# pynetworktables2js Documentation

***Release 2020.0.1***

**RobotPy development team**

**Jul 13, 2020**

# Contents

A cross platform library that forwards NetworkTables key/values over a WebSocket, so that you can easily write a Driver Station Dashboard for your robot in HTML5 + JavaScript.

This library does not provide a full dashboard solution, but is intended to provide the necessary plumbing for one to create one with only knowledge of HTML/JavaScript. Because the communications layer uses NetworkTables, you can connect to all FRC languages (C++, Java, LabVIEW, Python).

---

**Note:** NetworkTables is a protocol used for robot communication in the FIRST Robotics Competition, and can be used to talk to Shuffleboard/SmartDashboard. It does not have any security, and should never be used on untrusted networks.

---

CHAPTER 1

# Documentation

Documentation can be found at http://pynetworktables2js.readthedocs.org/

CHAPTER 2

# Installation

## 2.1 Easy install (Windows only)

1. Download the latest pynetworktables2js.exe from GitHub at https://github.com/robotpy/pynetworktables2js/releases .
2. Extract the exe from the zipfile, and copy it to your directory of HTML/JS files.
3. Double click the exe to run it!

**Note:** By default, it will connect to 127.0.0.1. To connect to a robot, you will need to pass the exe arguments to tell it where the robot is. Use `--help` to see the available options.

## 2.2 Manual install

Make sure to install python 3 on your computer, and on Windows you can execute:

```
py -3 -m pip install pynetworktables2js
```

On Linux/OSX you can execute:

```
pip install pynetworktables2js
```

**Note:** Technically, there's nothing stopping you from installing this on your robot, as there is a python interpreter available on the roboRIO (RobotPy). However, due to FRC bandwidth limitations, it's probably best to run the UI + server on your driver station laptop.

# CHAPTER 3

## Why make an HTML/Javascript dashboard?

**TL;DR**: It's simpler.

pynetworktables2js lowers the barrier of entry for teams that want an additional way to tune/control their robot with a minimal amount of programming.

Lots of students and mentors know how to create simple web pages to display content, and there's lots of resources out there for creating dynamic content for webpages that use javascript. There is a lot of visually appealing content that others have created using web technologies – why not leverage those resources to make something cool to control your robot?

CHAPTER 4

# Usage

You can just distribute your HTML files, and run a pynetworktables server using the following command from inside the directory:

```
python3 -m pynetworktables2js
```

Or on Windows:

```
py -3 -m pynetworktables2js
```

This will start a pynetworktables2js server using Tornado (which is installed by default) and it will serve the current directory. You can navigate your browser (I recommend Chrome) to http://127.0.0.1:8888 and see your website.

You will want to also pass either the `--robot` or `--team` switch:

```
py -3 -m pynetworktables2js --robot roborio-XXXX-frc.local
py -3 -m pynetworktables2js --team XXXX
```

Dashboard mode currently doesn't work, as the underlying support in pynetworktables hasn't been implemented yet for the newer FRC Driver Station.

## 4.1 Customized python server

There are two example servers distributed with pynetworktables2js, one that uses tornado, and one that uses aiohttp. Either one should work.

Go to the 'example' directory distributed with pynetworktables2js, and run:

```
python3 tornado_server.py --robot 127.0.0.1
```

If you want to try this out with your current robot, you can do:

```
python3 tornado_server.py --robot roborio-XXX.local
```

If you are running pynetworktables2js on your driver station laptop, you can receive robot IP information directly from the Driver Station (handy during actual competitions):

```
python3 tornado_server.py --dashboard
```

If you navigate your browser (I recommend Chrome) to http://127.0.0.1:8888, all of the current NetworkTables values will be shown as they change.

One way of testing this out is use FIRST's TableViewer application (you can launch it using the "Outline Viewer" WPILib menu item in Eclipse), and start it in server mode.

Feel free to copy the example directory to create your own customized dashboard. Just add your custom files to the www directory.

# CHAPTER 5

## Contributing new changes

pynetworktables2js is intended to be a project that all members of the *FIRST* community can quickly and easily contribute to. If you find a bug, or have an idea that you think others can use:

1. Fork this git repository to your GitHub account
2. Create your feature branch (`git checkout -b my-new-feature`)
3. Commit your changes (`git commit -am 'Add some feature'`)
4. Push to the branch (`git push -u origin my-new-feature`)
5. Create new Pull Request on GitHub

One place in particular I would love to see contributions is in adding useful JavaScript functions/objects that make creating dashboards even easier!

# CHAPTER 6

# Authors

Leon Tan of FRC Team 1418 did the initial research/work to get this working, and created an initial working prototype for Team 1418's 2015 Dashboard, which was instrumental to winning an Innovation In Control award at the 2015 Greater DC Regional.

Dustin Spicuzza cleaned stuff up, rewrote things, added more functionality, wrote documentation, and packaged it so other teams could use it.

## 6.1 JS API: NetworkTables

To use these functions, add this to your HTML page:

```
<script src="/networktables/networktables.js"></script>
```

**Note**:

It's very important to note that the Javascript NetworkTables API currently has no concept of a table or subtable. When referring to keys when accessing the API you must use absolute paths, and not just key names. For example, if you use `SmartDashboard.putNumber('foo', 1)` to put a value called `foo`, then to access the value using the Javascript API you would use `NetworkTables.getValue('/SmartDashboard/foo')`.

### 6.1.1 Listeners

These functions allow your code to listen for particular NetworkTables events.

`NetworkTables.`**`addWsConnectionListener`**(*f*[, *immediateNotify*])

Sets a function to be called when the websocket connects/disconnects

**Arguments**

- **`f`** – a function that will be called with a single boolean parameter that indicates whether the websocket is connected

- **immediateNotify** – If true, the function will be immediately called with the current status of the websocket

Example usage:

```
NetworkTables.addWsConnectionListener(function(connected){
    console.log("Websocket connected: " + connected);
}, true);
```

NetworkTables.**addRobotConnectionListener**(*f*[, *immediateNotify*])

Sets a function to be called when the robot connects/disconnects to the pynetworktables2js server via NetworkTables. It will also be called when the websocket connects/disconnects.

When a listener function is called with a 'true' parameter, the NetworkTables.getRobotAddress() function will return a non-null value.

**Arguments**

- **f** – a function that will be called with a single boolean parameter that indicates whether the robot is connected
- **immediateNotify** – If true, the function will be immediately called with the current robot connection state

Example usage:

```
NetworkTables.addRobotConnectionListener(function(connected){
    console.log("Robot connected: " + connected);
}, true);
```

NetworkTables.**addGlobalListener**(*f*[, *immediateNotify*])

Set a function that will be called whenever any NetworkTables value is changed

**Arguments**

- **f** – When any key changes, this function will be called with the following parameters; key: key name for entry, value: value of entry, isNew: If true, the entry has just been created
- **immediateNotify** – If true, the function will be immediately called with the current value of all keys

Example usage:

```
NetworkTables.addGlobalListener(function(key, value, isNew){
    // do something with the values as they change
}, true);
```

NetworkTables.**addKeyListener**(*key*, *f*[, *immediateNotify*])

Set a function that will be called whenever a value for a particular key is changed in NetworkTables

**Arguments**

- **key** – A networktables key to listen for
- **f** – When the key changes, this function will be called with the following parameters; key: key name for entry, value: value of entry, isNew: If true, the entry has just been created
- **immediateNotify** – If true, the function will be immediately called with the current value of the specified key

Example usage:

```
NetworkTables.addKeyListener(function(key, value, isNew){
    // do something with the values as they change
}, true);
```

### 6.1.2 NetworkTables Interface

NetworkTables.**containsKey**(*key*)

Use this to test whether a value is present in the table or not

**Arguments**

- **key** – A networktables key

**Returns** true if a key is present in NetworkTables, false otherwise

**Warning:** This may not return correct results when the websocket is not connected

NetworkTables.**getKeys**()

**Returns** all the keys in the NetworkTables

**Warning:** This may not return correct results when the websocket is not connected

NetworkTables.**getValue**(*key*[, *defaultValue* ])

Returns the value that the key maps to. If the websocket is not open, this will always return the default value specified.

**Arguments**

- **key** – A networktables key
- **defaultValue** – If the key isn't present in the table, return this instead

**Returns** value of key if present, `undefined` or `defaultValue` otherwise

**Warning:** This may not return correct results when the websocket is not connected

---

**Note:** To make a fully dynamic webpage that updates when the robot updates values, it is recommended (and simpler) to use `addKeyListener()` or `addGlobalListener()` to listen for changes to values, instead of using this function.

---

NetworkTables.**getRobotAddress**()

**Returns** null if the robot is not connected, or a string otherwise

NetworkTables.**isRobotConnected**()

**Returns** true if the robot is connected

NetworkTables.**isWsConnected**()

**Returns** true if the websocket is connected

NetworkTables.**putValue**(*key*)

Sets the value in NetworkTables. If the websocket is not connected, the value will be discarded.

**Arguments**

- **key** – A networktables key
- **value** – The value to set (see warnings)

**Returns** True if the websocket is open, False otherwise

---

**Note:** When you put a value, it will not be immediately available from `getValue`. The value must be sent to the NetworkTables server first, which will then send the change notification back up to the javascript NetworkTables key/value cache.

---

**Warning:** NetworkTables is type sensitive, whereas Javascript is loosely typed. This function will **not** check the type of the value that you are trying to put, so you must be careful to only put the correct values that are expected. If your robot tries to retrieve the value and it is an unexpected type, an exception will be thrown and your robot may crash. Make sure you test your code – you have been warned.

### 6.1.3 Utility functions

NetworkTables.**create_map**()

Creates a new empty map (or hashtable) object and returns it. The map is safe to store NetworkTables keys in.

**Returns** map object, with forEach/get/has/set functions defined. Simlar to a map object when using d3.js

NetworkTables.**keyToId**(*key*)

Escapes NetworkTables keys so that they're valid HTML identifiers.

**Arguments**

- **key** – A networktables key

**Returns** Escaped value

NetworkTables.**keySelector**(*key*)

Escapes special characters and returns a valid jQuery selector. Useful as NetworkTables does not really put any limits on what keys can be used.

**Arguments**

- **key** – A networktables key

**Returns** Escaped value

For example, to set the text of an element which has an id that corresponds to a value in NetworkTables:

```
$('#' + NetworkTables.keySelector(key)).text(value);
```

## 6.2 JS API: JQuery Extensions

```
<script src="/networktables/jquery_ext.js"></script>
```

**Note:** These functions require jQuery to be loaded first!

`$.nt_toggle` (*key*, *function*)

When a networktables variable changes, a checkbox element will be updated when a NT variable changes and when a user clicks it.

Alternatively, you can use this with custom elements, by providing a function that will be called only when the NT value is changed. The NT value will be toggled when the user clicks the selected element(s).

**Arguments**

- **k** – Networktables key
- **fn** – (optional) function that accepts a single param, will be called on change
- **evt** – (optional) Which event to toggle the value on (defaults to 'click')

Example usage:

```
// this works on a checkbox
$('#my_checkbox').nt_toggle('/SmartDashboard/some_boolean');

// or on a clickable element
$('#my_clickable').nt_toggle('/SmartDashboard/b', function(v) {
        this.css('background-color', v ? 'green' : 'gray');
});
```

## 6.3 JS API: Utilities

To use these functions, add this to your HTML page:

```
<script src="/networktables/utils.js"></script>
```

**Note:** These functions require jQuery and D3 to be loaded first!

The functions in this file are still experimental in nature, and as we expand the number of functions in this file it is expected that the API will change.

### 6.3.1 SendableChooser

**attachSelectToSendableChooser** (*html_id*, *nt_key*)

Given the id of an HTML `<select>` element and the key name of a SendableChooser object setup in networktables, this will sync the select combo box with the contents of the SendableChooser, and you will be able to select an object using the select element.

**Arguments**

- **html_id** – An ID of an HTML select element
- **nt_key** – The name of the NetworkTables key that the SendableChooser is associated with

See the WPILib documentation for information on how to use SendableChooser in your robot's program.

**updateSelectWithChooser**(*html_id*, *nt_key*)

This function is designed to be used from the onValueChanged callback whenever values from a SendableChooser change, but you probably should prefer to use attachSelectToSendableChooser instead.

See attachSelectToSendableChooser documentation.

### 6.3.2 Indicators

**attachRobotConnectionIndicator**(*html_id*[, *size*, *stroke_width*])

Creates a circle SVG that turns red when robot is not connected, green when it is connected.

**Arguments**

- **html_id** – ID to insert svg into
- **size** – Size of circle
- **stroke_width** – Border of circle

## 6.4 JS API: Camera Integration

```
<script src="/networktables/camera.js"></script>
```

**Note:** These functions require jQuery to be loaded first!

**loadCameraOnConnect**(*args*)

This useful helper function will create an img or svg element inside of the div element that you specify. The image will only be connected when a successful NetworkTables connection is detected, to prevent timeout issues. Additionally, this function will attempt to verify that the webcam server is actually up and running before creating the image.

You should provide an object with an object that can have the following attributes:

**Arguments**

- **container** – Where to draw things
- **proto** – optional, defaults to http://
- **host** – optional, if null will use robot's autodetected IP
- **port** – optional, webserver port
- **image_url** – path to mjpg stream
- **data_url** – a file or page that must exist on the webcam server
- **wait_img** – optional image to show when not connected
- **error_img** – optional image to show when not connected
- **attrs** – optional attributes to set on svg or img element
- **nosim** – if true, connect to the webcam in simulation mode

For example, to connect to mjpg-streamer on the RoboRIO:

```
loadCameraOnConnect({
    container: '#my_div_element',
    port: 5800,
    image_url: '/?action=stream',
    data_url: '/program.json',
    attrs: {
        width: 640,
        height: 480
    }
});
```

**Note:** This has only been tested with mjpg-streamer, but should work for other HTTP webcams as well.

## 6.5 Troubleshooting

Because pynetworktables2js uses pynetworktables, if you're having problems getting pynetworktables2js working, you may find the pynetworktables troubleshooting page to be a useful reference.

CHAPTER 7

# Indices and tables

- genindex
- modindex
- search

# Index

## Symbols



## A



## L



## N



## U