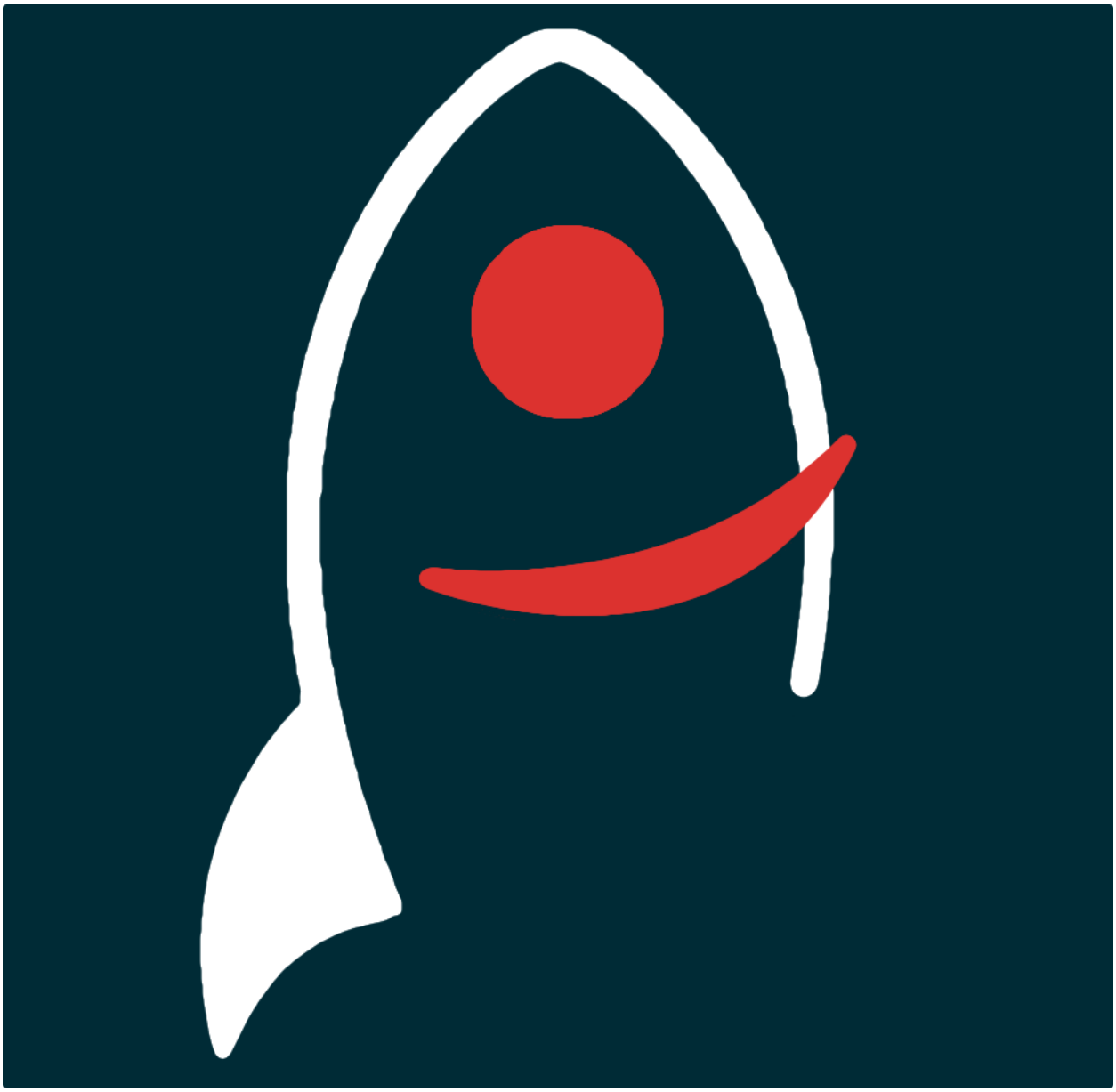


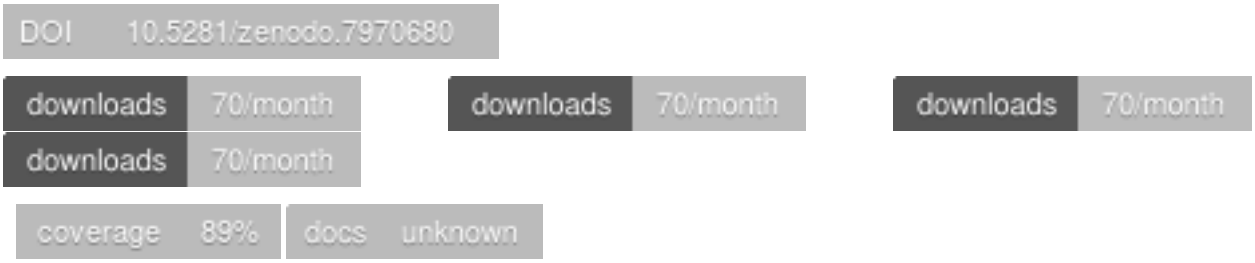
transientNamer Documentation

Release v0.4.6

Dave Young

2023

TABLE OF CONTENTS



Python API for reading and caching TNS reports.

Documentation for transientNamer is hosted by Read the Docs (development version and master version). The code lives on github. Please report any issues you find here.

CHAPTER

ONE

FEATURES

-

CHAPTER

TWO

HOW TO CITE TRANSIENTNAMER

If you use `transientNamer` in your work, please cite using the following BibTeX entry:

```
@software{Young_transientNamer,
    author = {Young, David R.},
    doi = {10.5281/zenodo.7970680},
    license = {GPL-3.0-only},
    title = ,
    url = {https://github.com/thespacedoctor/transientNamer}
}
```

2.1 Installation

The easiest way to install transientNamer is to use `pip` (here we show the install inside of a conda environment):

```
conda create -n transientNamer python=3.7 pip
conda activate transientNamer
pip install transientNamer
```

Or you can clone the github repo and install from a local version of the code:

```
git clone git@github.com:thespacedoctor/transientNamer.git
cd transientNamer
python setup.py install
```

To upgrade to the latest version of transientNamer use the command:

```
pip install transientNamer --upgrade
```

To check installation was successful run `transientNamer -v`. This should return the version number of the install.

2.1.1 MySQL

If you wish to make use of the tools that allow interaction with a MySQL database you have to have MySQL/Maria DB server installed and access to a database. Credentials to the database are to be added to the settings file (see `initialisation`).

You also need to install PyMySQL into your conda environment via:

```
conda install pymysql
```

2.1.2 Development

If you want to tinker with the code, then install in development mode. This means you can modify the code from your cloned repo:

```
git clone git@github.com:thespacedoctor/transientNamer.git
cd transientNamer
python setup.py develop
```

Pull requests are welcomed!

2.2 Initialisation

Before using transientNamer you need to use the `init` command to generate a user settings file. Running the following creates a yaml settings file in your home folder under `~/.config/transientNamer/transientNamer.yaml`:

```
transientNamer init
```

The file is initially populated with transientNamer's default settings which can be adjusted to your preference.

If at any point the user settings file becomes corrupted or you just want to start afresh, simply trash the `transientNamer.yaml` file and rerun `transientNamer init`.

2.2.1 Modifying the Settings

Once created, open the settings file in any text editor and make any modifications needed. For example:

```
database settings:
    db: myDB
    host: localhost
    user: dbuser
    password: dbpass

astronote-cache: ~/Desktop/astronotes

# FIND IN YOUR USER ACCOUNT SETTING ON TNS : https://www.wis-tns.org
user-agent: 'tns_marker{"XXXX"}'
```

You will need to register to the TNS to get a unique `tns-marker` from your account settings on https://www.wis-tns.org. Once you have the `tns-marker` add it to your settings file.

2.2.2 Basic Python Setup

If you plan to use `transientNamer` in your own scripts you will first need to parse your settings file and set up logging etc. One quick way to do this is to use the `fundamentals` package to give you a logger, a settings dictionary and a database connection (if connection details given in settings file):

```
## SOME BASIC SETUP FOR LOGGING, SETTINGS ETC
from fundamentals import tools
from os.path import expanduser
home = expanduser("~")
settingsFile  = home + "/.config/transientNamer/transientNamer.yaml"
su = tools(
    arguments={"settingsFile": settingsFile},
    docString=__doc__,
)
arguments, settings, log, dbConn = su.setup()
```

2.3 Caching and Parsing Astronotes

Before parsing the Astronote contents it's necessary to download the notes into a local cache to avoid redownloading every time you wish to parse them and (reduces strain placed on the TNS). Before you begin make sure you have set up the `astronote-cache` setting in the settings file (see `initalisation` instructions).

In the spirit of fair-usage the downloader will pause for 1 second in-between each individual note download.

2.3.1 From the Command-Line

To cache the notes run the command:

```
transientNamer notes <reportedInLastDays>
```

So to cache notes from the last 10 days run `transientNamer notes 10`.

To also parse and import the cached notes into 3 MySQL database tables (`astronotes_content`, `astronotes_keywords`, `astronotes_transients`) run the same command but with the `--import` flag:

```
transientNamer --import notes <reportedInLastDays>
```

This will cause all unseen notes in the cache directory to be parsed and added to the database.

2.3.2 From Python Script

To `download` the notes from the last 30 days use the following snippet.

```
from transientNamer import astronotes
an = astronotes(
    log=log,
    dbConn=dbConn,
    settings=settings
)
downloadCount = an.download(
    cache_dir=settings["astronote-cache"], inLastDays=30)
print(f"{downloadCount} new astronotes downloaded and cached")
```

And to then parse the notes to the database tables add:

```
an.notes_to_database()
```

2.4 Todo List

Todo:

- nice!

(The *original entry* is located in /home/docs/checkouts/readthedocs.org/user_builds/transientnamer/checkouts/develop/docs/source/_tem
line 1.)

2.5 Release Notes

v0.4.6 - July 3, 2023

- **FIXED**: small fixes to unit-tests

v0.4.5 - January 10, 2023

- **FIXED**: changes to search url to reflectv changes made on the TNS. All transients should now be reported again.

v0.4.4 - June 17, 2022

- **FIXED**: unit-tests failing due to throttling from TNS

v0.4.3 - January 25, 2021

- **FIXED**: call to settings file now explicit from command-line (was not required until tns-marker added)

v0.4.2 - January 25, 2021

- **REFACTOR:** requests will fail if TNS responds with any status other than 200
- **REFACTOR:** added user-agent header now required by TNS (needs added in the settings file)

v0.4.1 - January 25, 2021

- **FEATURE**:** you can now download and parse astronotes (including ability to add to MySQL database tables)
- **REFACTOR:** Updated URL to the new TNS URL https://www.wis-tns.org

v0.3.2 - December 21, 2020

- **ENHANCEMENT**: now using discovered_period_value for discInLastDays. More efficient than adding range of obsdate.

v0.3.1 - October 25, 2020

- **FIXED:** bytes to string with utf-8 encoding bug stopping regexes from passing

v0.3.0 - May 22, 2020

- Now compatible with Python 3.*

CHAPTER

THREE

API REFERENCE

3.1 Modules

transientNamer.commonutils	*common tools used throughout package*
transientNamer.utKit	*Unit testing tools*

3.1.1 commonutils *(module)*

common tools used throughout package

Functions

getpackagepath()	*getpackagepath*

Sub-modules

getpackagepath()	*getpackagepath*

3.1.2 utKit *(module)*

Unit testing tools

Classes

utKit(moduleDirectory[, dbConn])	*Override dryx utKit*

Sub-modules

utKit(moduleDirectory[, dbConn])	*Override dryx utKit*

class utKit(*moduleDirectory*, *dbConn=False*)
Bases: fundamentals.utKit.utKit

Override dryx utKit

get_project_root()
Get the root of the ``python`` package - useful for getting files in the root directory of a project

Return

- rootPath – the root path of a project

refresh_database()
Refresh the unit test database

setupModule()
The setupModule method

Return

- log – a logger
- dbConn – a database connection to a test database (details from yaml settings file)
- pathToInputDir – path to modules own test input directory
- pathToOutputDir – path to modules own test output directory

tearDownModule()
The tearDownModule method

3.2 Classes

transientNamer.astronotes	*Tools to download, parse and ingest astronote content into a MySQL database*
transientNamer.search	*Search the Transient Name Server with various search constraints*

3.2.1 astronotes *(class)*

class astronotes(*log*, *dbConn=False*, *settings=False*)
Bases: object

Tools to download, parse and ingest astronote content into a MySQL database

Key Arguments:

- log – logger
- dbConn – database connection. Default *False*
- settings – the settings dictionary

Usage:

To setup your logger, settings and database connections, please use the `fundamentals` package (see tutorial here).

To initiate a astronotes object, use the following:

```
from transientNamer import astronotes
an = astronotes(
    log=log,
    dbConn=dbConn,
    settings=settings
)
```

Methods

download(cache_dir[, inLastDays])	*Download astronotes reported in the lasy N days.
get_all_noteids([inLastDays])	*get the noteids of those notes released in the last N days*
notes_to_database()	*read the notes and import them into indexed MySQL database tables*

download(*cache_dir*, *inLastDays=False*)

Download astronotes reported in the lasy N days. Check cache for notes alreaedy downloaded.

Key Arguments:

```
- `cache_dir` -- the directory to cache the json notes to.
- `inLastDays` -- download only notes reported in the last N days. Default␣
↪*False*. (Download all)
```

Return:

```
- `downloadCount` -- number of new files cached
```

Usage:

```
from transientNamer import astronotes
an = astronotes(
    log=log,
    dbConn=dbConn,
    settings=settings
)
downloadCount = an.download(
    cache_dir=settings["astronote-cache"], inLastDays=30)
print(f"{downloadCount} new astronotes downloaded anc cached")
```

get_all_noteids(*inLastDays=False*)

get the noteids of those notes released in the last N days

Key Arguments:

- `inLastDays` – report only notesIds released in the last N days. Default *False*. (Report all)

Return:

```
- `noteIds` -- list of all reported noteIds
```

Usage:

```
from transientNamer import astronotes
an = astronotes(
    log=log,
    settings=settings
)
noteIds = an.get_all_noteid(inLastDays=3000)
print(f"Astronote IDs: {noteIds}")
```

notes_to_database()

read the notes and import them into indexed MySQL database tables

Usage:

```
from transientNamer import astronotes
an = astronotes(
    log=log,
    dbConn=dbConn,
    settings=settings
)
an.notes_to_database()
```

3.2.2 search *(class)*

class search(*log*, *ra=''*, *dec=''*, *radiusArcsec=''*, *name=''*, *discInLastDays=''*, *settings=False*, *comments=False*)

Bases: `object`

Search the Transient Name Server with various search constraints

Key Arguments

- `log` – logger
- `settings` – the settings dictionary
- `ra` – RA of the location being checked
- `dec` – DEC of the location being searched
- `radiusArcsec` - the radius of the conesearch to perform against the TNS
- `name` – name of the object to search the TNS for
- `discInLastDays` – search the TNS for transient reported in the last X days
- `comments` – print the comments from the TNS, note these can be long making table outputs somewhat unreadable. Default *False*

Usage

To initiate a search object to search the TNS via an object name (either TNS or survey names accepted):

```
from transientNamer import search
tns = search(
    log=log,
    name="Gaia16bbi"
)
```

or for a conesearch use something similar to:

```
from transientNamer import search
tns = search(
    log=log,
    ra="06:50:36.74",
    dec="+31:06:44.7",
    radiusArcsec=5
)
```

Note the search method can accept coordinates in sexagesimal or decimal defree formats.

To list all new objects reported in the last three weeks, then use:

```
from transientNamer import search
tns = search(
    log=log,
    discInLastDays=21
)
```

Methods

`csv`([dirPath])	*Render the results in csv format*
`json`([dirPath])	*Render the results in json format*
`markdown`([dirPath])	*Render the results in markdown format*
`mysql`([tableNamePrefix, dirPath])	*Render the results as MySQL Insert statements*
`table`([dirPath])	*Render the results as an ascii table*
`yaml`([dirPath])	*Render the results in yaml format*

Properties

`files`	*The associated source files*
`photometry`	*The associated source photometry*
`sources`	*The results of the search returned as a python list of dictionaries*
`spectra`	*The associated source spectral data*
`url`	*The generated URL used for searching of the TNS*

csv(*dirPath=None*)
Render the results in csv format

Key Arguments

- `dirPath` – the path to the directory to save the rendered results to. Default *None*

Return

- `csvSources` – the top-level transient data
- `csvPhot` – all photometry associated with the transients
- `csvSpec` – all spectral data associated with the transients
- `csvFiles` – all files associated with the matched transients found on the tns

Usage

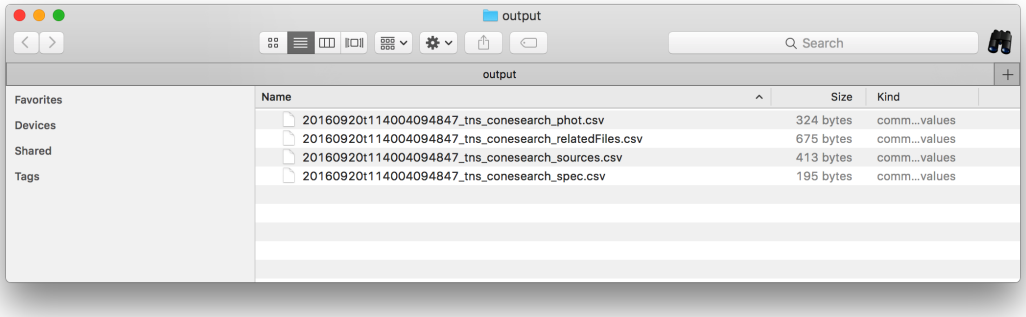
To render the results in csv format:

```
csvSources, csvPhot, csvSpec, csvFiles  = tns.csv()
print(csvSources)
```

```
TNSId,TNSName,discoveryName,discSurvey,raSex,decSex,raDeg,decDeg,
 ↪transRedshift,specType,discMag,discMagFilter,discDate,objectUrl,hostName,
 ↪hostRedshift,separationArcsec,separationNorthArcsec,separationEastArcsec
2016asf,SN2016asf,ASASSN-16cs,ASAS-SN,06:50:36.73,+31:06:45.36,102.6530,31.
 ↪1126,0.021,SN Ia,17.1,V-Johnson,2016-03-06 08:09:36,https://www.wis-tns.org/
 ↪object/2016asf,KUG 0647+311,,0.66,0.65,-0.13
```

You can save the results to file by passing in a directory path within which to save the files to. The four flavours of data (sources, photometry, spectra and files) are saved to separate files but all data can be assoicated with its transient source using the transient's unique `TNSId`.

```
tns.csv("~/tns")
```




```
:width: 800px
:alt: csv output
```

`json`(*dirPath=None*)

Render the results in json format

Key Arguments

- `dirPath` – the path to the directory to save the rendered results to. Default *None*

Return

- `jsonSources` – the top-level transient data
- `jsonPhot` – all photometry associated with the transients
- `jsonSpec` – all spectral data associated with the transients
- `jsonFiles` – all files associated with the matched transients found on the tns

Usage

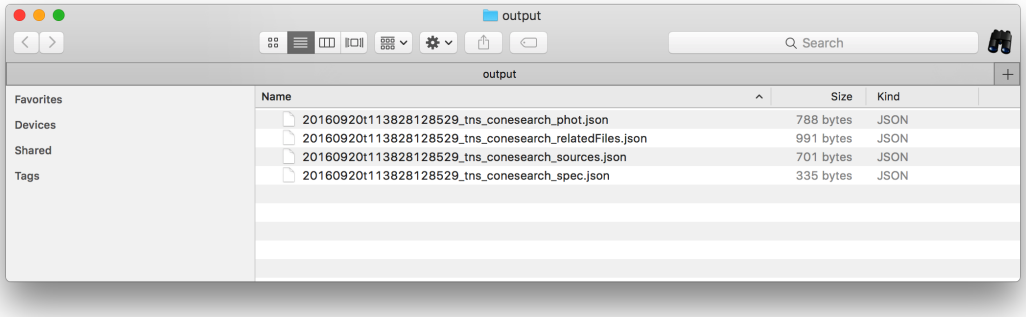
To render the results in json format:

```
jsonSources, jsonPhot, jsonSpec, jsonFiles  = tns.json()
print(jsonSources)
```

```
[
    {
        "TNSId": "2016asf",
        "TNSName": "SN2016asf",
        "decDeg": 31.1126,
        "decSex": "+31:06:45.36",
        "discDate": "2016-03-06 08:09:36",
        "discMag": "17.1",
        "discMagFilter": "V-Johnson",
        "discSurvey": "ASAS-SN",
        "discoveryName": "ASASSN-16cs",
        "hostName": "KUG 0647+311",
        "hostRedshift": null,
        "objectUrl": "https://www.wis-tns.org/object/2016asf",
        "raDeg": 102.65304166666667,
        "raSex": "06:50:36.73",
        "separationArcsec": "0.66",
        "separationEastArcsec": "-0.13",
        "separationNorthArcsec": "0.65",
        "specType": "SN Ia",
        "transRedshift": "0.021"
    }
]
```

You can save the results to file by passing in a directory path within which to save the files to. The four flavours of data (sources, photometry, spectra and files) are saved to separate files but all data can be assoicated with its transient source using the transient's unique `TNSId`.

```
tns.json("~/tns")
```




```
:width: 800px
:alt: json output
```

markdown(*dirPath=None*)
Render the results in markdown format

Key Arguments

- `dirPath` – the path to the directory to save the rendered results to. Default *None*

Return

- `markdownSources` – the top-level transient data
- `markdownPhot` – all photometry associated with the transients

- `markdownSpec` – all spectral data associated with the transients
- `markdownFiles` – all files associated with the matched transients found on the tns

Usage

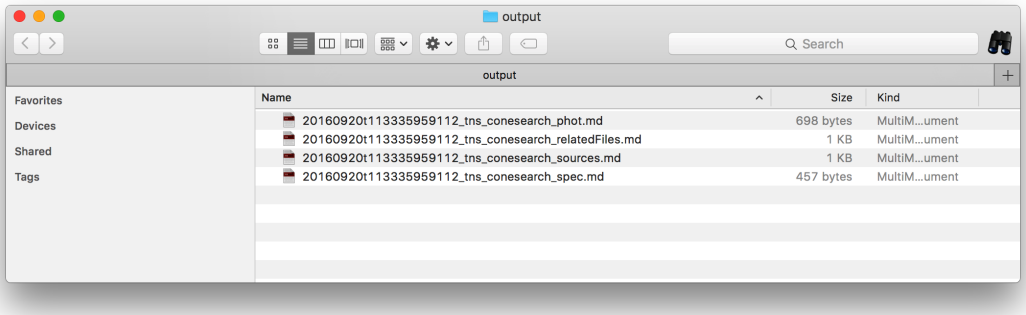
To render the results in markdown table format:

```
markdownSources, markdownPhot, markdownSpec, markdownFiles  = tns.markdown()
print(markdownSources)
```

```
| TNSId    | TNSName    | discoveryName  | discSurvey  | raSex         |␣
↪decSex         | raDeg     | decDeg    | transRedshift  | specType  | discMag␣
↪ | discMagFilter  | discDate              | objectUrl                          ␣
↪              | hostName      | hostRedshift  | separationArcsec  |␣
↪separationNorthArcsec  | separationEastArcsec  |
|:---------|:-----------|:---------------|:------------|:-------------|:------
↪--------|:----------|:---------|:---------------|:----------|:---------|:---
↪------------|:---------------------|:-------------------------------------
↪-------|:-------------|:-------------|:-----------------|:--------------
↪--------|:---------------------|
| 2016asf  | SN2016asf  | ASASSN-16cs    | ASAS-SN     | 06:50:36.73  |␣
↪+31:06:45.36  | 102.6530  | 31.1126  | 0.021          | SN Ia     | 17.1    ␣
↪ | V-Johnson      | 2016-03-06 08:09:36  | https://www.wis-tns.org/object/
↪2016asf  | KUG 0647+311  |               | 0.66             | 0.65        ␣
↪          | -0.13                |
```

You can save the results to file by passing in a directory path within which to save the files to. The four flavours of data (sources, photometry, spectra and files) are saved to separate files but all data can be assoicated with its transient source using the transient's unique `TNSId`.

```
tns.markdown("~/tns")
```




```
:width: 800px
:alt: markdown output
```

`mysql`(*tableNamePrefix='TNS', dirPath=None*)
Render the results as MySQL Insert statements

Key Arguments

- `tableNamePrefix` – the prefix for the database table names to assign the insert statements to. Default *TNS*.
- `dirPath` – the path to the directory to save the rendered results to. Default *None*

Return

- `mysqlSources` – the top-level transient data
- `mysqlPhot` – all photometry associated with the transients
- `mysqlSpec` – all spectral data associated with the transients
- `mysqlFiles` – all files associated with the matched transients found on the tns

Usage

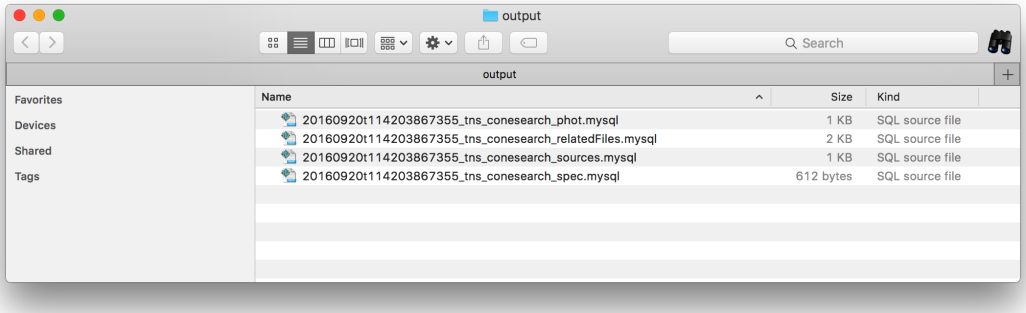
To render the results in mysql insert format:

```
mysqlSources, mysqlPhot, mysqlSpec, mysqlFiles  = tns.mysql("TNS")
print(mysqlSources)
```

```
INSERT INTO `TNS_sources` (TNSId,TNSName,dateCreated,decDeg,decSex,discDate,
↪discMag,discMagFilter,discSurvey,discoveryName,hostName,hostRedshift,
↪objectUrl,raDeg,raSex,separationArcsec,separationEastArcsec,
↪separationNorthArcsec,specType,transRedshift) VALUES ("2016asf" ,"SN2016asf
↪" ,"2016-09-20T11:22:13" ,"31.1126" ,"+31:06:45.36" ,"2016-03-06 08:09:36" ,
↪"17.1" ,"V-Johnson" ,"ASAS-SN" ,"ASASSN-16cs" ,"KUG 0647+311" ,null ,
↪"https://www.wis-tns.org/object/2016asf" ,"102.653041667" ,"06:50:36.73" ,
↪"0.66" ,"-0.13" ,"0.65" ,"SN Ia" ,"0.021")  ON DUPLICATE KEY UPDATE  TNSId=
↪"2016asf", TNSName="SN2016asf", dateCreated="2016-09-20T11:22:13", decDeg=
↪"31.1126", decSex="+31:06:45.36", discDate="2016-03-06 08:09:36", discMag=
↪"17.1", discMagFilter="V-Johnson", discSurvey="ASAS-SN", discoveryName=
↪"ASASSN-16cs", hostName="KUG 0647+311", hostRedshift=null, objectUrl=
↪"https://www.wis-tns.org/object/2016asf", raDeg="102.653041667", raSex=
↪"06:50:36.73", separationArcsec="0.66", separationEastArcsec="-0.13",␣
↪separationNorthArcsec="0.65", specType="SN Ia", transRedshift="0.021",␣
↪updated=1, dateLastModified=NOW() ;
```

You can save the results to file by passing in a directory path within which to save the files to. The four flavours of data (sources, photometry, spectra and files) are saved to separate files but all data can be assoicated with its transient source using the transient's unique `TNSId`.

```
tns.mysql("TNS", "~/tns")
```




```
:width: 800px
:alt: mysql output
```

table (*dirPath=None*)
Render the results as an ascii table

Key Arguments

- `dirPath` – the path to the directory to save the rendered results to. Default *None*

Return

- `tableSources` – the top-level transient data
- `tablePhot` – all photometry associated with the transients
- `tableSpec` – all spectral data associated with the transients
- `tableFiles` – all files associated with the matched transients found on the tns

Usage

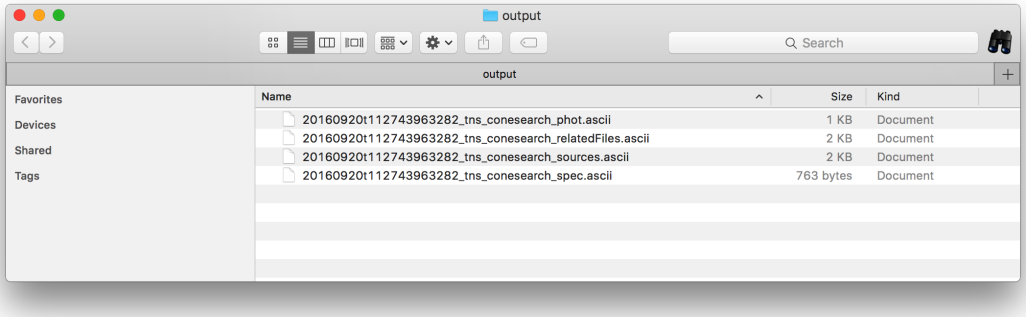
To render the results in ascii table format:

```
tableSources, tablePhot, tableSpec, tableFiles  = tns.table()
print(tableSources)
```

```
+----------+------------+----------------+-------------+--------------+-------
↪--------+-----------+----------+----------------+-----------+----------+----
↪------------+----------------------+-----------------------------------------
↪-------+---------------+---------------+------------------+----------------
↪--------+-----------------------+
| TNSId    | TNSName    | discoveryName  | discSurvey  | raSex        |␣
↪decSex        | raDeg     | decDeg    | transRedshift  | specType  | discMag␣
↪ | discMagFilter  | discDate             | objectUrl                        ␣
↪             | hostName      | hostRedshift  | separationArcsec  |␣
↪separationNorthArcsec  | separationEastArcsec  |
+----------+------------+----------------+-------------+--------------+-------
↪--------+-----------+----------+----------------+-----------+----------+----
↪------------+----------------------+-----------------------------------------
↪-------+---------------+---------------+------------------+----------------
↪--------+-----------------------+
| 2016asf  | SN2016asf  | ASASSN-16cs    | ASAS-SN     | 06:50:36.73  |␣
↪+31:06:45.36  | 102.6530  | 31.1126  | 0.021          | SN Ia     | 17.1    ␣
↪ | V-Johnson      | 2016-03-06 08:09:36  | https://www.wis-tns.org/object/
↪2016asf  | KUG 0647+311  |               | 0.66             | 0.65         ␣
↪          | -0.13                |
+----------+------------+----------------+-------------+--------------+-------
↪--------+-----------+----------+----------------+-----------+----------+----
↪------------+----------------------+-----------------------------------------
↪-------+---------------+---------------+------------------+----------------
↪--------+-----------------------+
```

You can save the results to file by passing in a directory path within which to save the files to. The four flavours of data (sources, photometry, spectra and files) are saved to separate files but all data can be assoicated with its transient source using the transient's unique `TNSId`.

```
tns.table("~/tns")
```




```
:width: 800px
:alt: ascii files
```

yaml(*dirPath=None*)

Render the results in yaml format

Key Arguments

- `dirPath` – the path to the directory to save the rendered results to. Default *None*

Return

- `yamlSources` – the top-level transient data
- `yamlPhot` – all photometry associated with the transients
- `yamlSpec` – all spectral data associated with the transients
- `yamlFiles` – all files associated with the matched transients found on the tns

Usage

To render the results in yaml format:

```
yamlSources, yamlPhot, yamlSpec, yamlFiles  = tns.yaml()
print(yamlSources)
```

```
- TNSId: 2016asf
  TNSName: SN2016asf
  decDeg: 31.1126
  decSex: '+31:06:45.36'
  discDate: '2016-03-06 08:09:36'
  discMag: '17.1'
  discMagFilter: V-Johnson
  discSurvey: ASAS-SN
  discoveryName: ASASSN-16cs
  hostName: KUG 0647+311
  hostRedshift: null
  objectUrl: https://www.wis-tns.org/object/2016asf
  raDeg: 102.65304166666667
  raSex: '06:50:36.73'
  separationArcsec: '0.66'
  separationEastArcsec: '-0.13'
  separationNorthArcsec: '0.65'
```

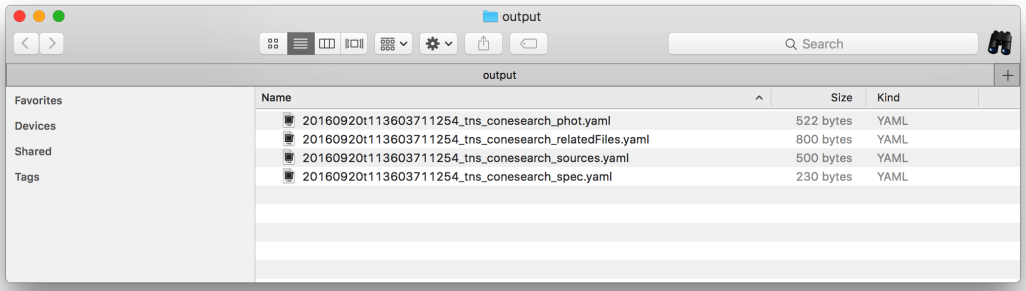
(continues on next page)

(continued from previous page)

```
  specType: SN Ia
  transRedshift: '0.021'
```

You can save the results to file by passing in a directory path within which to save the files to. The four flavours of data (sources, photometry, spectra and files) are saved to separate files but all data can be assoicated with its transient source using the transient's unique `TNSId`.

```
tns.yaml("~/tns")
```




```
:width: 800px
:alt: yaml output
```

property files

The associated source files

Usage

```
sourceFiles = tns.files
```

property photometry

The associated source photometry

Usage

```
sourcePhotometry = tns.photometry
```

property sources

The results of the search returned as a python list of dictionaries

Usage

```
sources = tns.sources
```

property spectra

The associated source spectral data

Usage

```
sourceSpectra = tns.spectra
```

property url

The generated URL used for searching of the TNS

Usage

```
searchURL = tns.url
```

3.3 Functions

transientNamer.commonutils.getpackagepath	*getpackagepath*

3.3.1 getpackagepath (*function*)

getpackagepath()
getpackagepath

3.4 A-Z Index

Modules

transientNamer.commonutils	*common tools used throughout package*
transientNamer.utKit	*Unit testing tools*

Classes

transientNamer.astronotes	*Tools to download, parse and ingest astronote content into a MySQL database*
transientNamer.search	*Search the Transient Name Server with various search constraints*

Functions

transientNamer.commonutils.getpackagepath	*getpackagepath*

CHAPTER

FOUR

RELEASE NOTES

v0.4.6 - July 3, 2023

- **FIXED**: small fixes to unit-tests

v0.4.5 - January 10, 2023

- **FIXED**: changes to search url to reflectv changes made on the TNS. All transients should now be reported again.

v0.4.4 - June 17, 2022

- **FIXED**: unit-tests failing due to throttling from TNS

v0.4.3 - January 25, 2021

- **FIXED**: call to settings file now explicit from command-line (was not required until tns-marker added)

v0.4.2 - January 25, 2021

- **REFACTOR:** requests will fail if TNS responds with any status other than 200
- **REFACTOR:** added user-agent header now required by TNS (needs added in the settings file)

v0.4.1 - January 25, 2021

- **FEATURE**:** you can now download and parse astronotes (including ability to add to MySQL database tables)
- **REFACTOR:** Updated URL to the new TNS URL https://www.wis-tns.org

v0.3.2 - December 21, 2020

- **ENHANCEMENT**: now using discovered_period_value for discInLastDays. More efficient than adding range of obsdate.

v0.3.1 - October 25, 2020

- **FIXED:** bytes to string with utf-8 encoding bug stopping regexes from passing

v0.3.0 - May 22, 2020

- Now compatible with Python 3.*

PYTHON MODULE INDEX

INDEX