

SignalKInputStreamerUsage

November 16, 2019

1 Signal K input streamer usage

Those who are interested in how things works may want read some theory from the *Signal K input streamer design description* document ([ipynb](#) | [html](#) | [pdf](#)).

If not, be still reminded that the Signal K input streamer of *DashT* plug-in is not needed to use OpenCPN with a Signal K server together as explained below.

1.1 Connecting Signal K node server and OpenCPN instances

There are many advantages to use a [Signal K node server](#) with OpenCPN. It enables your networked boat instrumentation ecosystem where OpenCPN is an important player but not the only one. With Signal K node server the various software and hardware units gets all the same data, at the same time thanks to its powerful interconnection capabilities.

Since the *DashT Signal K input streamer* does **not** feed and will never feed data to OpenCPN it is necessary to provide OpenCPN with its own Signal K connection:

If you missed the theory part, be aware that not all NMEA sentences will be served by Signal K input streamer but only those available in the delta channel of the Signal K node server. For example, AIS data is not. Even without AIS, OpenCPN needs its own GPS data. So, before setting the by-pass one needs to feed the OpenCPN so that it works also without the Signal K input streamer of the *DashT* plug-in.

In this document we expect that the Signal K server node's input is from USB as in the use case referred in above documents. Otherwise, it does not matter what are its inputs. We have set Signal K's AIS-to-NMEA forwarder plug-in output to 12000 (for no particular reason).

If you are not familiar with Signal K, please follow the above link, install a server and play around with it and its incorporated instruments. Once get your data visible using your browser, you will be familiar at this point with Signal K Server administration, which can be accessed typically through this URL on the system where the Signal K server node is running (or on a remote server on another computer using its IP-address or its network name):

`http://localhost:3000/admin/#/dashboard`

1.1.1 Classical TCP settings

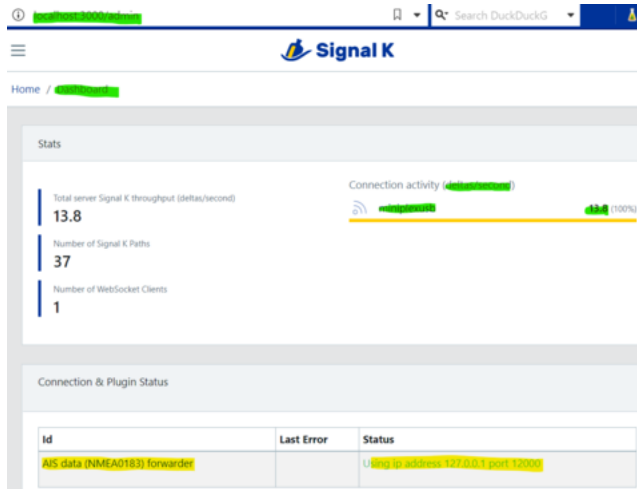
These are explained better in various Signal K node server documents. They results to following views, first on the Signal K server node's server administration side...

| | | |
|-------------------|---|------|
| Input Type | NMEA0183 | |
| Enabled | <input checked="" type="checkbox"/> YES | |
| Logging | <input type="checkbox"/> NO | |
| ID | miniplexusb | |
| NMEA 0183 Source | Serial | |
| Serial port | COM4 | COM4 |
| Baud Rate | 115200 | |
| | Example: 4800 | |
| sentenceEvent | MiniPlexUSB | |
| | Example: nmea1data | |
| Validate Checksum | <input checked="" type="checkbox"/> YES | |

For the AIS, it is a plug-in, which here we set to port 12000, for no particular reason other than to have something easy to remember.

The image shows the Signal K web interface. On the left is a dark sidebar menu with the following items: Dashboard, Webapps, Appstore, Server, Settings, Vessel data, Plugin Config (circled in blue), Connections, Data log files, Update, and Security. The main content area is titled 'Home' and shows the configuration for the 'AIS data (NMEA0183) forwarder'. The status is 'Using ip address 127.0.0.1 port 12000'. There are checkboxes for 'Active' (checked), 'Enable Logging' (unchecked), 'Forward AIVDO sentences (own vessel)' (unchecked), and 'Forward AIVDM sentences (other vessels)' (checked). The 'IP Address' field contains '127.0.0.1' and the 'Port' field contains '12000' (highlighted in green). A blue 'Submit' button is at the bottom.

| | |
|----------------|---|
| Dashboard | Home |
| Webapps | ▼ AIS data (NMEA0183) forwarder |
| Appstore | Status: Using ip address 127.0.0.1 port 12000 |
| Server | <input checked="" type="checkbox"/> Active |
| Settings | <input type="checkbox"/> Enable Logging |
| Vessel data | IP Address |
| Plugin Config | 127.0.0.1 |
| Connections | Port |
| Data log files | 12000 |
| Update | <input type="checkbox"/> Forward AIVDO sentences (own vessel) |
| Security | <input checked="" type="checkbox"/> Forward AIVDM sentences (other vessels) |
| | Submit |



... and on the OpenCPN communication settings side:

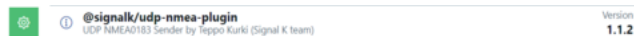
| Data Connections | | | | | |
|--|---------|-----------|----------|-----------------|--------------|
| <input checked="" type="checkbox"/> Enable | Type | Direction | Protocol | Network Address | Network Port |
| | Network | Input | TCP | 127.0.0.1 | 10110 |
| Comment: Signal K out | | | | | |
| <input checked="" type="checkbox"/> Enable | Type | Direction | Protocol | Network Address | Network Port |
| | Network | Input | TCP | 127.0.0.1 | 12000 |
| Comment: Signal K AIS forwarder | | | | | |

1.1.2 UDP broadcasting method

This is the alternative method for those having, for example multiple OpenCPN or other software clients requesting NMEA sentences on different computers. For example, one can have one OpenCPN in the Raspberry Pi or other Linux box on which the Signal K node server is running. But one may want to have an other OpenCPN instance on a laptop or tablet.

With UDP broadcasting Signal K node server becomes the broadcaster of the all data it receives while the listeners will be... well, listening! Very efficient but with a possibility of a data loss. But for all practical terms, with these data rates that's not an issue.

You need to install @signalk/udp-nmea-plugin, if not yet installed,



Configuration: provide your network's address as base to broadcast to. 255 is the broadcast address. Line delimiter is optional, if you want to debug it simplifies your life. You can debug now, with (Linux) command `nc -ulkw 0 192.168.8.255 2000` where you would replace 192.168.8 with your network IP-address. Typically, in a local system like Raspberry Pi, you would use the local network broadcast address 127.0.0.255 (it goes well also outside to all other systems in Raspberry Pi's network, no worries).

▼ UDP NMEA0183 Sender

Status: Using address 192.168.8.255

☒ Active

☐ Enable Logging

IP Address (overrides broadcast address if entered)

broadcastAddress

192.168.8.255

Port *network*

2000

☒ Use server event nmea0183

☒ Use server event nmea0183out

Line delimiter *optional*

LF

On your local instance of OpenCPN (*i.e.* on the same machine where the Signal K node server is running) you can use broadcast client address 127.0.0.255 - in case you change networks this address remains unchanged. You can also use 0.0.0.255 which works as well, but both needs the port number to be same as above.

| Enable | Type | Direction | Protocol | Network Address | Network Port |
|-------------------------------------|---------|-----------|----------|-----------------|--------------|
| <input checked="" type="checkbox"/> | Network | Input | UDP | 127.0.0.255 | 2000 |
| Comment: Signal K UDP NMEA plug-in | | | | | |

Elsewhere in the network you would need to provide the network's address.

| Type | Direction | Protocol | Network Address | Network Port | Priority |
|--|-----------|----------|-----------------|--------------|----------|
| <input checked="" type="checkbox"/> Enable | Input | UDP | 192.168.8.255 | 2000 | 1 |
| Comment: Signal K UDP Nmea Plug-in | | | | | |

Your all set!

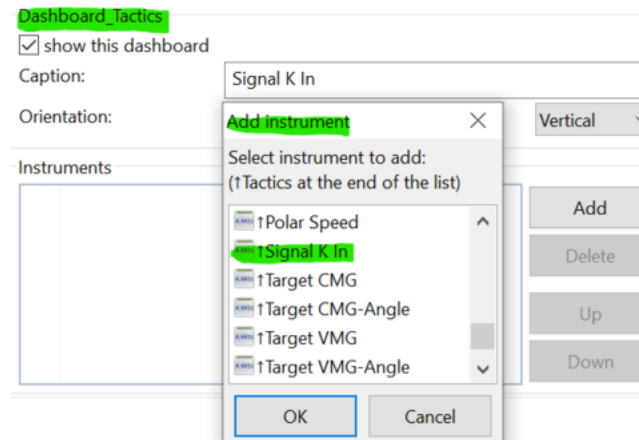
Actually, everything works as before and so smoothly that why would you continue from here?

Since now we are going to by-pass all that with *DashT* plug-in's Signal K input streamer!

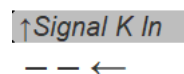
1.2 Enabling and configuring Signal K input streamer

Start your OpenCPN v5 with *DashT*.

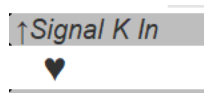
Open the Preferences dialog and create a new instrument panel with a single instrument in it: *Signal K In*:



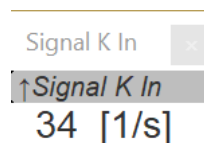
The Signal K In instrument is showing the state of the connection, normally attempting to connect to Signal K node server's delta format output channel, indicated with the arrow moving from right to left:



Once the connection is made, there is an intermediate illustration (which normally does no last for long) indicating that the connection has been made and heartbeat is found. Data is already flowing in.



Within approximately five seconds the display starts to indicate the Signal K input streamer's throughput in OpenCPN Dashboard sentences parsed in second.



Please note that this value is approximately twice the value the Signal K node server indicates. This is because one delta sentence it puts out contains usually more than one OpenCPN Dashboard's internal sentences. For example, latitude and longitude is one data set in Signal K while in OpenCPN's Dashboard type of instruments we are using it is actually two, separate data.

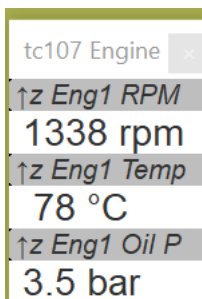
1.3 Using data coming in Signal K format but unknown to OpenCPN

The above data throughput counter indicates the amount of data received from the Signal K node server's delta channel. Every data set is parsed and effectively passed to OpenCPN and its Dashboard instruments and Tactics plug-in performance functions.

But not all parsed data is necessarily used: the input data stream in Signal K delta format can contain much more information than Dashboard and Tactics are aware of. For example, if you have some extra boards on your Raspberry Pi or your boat's instrumentation bus is **NMEA-2000** instead of NMEA-0183, you will get information which you can visualize with the Signal K's ecosystem.

Nothing prevents *DashT* to make use of the extra data and provide enhanced instruments for energy and for engine monitoring.

The current version of *DashT* makes a proof of concept for this by providing three instruments to monitor the port side engine, *i.e.* the main engine as it appears in NMEA-2000 originated data.



The above use case illustrates well the achieved goal: While the main motivation for the *DashT* plug-in's Signal K by-passing input streamer is in the overall system performance, the secondary requirement to be NMEA-2000 compatible has also been met, opening possibilities for enhancement requests in the future versions - albeit the Tactics part of Dashboard-Tactics plug-in is made for *voileux* this connector is available for everybody. Sharpen your C++ pencils and jump on-board, Dashboard's dial gauge classes are waiting for you and your engine data!

This would, hopefully lead to a development of *DashE* where E stands for Energy or Engine, according your inspiration.

1.4 Debugging

1.4.1 Configuration file and its usage in debugging

The default configuration file is in JSON-format (like Signal K data). The default values are good for normal, local-only operation. It may require changes in case when things are not working. It is located:

Windows \ProgramData\opencpn\plugins\dashboard_tactics_pi\streamin-sk.json

Linux ~/.opencpnplugins/dashboard_tactics_pi/streamin-sk.json

The configuration file is read only in startup so you would need to restart OpenCPN after modifying this file.

When debugging you would set the verbosity parameter to a value between 2... 5, the 5 being really verbose; so talkative that it would actually affect the performance and the OpenCPN log file will get really big, really fast.

Typically, you would stop OpenCPN, set the debug value and, before starting OpenCPN you would set a line-by line observation to its log file. With `grep` command you can further reduce the filtering if it is too verbose.

On Windows using PowerShell:

```
PS C:\ProgramData\opencpn>
Get-Content ./opencpn.log -Wait -Tail 20
```

On *nix systems:

```
tail -f ~/.opencpn/opencpn.log
or with filtering
tail -f ~/.opencpn/opencpn.log | grep dashboard_tactics_pi
```

1.4.2 No connection gets established

If the arrows keep on moving for ever from right to left, it indicates that the TCP/IP cannot get connection. Check the configuration file's parameter, which is by default:

```
"source"          : "localhost:8375", // not limited to localhost
```

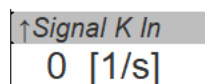
If your Signal K node server is located in another computer, replace the `localhost` with the computer's name or if that does not work, with its IP-address (numerical).

if the remote Signal K server is still not answering it may be that it does not implement the delta service in the port 8375, or in any other port perhaps; it is not a mandatory requirement for a Signal K server to provide this service.

If you know that the local Signal K node server is there, that it is based on Node.js and it is equal or greater to version 1.17, there is indeed no reason why the port 8375 would not be served. In this case you may try simply to use IP-address `127.0.0.1` instead of `localhost`, maybe there is an issue with your systems' Domain Name Service (DNS).

1.4.3 Connection gets established by no data is coming in

This is indicated by the following condition in the throughput display:



The image shows a small rectangular box with a grey header containing the text '↑Signal K In'. Below the header, the box displays the value '0 [1/s]' in a blue font.

First, see the Signal K node server's dashboard and verify that it gets indeed some data in - if not, nothing will come out, of course.

If all looks good both for Signal K node server's input and also the Dashboard's instruments keep hopping around as they normally do, there is simply no data in 8375 port. See above, that's not a mandatory requirement for a Signal K server, maybe you are using some commercial implementation of it?

1.4.4 The Signal K Stream In indicates HALT state

↑Signal K In
– HALT –

The message indicates that the continuously running communication thread has been stopped. There is no other remedy for this condition but to stop gracefully OpenCPN and restart it.

To avoid this to happen one should keep both the Signal K input stream *instrument* and the Influx DB output stream *instrument* both in their own, distinct instrument windows. In other words, they shall be separated from other instruments but also from each other. This is to avoid that the communication thread would get orphan when instrument windows get reorganized.