation-toolbox/natlink/>) (scroll down a bit...)

CONFIGURE NATLINK

5.1 Location of ‘natlink.ini’

The config file *natlink.ini* will by default be in `~\.natlink`, so a subdirectory of your home directory, typically `C:\Users\Name`.

When you run the config program *natlinkconfig_gui.py*, the next step is automatically performed:

- copy the file *natlink.ini* from the ‘fallback’ location into your home directory, subdirectory *.natlink*. So copy “C:\Program Files (x86)\Natlink\DefaultConfig\natlink.ini” to “C:\Users\Name\.natlink\natlink.ini”.
- When you want another directory for your file *natlink.ini* to be in, you can set this in the environment variable ‘NATLINK_USERDIR’. For example, when you want your config files (especially *natlink.ini*) in a subdirectory of your Documents folder, set this environment variable to ‘~/Documents/.natlink’.

5.2 Adding directories to the configuration

This will be done with the config program (*natlinkconfig_gui.py*), or with the Command Line Interface *natlinkconfig_cli.py*. Both programs can be started from the Windows Command Line.

Note: the Command Line Interface can also be used in batch mode.

Special directory directives:

- “~” points to your ‘home’ directory, most often ‘C:\Users\Name’. This is one directory above your ‘Documents’ directory. The config program translates chosen paths to this better readable ‘~’, when possible.
- “natlink_userdir”: this “variable” points to the directory where your config file, ‘natlink.ini’, is located. By default this is the directory ‘.natlink’ in your home directory. But you can set the environment variable NATLINK_USERDIR, see above. Note: the path should always end with the directory ‘.natlink’! For some “automatic defined” directories, especially for *unimacro* and *vocola*, this ‘natlink_userdir’ is also used. The directory MUST be a local directory.

5.3 User Directories

Other directories, “user directories”, can be set on a network drive (Dropbox, OneDrive), if you prefer. When you enable ‘Dragonfly’, ‘Vocola’ or ‘Unimacro’ (or any of them), the config program will also install and if possible update the required package. When you want to force an upgrade of a package, first disable the package and then enable again.

Dragonfly

The user directory is where you can put your python grammar files. Previous, the Natlink user directory was defined for this. You could keep using this directory too, but install/upgrade of Dragonfly is not maintained in that case.

Vocola

By specifying a directory where your user command files will be located (.vcl), you enable Vocola.

Unimacro

By specifying a directory where your user config files (.ini) will be located, you enable Unimacro. Note Unimacro is not ready for a release yet.

NatlinkUser

The “package independent” Natlink user directory can also be specified as a directory where you can put your python grammar files.

5.4 Set the log_level

You can set the log_level, controlling the abundance of information messages in the “Messages from Natlink” window with the following option (choices are DEBUG, INFO, WARNING).

```
[settings]
log_level = INFO
```

5.5 Manual editing natlink.ini

You can always manually inspect and adapt your ‘natlink.ini’ file.

5.6 Restart Dragon

After making changes in your configuration, always restart Dragon! Keep your eye on the ‘Messages from Natlink’ window!

MODULES

The python modules of the Natlink project are now in repository *natlinkcore*, see ASAP natlinkcore.readthedocs.io

DEVELOPERS INSTRUCTIONS









When you want to contribute to the Natlink development, you will need to compile the C++ code and compile the inno setup program. Try the instructions below.

7.1 Setup Visual Studio Code environment

1. Install [Visual Studio](#) (Community Edition 2019 or above) with C++ Desktop Development and Microsoft Visual C++ Redistributable.
 - [C++ Desktop Development](#) This contains the necessary compilers for (**Visual Studio** and **Visual Studio Code**)
 - [Microsoft Visual C++ Redistributable 2015, 2017, 2019, and 2022](#) (32-bit/X86 required)
2. Install [Visual Studio Code](#) with the following Extensions
 - [C/C++](#)
 - [CMake](#)
3. Install [Inno](#) version 6.x.
4. Install [Python](#) version 3.8.x, 3.9.x, or 3.10.x 32-bit/X86 (Does not need to be on path)
5. After cloning Nalink open the project up in Visual Studio Code
 - Set the Python Version `PYTHON_VERSION 3.10` in [CMakeLists.txt](#) (The CMakeLists.txt in top directory of the project)
 - example for Python 3.8.x set(`PYTHON_VERSION 3.8 CACHE STRING "3.X for X >= 8"`)
 - Selective equivalent to [Visual Studio Community 2022 Release - x86](#) (32-bit/X86 required) to configure the compiler.



–  master  0 0  Live Share  CMake: [Debug]: Ready  [Visual Studio Community 2022 Release - x86]  Build [ALL_BUILD]  

6. The build directory will generate containing the configuration selected and build artifacts (compiled code and installer)
 - The build directory can be safely deleted if you need to reconfigure the project as it will just regenerate.
7. Click the “build” button at the bottom of the editor to to build the project and create the installer.
 -  master  0 0  Live Share  CMake: [Debug]: Ready  [Visual Studio Community 2022 Release - x86]  Build [ALL_BUILD]  
 - build directory Installer location `{project source directory}\build\InstallerSource\natlink5.1-py3.10-32-setup.exe`

7.2 Further instructions

7.3 Invalid options Visual Studio

When the C++ compile redistributable is wrongly configured, the program *dumpbin.exe* reports a dependency, which is not wanted:

```
PS C:\dt\NatlinkDoc\natlink\documentation> ."C:\Program Files (x86)\Microsoft Visual
↪Studio\2019\Community\VC\Tools\MSVC\14.29.30133\bin\Hostx86\x86\dumpbin.exe" /
↪DEPENDENTS "C:\Program Files (x86)\Natlink\site-packages\natlink\_natlink_core.pyd"
Microsoft (R) COFF/PE Dumper Version 14.29.30136.0
Copyright (C) Microsoft Corporation. All rights reserved.

Dump of file C:\Program Files (x86)\Natlink\site-packages\natlink\_natlink_core.pyd

File Type: DLL

Image has the following dependencies:

python38.dll
KERNEL32.dll
USER32.dll
SHELL32.dll
ole32.dll
OLEAUT32.dll
ADVAPI32.dll
MSVCP140D.dll
VCRUNTIME140D.dll
ucrtbased.dll

(...)
```

The *VCRUNTIME140D.dll* should not be there.

7.4 Fix

Static linking is established by installing: <https://docs.microsoft.com/en-us/cpp/c-runtime-library/crt-library-features?view=msvc-170&viewFallbackFrom=vs-2019>

Also see “Bundling vc redistributables”: <https://stackoverflow.com/questions/24574035/how-to-install-microsoft-vc-redistributables-silently-in-inno-setup>

With install version 5.1.1 (with python 3.8), now the following output is given:

```
(Powershell) ."C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\
↪MSVC\14.29.30133\bin\Hostx86\x86\dumpbin.exe" /DEPENDENTS "C:\Program Files (x86)\
↪Natlink\site-packages\natlink\_natlink_core.pyd"
Dump of file C:\Program Files (x86)\Natlink\site-packages\natlink\_natlink_core.pyd

File Type: DLL
```

(continues on next page)

(continued from previous page)

Image has the following dependencies:

```
python38.dll
KERNEL32.dll
USER32.dll
SHELL32.dll
ole32.dll
OLEAUT32.dll
ADVAPI32.dll
```

(...)

So issue#86(<https://github.com/dictation-toolbox/natlink/issues/86>) is hopefully solved and explained with this all.

8.1 Documentation

The documentation for Natlink is written in [reStructuredText format](#). ReStructuredText is similar to the Markdown format. If you are unfamiliar with the format, the [reStructuredText primer](#) might be a good starting point.

The [Sphinx documentation engine](#) and [Read the Docs](#) are used to generate documentation from the *.rst* files in the *documentation/* folder. Docstrings in the source code are included in a semi-automatic way through use of the [sphinx.ext.autodoc](#) extension.

To build the documentation locally, install Sphinx and any other documentation requirements:

```
$ cd documentation
$ pip install -r requirements.txt
```

Then run the following command on Windows to build the documentation:

```
$ make.bat html
```

Or use the Makefile on other systems:

```
$ make html
```

If there were no errors during the build process, open the *_build/html/index.html* file in a web browser. Make changes, rebuild the documentation and reload the doc page(s) in your browser as you go.

INDICES AND TABLES

- [genindex](#)
- [modindex](#)
- [search](#)