SignalKInputStreamerUsage

November 5, 2019

1 Signal K input streamer usage

You may want to first 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 Dashboard-Tactics plug-in is not needed to use OpenCPN with a Signal K server together:

1.1 Connecting Signal K node server and OpenCPN instances

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. 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 Dashboard Tactics 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 other than aesthetic).

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	**YES**
Logging	NO
ID	miniplexusb
NMEA 0183 Source	Serial ▼
Serial port	COM4 ▼ COM4
Baud Rate	115200
	Example: 4800
sentenceEvent	MiniPlexUSB
	Example: nmea1data
Validate Checksum	**YES**

Signal K ≡

- Dashboard
- Webapps
- Appstore
- Server
  - Settings
  - Vessel data
  - Plugin Config
  - Connections
  - Data log files
  - Update
- Security

Home

∨ AIS data (NMEA0183) forwarder

Status: Using ip address 127.0.0.1 port 12000
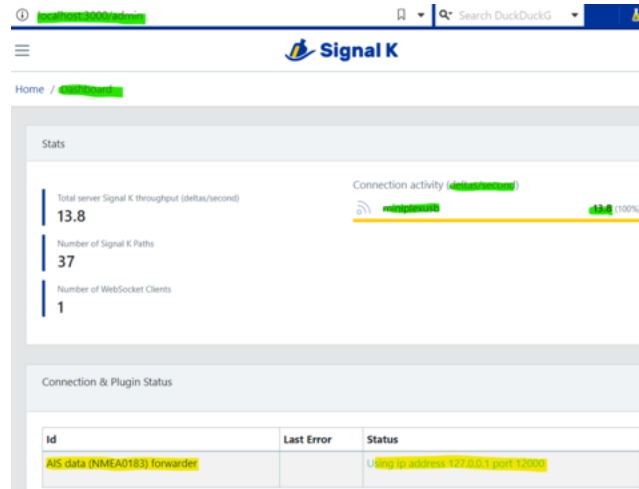
☑ Active

☐ Enable Logging

IP Address

127.0.0.1

Port

12000.

☐ Forward AIVDO sentences (own vessel)

☑ Forward AIVDM sentences (other vessels)

Submit

... and on the OpenCPN communication settings side:

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 diffent computers. Signal K node servers becomes the broadcaster of those and asynchronous 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.

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.

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


Your all set!

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

That question you need to ask from yourself (or read the theory), since now we are going to by-pass all that with Dashboard-Tactics plug-in's Signal K input streamer!

1.2   Enabling and configuring Signal K input streamer

Start your OpenCPN v5 with Dashboard-Tactics.

Open the Preferences dialog and create a new instrument panel with a single instrument in it:
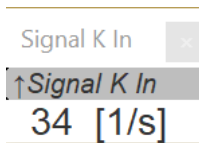*Signal K In*:
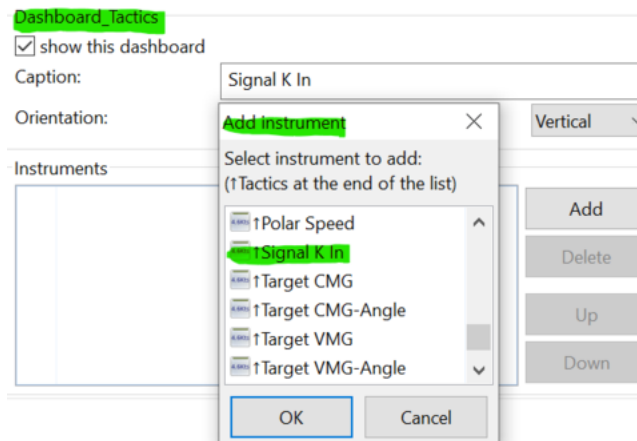
The Signal K In instrument is showing the state of the connection, normally attempting to connect to Signal K node server's delta format output channel, indicated with the arrow moving from right to left:
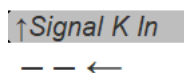
Once the connection is made, there is an intermediate illustration (which normally does no last for long) indicating that the connection has been made and heartbeat is found. Data is already flowing in.
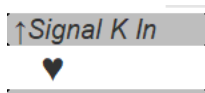
Within approximately five seconds the display starts to indicate the Signal K input streamer's throughput in OpenCPN Dashboard sentences parsed in second.

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`.

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

The above data throughput counter indicates only the data which is parsed and effectively passed to OpenCPN and its Dashboard plug-in. But not all of it is necessarily used: the input data stream

in Signal K delta format can contain much more information, depending of your boat's instrumentation bus (perhaps **NMEA-2000**?). To make a proof of concept for this, the Dashboard-Tactics plug-in provides three instruments to monitor the port side engine, *i.e.* the main engine.
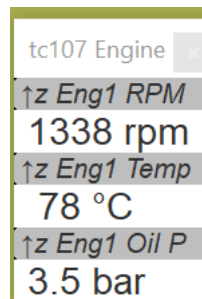
This illustrates that while the main motivation for the Dashboard-Tactics plug-in's Signal K bypassing input streamer is in the overall system performance, the secondary requirement to be NMEA-2000 compatible has also been met, opening possibilities for enhancement requests in the future versions - albeit the Tactics part of Dashboard-Tactis 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!

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\dashoard_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

6

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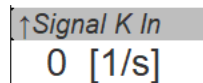
If your Signal K node server is located in another computer, replace the `localhost` with the computer's name or if that does not work, with its IP-address (numerical).

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 of 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 `loccalhost`, maybe there is an issue with your systems' Domain Name Service (DNS).

1.4.3   Connection gets established by no data is coming in

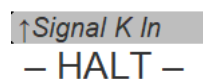This is indicated by the following condition in the throughput display:

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

The message indicates that the continously 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.