Table of Contents

Introduction

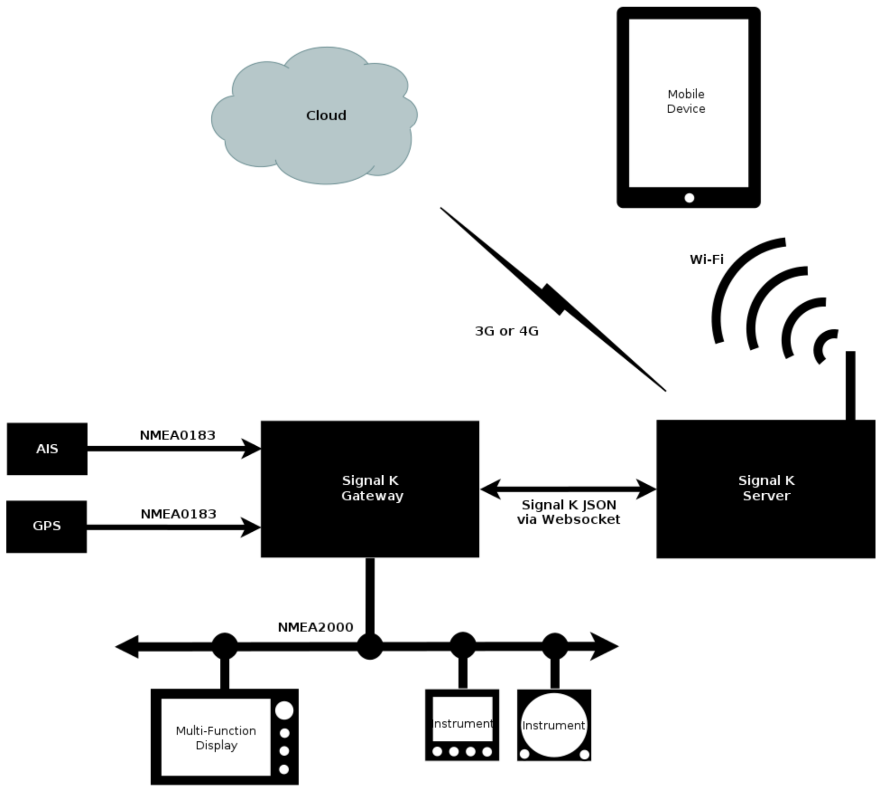
This is the documentation for the Signal K Specification 1.5.0 version, which is available in the following formats;

- html (this document) 1.5.0
- pdf
- epub
- mobi

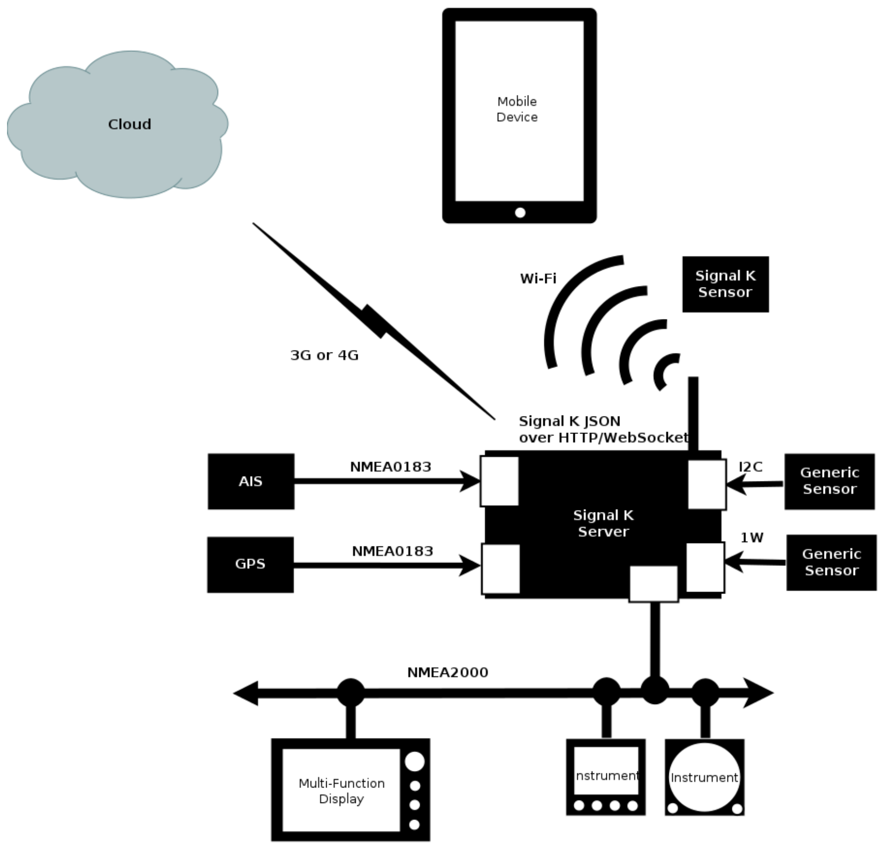
What is Signal K?

Signal K is a modern and open data format for marine use. It is an Internet friendly standard built on common web technologies, such as JSON and WebSockets. Signal K is Free and Open Source software. This document is licensed under the Creative Commons CC-BY-SA license. All Signal K source code is licensed under the Apache License, Version 2.0. Signal K is developed in the open with help from the marine community. Your ideas and feedback are valuable and welcome.

Signal K is designed to work in harmony with a boat’s existing navigation equipment which may use NMEA0183, NMEA2000 or proprietary data protocols, converting and enhancing this information into a modern “web friendly” format which can be shared, processed and displayed on the latest web apps, mobile devices and cloud servers. A typical NMEA based installation consists of an NMEA to Signal K gateway and an optional Signal K server. The gateway translates NMEA data into Signal K format and the server can host additional functions like logging, cloud integration and data analysis.



One major advantage of Signal K is the ability to represent data from heterogeneous sources. In addition to traditional NMEA sources, data from generic sensors as well as modern Signal K enabled sensors can be fused into a single data model and a single protocol for accessing the data. Another typical configuration is a Signal K server with adapters and converters for the different sources.



Signal K Data Model (A.K.A. The Schema)

The Signal K Data Model or schema defines a universal model for marine related information and it is specified as a JSON schema. See the Signal K Data Model section for details.

In traditional marine standards there are many tightly defined messages, each with a specific purpose, but there is no data model to relate them. Furthermore, any device which needs to decode those messages must have a copy of the data dictionary in order to do so. By defining a data model in JSON we can make the messaging layer simpler and easily extensible. We define consistent units and meta data for each data point in the model. This means that a specific data point (e.g. COG) will always be found at a predictable address.

It also means that a display device such as a chartplotter implementing Signal K does not need to know about the data model beforehand. It can query the central Signal K server on the boat to get all the information it needs to display any data point. This **metadata** may include information such as the unit of measure, minimum and maximum permissible values, alarm thresholds and localized display name for every data point in the model.

Signal K Message Format

Signal K defines methods for combining arbitrary data from the Data Model into valid messages. These messages are in UTF-8 JSON format.

Rather than define hundreds of specific messages, Signal K has a few common message formats which can contain any combination of data from the Data Model. This means that there is no need to define new messages for every new data point or command which needs to be supported, rather the message format remains the same and only the data transmitted needs to change. It means that any device can read any message and a device can introduce a new data point which can be understood by existing devices without the need for firmware upgrades.

Signal K Transport Layer

Signal K does not define the transport or wire protocol. Signal K messages are JSON text and can be sent over almost any physical transport layer. However, the Signal K standard does provide guidance on how to establish an initial connection, handle negotiation, subscription, and disconnection for a given transport (e.g. TCP/IP or serial).

Where possible Signal K uses well established standards like HTTPS, REST, and WebSockets. However, these should not be interpreted as required dependencies of Signal K. It is entirely possible and permissible to implement the protocol over any transport that your implementation requires.

The goal is to try to establish sensible conventions for each transport in order to make development and interconnection more predictable.

Signal K Implementations

The Signal K project has Open Source reference server implementations in Node and Java. There are also several web apps provided by the project which can be installed directly in the Node and Java servers or downloaded from Signal K GitHub. There are also commercial Signal K applications and solutions, including mobile apps available on the Apple App and Android Play Stores, as well as hardware products like iKommunicate.

Getting Started Using Signal K

You can start using Signal K by

- connecting to the demo server on the Internet with any web browser
- installing either the Node or Java server on any computer
- getting some hardware for your boat, such as a Raspberry Pi, suitable USB adapters for your boat's network (NMEA 0183, NMEA 2000 or roll your own with I2C sensors) and installing the Node or Java server
- purchasing a commercial Signal K gateway such as an iKommunicate by Digital Yacht
- installing OpenPlotter, which includes a Signal K server

Once you have a server running (or you start by using the demo server) you can install some Signal K supporting mobile apps such as

- WilhelmSK by Scott Bender (iOS)
- NMEARemote by Zapfware (iOS)
- OceanIX (Android)

Getting Started in Developing with Signal K

- Example HTML5 Applications
- Signal K JavaScript Client
- iKommunicate Developer’s Guide

Signal K Data Model

Formats

Signal K defines two data formats—full and delta—for representing and transmitting data. All Signal K data is transmitted as UTF-8 JSON.

Full Format

The full format is conceptually the simplest representation of data in Signal K. It contains all of the data from a Signal K node, which in the case of a Signal K server could be many hundreds of data points.

```
{
  "version": "1.0.0",
  "self": "urn:mrn:signalk:uuid:705f5f1a-efaf-44aa-9cb8-a0fd6305567c",
  "vessels": {
    "urn:mrn:signalk:uuid:705f5f1a-efaf-44aa-9cb8-a0fd6305567c": {
      "navigation": {
        "speedOverGround": {
          "value": 4.32693662,
          "$source": "ttyUSB0.GP",
          "sentence": "RMC",
          "timestamp": "2017-05-16T05:15:50.007Z"
        },
        "position": {
          "value": {
            "altitude": 0.0,
            "latitude": 37.81479,
            "longitude": -122.44880152
          },
          "$source": "ttyUSB0.GP",
          "sentence": "RMC",
          "timestamp": "2017-05-16T05:15:50.007Z"
        },
        "headingMagnetic": {
          "value": 5.55014702,
          "$source": "ttyUSB0.II",
          "sentence": "HDM",
          "timestamp": "2017-05-16T05:15:54.006Z"
        }
      },
      "name": "Motu",
      "uuid": "urn:mrn:signalk:uuid:705f5f1a-efaf-44aa-9cb8-a0fd6305567c"
    }
  },
  "sources": {
    "ttyUSB0": {
      "label": "ttyUSB0",
      "type": "NMEA0183",
      "GP": {
        "talker": "GP",
        "sentences": {
          "RMC": "2017-04-03T06:14:04.451Z"
        }
      },
      "II": {
        "talker": "II",
        "sentences": {
          "HDM": "2017-05-16T05:15:54.006Z"
        }
      }
    }
  }
}
```

There are several top level attributes or keys which are always present and others which are optional. The `version` key specifies which version of the Signal K specification is being used and must always present in a full Signal K model. Also always present in the full model is the `self` key. The value of `self` is the key within the `vessels` object which is the local boat. Effectively, it is a pointer into the `vessels` object.

Below the `vessels` object is a list of vessels, identified by their MMSI number or a generated unique ID. There may be many vessels if data has been received from AIS or other sources. The format for each vessel's data uses the same standard Signal K structure but may not have the same content; likely you will not have as much data about other vessels as you have about your own.

At the same level as `vessels` is `sources`. This contains a list of sources the data was obtained from. Each data object within a `vessel` may have a `$source` key which point to a source within `sources`. Several data objects may reference the same `source` since a single NMEA sentence or PGN may map to multiple keys in Signal K.

Alternatively the source data may be embedded directly in place of the `$source` by using the `source` key:

```
{
  "vessels": {
    "urn:mrn:signalk:uuid:705f5f1a-efaf-44aa-9cb8-a0fd6305567c": {
      "navigation": {
        "position": {
          "value": {
            "altitude": 0.0,
            "latitude": 37.81479,
            "longitude": -122.44880152
          },
          "source": {
            "label": "ttyUSB0",
            "type": "NMEA0183",
            "talker": "GP",
            "sentence": "PRMC"
          },
          "timestamp": "2017-05-16T05:15:50.007Z"
        }
      }
    }
  }
}
```

For more information on sources, see the sources section.

Data objects in Signal K are organized hierarchically, for example data related to navigation such as position, speed through water and heading are all organized under a `navigation` sub-topic within the `vessel` object. Each data object has a `value` property which holds the actual value for that specific key. The `value` property may contain a number, a string or another object. Signal K keys that are object valued are object valued because the values don't have much semantic meaning individually. For example position – latitude doesn't have much meaning without an associated longitude. Therefore, these (and altitude) are grouped together in a single `navigation.position` key.

The values are always SI units, and always the same units for the same key. Therefore, `speedOverGround` is always meters per second, never knots, km/hr, or miles/hr. This means you never have to send units with data, the units are specific for a key, and defined in the data schema. A simplified version of the JSON schema with the units is available in Keys Reference in Appendix A. The units are also always specified in the values' metadata which is available via the REST API in the `meta.units` property. Besides the `units` property, `meta` provides a lot of other useful information for consumers of the data.

Finally, each data object also has a `timestamp` property which represents the time that the value was measured. Timestamps are in ISO 8601 format – specifically the RFC 3339 extension format, which is slightly more strict than the ISO specification. For instance, it requires four digit years and specifies that `T` is used as a separator between the data and time portions of the timestamp.

The ordering of keys is also not important, they can occur in any order. In fact, if you are designing a device which consumes Signal K data, it is important to remember that the JSON standard does not guarantee the order of properties in an object. You MUST NOT rely on the data you receive to always be in the same order within a Signal K message.

The full format is most useful for getting the initial state of a Signal K system, for example when a display device first connects to the network or for refreshing a device's state when it loses a network connection.

However sending the full data model is wasteful of both bandwidth and CPU, especially when there is a large amount of available data, or the consuming device is only interested in a small portion of it. In the majority of cases, it is preferable to only exchange small, specific portions of the data.

Delta Format

By far, the most commonly produced Signal K format is the delta format. Conceptually, the delta is an update to an existing Signal K data model. A device consuming deltas could either build up a view of the Signal K full tree by consuming and combining deltas or it could request from a Signal K server the current full tree model and apply deltas to that as they are received. It is also entirely possible for a device to remain essentially stateless and treat Signal K deltas as independent packets of data, much the same way as it would handle NMEA sentences or PGNs.

An example delta message is presented below.

```
{
  "context": "vessels.urn:mrn:imo:mmsi:234567890",
  "updates": [
    {
      "source": {
        "label": "N2000-01",
        "type": "NMEA2000",
        "src": "017",
        "pgn": 127488
      },
      "timestamp": "2010-01-07T07:18:44Z",
      "values": [
        {
          "path": "propulsion.0.revolutions",
          "value": 16.341667
        },
        {
          "path": "propulsion.0.boostPressure",
          "value": 45500
        }
      ]
    }
  ]
}
```

The top level of a delta message contains an `updates` property and an optional `context` property.

```
{
  "context": "vessels.urn:mrn:imo:mmsi:234567890",
  "updates": [...]
}
```

The optional `context` property roots the updates to a particular location in the Signal K tree. If `context` is missing it is assumed that the data is related to the `self` context. The `self` context is the `vessel` object which the `self` property of the full model points to.

Context is a path from the root of the full tree to the *container object*, which for vessel related data must refer to a vessel directly under `vessels`. The delimiter in the context path is `.` (period). In this case the context is `vessels.urn:mrn:imo:mmsi:234567890`. All subsequent data is relative to that location.

The `updates` property holds a JSON array of update objects, each of which may have a `source` property, a `timestamp` property and an array of `values` containing one or more value objects.

```
{
  "source": {
    "label": "N2000-01",
    "type": "NMEA2000",
    "src": "115",
    "pgn": 128267
  },
  "timestamp": "2014-08-15T16:00:00.081Z",
  "values": [
    {
      "path": "navigation.courseOverGroundTrue",
      "value": 2.971
    },
    {
      "path": "navigation.speedOverGround",
      "value": 3.85
    }
  ]
}
```

An `update` has a single `source` value and it applies to each of the `values` items. In cases where data can only come from a single source, such as an NMEA 0183 talker connected to a serial port, then the source may be omitted. However, if the delta is being passed on by a Signal K server or multiplexer then `source` must be filled in by the server so that downstream consumers can discern where the update comes from.

In cases where a Signal K producer does not have access to a real time clock or GPS time then `timestamp` should be omitted. Elements in the Signal K processing chain-such as a server receiving data from a producer-should fill in timestamp if it is missing in the incoming delta message.

Each `value` item is then simply a pair of `path` and `value` . The `path` must be a *leaf path*: it must be a path to a leaf the of the full model. A leaf is where the actual value of the Signal K property is and where `timestamp` , `$source` and `values` properties are in the full model. The value is often a scalar-a single numeric value, as in the example above-but it can also be an object. For example a `navigation.position` value would be an object like `{"latitude": -41.2936935424, "longitude": 173.2470855712}` .

There are some static properties in the full model that lack the support for multiple values and metadata such as source and timestamp. An example is a vessel‘s name, directly under the vessel‘s root. This static data may appear in the delta stream, for example when received in AIS transmission. In this case the value should be the subtree of the full model, starting from the vessel's root, with just the relevant parts, and the path must be empty, indicating that the value should be merged to the full model mounted where the delta‘s context property points:

```
{
  "context": "vessels.urn:mrn:imo:mmsi:234567890",
  "updates": [
    {
      "source": {...},
      "timestamp": "2014-08-15T19:02:31.507Z",
      "values": [
        {
          "path": "",
          "value": {
            "name": "WRANGO"
          }
        }
      ]
    }
  ]
}
```

Data Quality

Data transmitted in Signal K format is assumed to be corrected for known sensor inaccuracies such as wind angle offset due to misalignment of a masthead unit on the mast, but there is no *guarantee* that data is accurate, or within certain bounds. Different sources will have different data quality and normal vigilance is always required.

Missing or Invalid Data

A sensor or gateway/server may want to send a message indicating known invalid data or the fact that the sensor is functioning but can not provide data, for example when a depth sensor has no bottom fix. In this case the value must be JSON `null` in the delta message and the server must return the value as a JSON `null` in the REST API.

Message Integrity

Many messaging systems specify checksums or other forms of message integrity checking. Signal K assumes a reliable transport will guarantee a valid message. This is true of TCP/IP and some other transports but not always the case. For other transports (e.g. RS-232 serial) a specific extended data format will apply, which is suited to that transport. Hence at the message level no checksum or other tests need to be made.

Encoding/Decoding

The JSON message format is supported across most programming environments and can be handled with any convenient library.

On micro-controllers with limited RAM it may be necessary to read and write Signal K data using a streaming process rather than reading the entire message into RAM before processing. There is an implementation of Signal K JSON streaming on an Arduino Mega (4K RAM) in the related Freeboard project.

Multiple Values for a Key

There are two use cases for multiple values for a single data point

- Multiple instances of a common device-e.g. two engines or multiple batteries.
- Multiple devices providing duplicate data-multiple values for the same Signal K key from different sensors. This is fairly common as most boats have multiple sensors capable of generating the same data (but not necessarily the same value). For example, course over ground (COG) may come from both a compass and GPS or a boat may be equipped with multiple depth sounders.

Multiple Instances of a Common Device

Some parts of the Signal K schema are **device oriented**.

For example, many boats have multiple batteries. However each battery has multiple, common quantities like voltage, current and temperature. In this case, it makes more sense for these values to be organized by instance. Therefore, in the Signal K model, each battery bank is it's own instance: for example `electrical.batteries.starter` and `electrical.batteries.house` . Then beneath that prefix there are the various properties for each battery.

This organisation allows a user interface to organise the individual readings in meaningful groups and allows consumers to query all the values related to that piece of equipment via the REST API. Furthermore, this structure maintains the primary requirement that a given data value have a fixed and unique URI, but gives flexibility in the structure and complexities of data.

The same device centric organisation is used within the `propulsion` subschema, to support the common use case of two engines via `propulsion.port` and `propulsion.starboard` .

> The values `starter` , `house` , `port` and `starboard` are examples and not specified in the schema. You are free to use application specific values within the regexp specified in the JSON schema.

Multiple Devices Providing Duplicate Data

It is quite possible for a key value to come from more than one device. Many modern devices have a built in GPS receiver and as a result on any given boat there may be several sources of position, speed over ground, and heading. Multiple depth sounders are also common, often installed to port and starboard on monohull sailboats or in each hull of a catamaran.

Given that the validity of these various sources of data may change in a context-specific way, Signal K provides a mechanism for these values to be grouped together so that the consumer of the data may choose which value (or values) to display[1].

When dealing with multiple sources of data, it becomes important to know where the data is coming from. Signal K provides a mechanism for that in the form of the `sources` top level object and references into that object via the `$source` property. See sources for detailed information on the structure of the `sources` object and how it is referenced by the `$source` property.

The first source of a particular data point becomes the default source for that data and a normal Signal K object is created.

```
{
  "self": "urn:mrn:signalk:uuid:c0d79334-4e25-4245-8892-54e8ccc8021d",
  "version": "0.9.0",
  "vessels": {
    "urn:mrn:signalk:uuid:c0d79334-4e25-4245-8892-54e8ccc8021d": {
      "uuid": "urn:mrn:signalk:uuid:c0d79334-4e25-4245-8892-54e8ccc8021d",
      "navigation": {
        "courseOverGroundTrue": {
          "value": 3.61562407843144,
          "$source": "ttyUSB0.GP",
          "timestamp": "2017-04-03T06:14:04.451Z"
        }
      }
    }
  }
}
```

It has come from device `sources.ttyUSB0.GP`, where further details can be found.

If another value with different source arrives, the Signal K server will add the `values` attribute with values from both the first and second sources. The initial source‘s data will continue to populate the `value` property in the key.

```
{
  "self": "urn:mrn:signalk:uuid:c0d79334-4e25-4245-8892-54e8ccc8021d",
  "version": "0.9.0",
  "vessels": {
    "urn:mrn:signalk:uuid:c0d79334-4e25-4245-8892-54e8ccc8021d": {
      "uuid": "urn:mrn:signalk:uuid:c0d79334-4e25-4245-8892-54e8ccc8021d",
      "navigation": {
        "courseOverGroundTrue": {
          "value": 3.615624078431440,
          "$source": "ttyUSB0.GP",
          "timestamp": "2017-04-03T06:14:04.451Z",
          "values":{
            "ttyUSB0.GP.RMC":{
              "value": 3.615624078431440,
              "timestamp": "2017-04-03T06:14:04.451Z"
            },
            "n2k.ikommunicate.128267":{
              "value": 3.615624078431453,
              "timestamp": "2017-04-03T06:14:04.451Z"
            }
          }
        }
      }
    }
  },
  "sources":{
    "ttyUSB0": {
      "GP": {
        "sentences": {
          "RMC": "2017-04-03T06:14:04.451Z"
        }
      }
    },
    "ikommunicate": {
      "2": {
        "n2k": {
          "src": "2",
          "pgns": {
            "128267": "2017-04-03T06:14:05.221Z"
          }
        }
      }
    }
  }
}
```

Multiple Values in Delta Messages

When a client subscribes to `navigation.courseOverGroundTrue` , they receive *all* the values held. The update message does not include the `values` path, the case above looks like:

```
{
  "context": "vessels.urn:mrn:signalk:uuid:c0d79334-4e25-4245-8892-54e8ccc8021
  "updates": [
    {
      "source": {
        "label": "GPS-1",
        "type": "NMEA0183",
        "talker": "GP",
        "sentence": "RMC"
      },
      "timestamp": "2017-04-03T06:14:04.451Z",
      "values": [
        {
          "path": "navigation.courseOverGroundTrue",
          "value": 3.615624078431440
        }
      ]
    },
    {
      "source": {
        "label": "actisense",
        "type": "NMEA2000",
        "src": "115",
        "pgn": 128267
      },
      "timestamp": "2017-04-03T06:14:04.451Z",
      "values": [
        {
          "path": "navigation.courseOverGroundTrue",
          "value": 3.615624078431453
        }
      ]
    }
  ]
}
```

Individual updates can be distinguished by their source.

If a client wants only the values of a specific source it should subscribe to a path that includes the full path under `values` including the source reference key of the source. The source reference should be enclosed in square brackets: `navigation.courseOverGroundTrue.values[n2k./dev/ikommunicate.128267]` . The client can retrieve the relevant data via REST API.

Note: The exact format of the update message is affected by the subscription policy. A policy of `instant` will result in changes being sent immediately, so typically one item in `values` per update. A policy of `fixed` will result in periodic updates which may contain many items in `values` .

The update allows grouping `values` by `source` .

[1] Specifying preferred sources is still an under-development enhancement to the Node server.

Metadata

A key part of Signal K is the ability for data consumers such as apps or MFDs to automatically configure themselves based on settings retrieved from the server. The metadata component of Signal K facilitates this through an optional `meta` object attached to each key in the Signal K data model.

Rationale

In an environment where various critical pieces of information are displayed in multiple locations it becomes quite difficult to ensure that all of these devices use the same scale and react the same way to changes in the data. This is especially true in an environment where these devices are not tied to the boat. A crew member may bring a personal tablet with them for their tactician role during a Wednesday evening race or a harbor pilot may bring a laptop on board loaded with local charts. If these devices can load critical configuration data from a central server on the boat, this saves time and prevents costly or even disastrous mistakes from occurring due to misconfigured devices.

Metadata for a Data Value

The `meta` object exists at the same level as `value` and `$source` in each key in the Signal K data model.

```
{
  "displayName": "Port Tachometer",
  "longName": "Engine 2 Tachometer",
  "shortName": "Tacho",
  "description": "Engine revolutions (x60 for RPM)",
  "units": "Hz",
  "timeout": 1,
  "displayScale": {"lower": 0, "upper": 75, "type": "linear"},
  "alertMethod": ["visual"],
  "warnMethod": ["visual"],
  "alarmMethod": ["sound", "visual"],
  "emergencyMethod": ["sound", "visual"],
  "zones": [
    {"upper": 4, "state": "alarm", "message": "Stopped or very slow"},
    {"lower": 4, "upper": 60, "state": "normal"},
    {"lower": 60, "upper": 65, "state": "warn", "message": "Approaching maximu
    {"lower": 65, "state": "alarm", "message": "Exceeding maximum"}
  ]
}
```

In the example `meta` object above, a definition is provided for an analog RPM gauge for the port engine. It provides a few different options for the consumer to use to display the name of the measurement and explicitly calls out the unit of measure.

`description`

This is the description for the Signal K path and must always be the same as the description property within the Signal K Schema for that path.

`displayName`

This is used on or near any display or gauge which shows the data. Units can change and are presented separately, therefore no indication of units should be included in displayName. eg. "Port"

`longName` `shortName`

These are human readable names for the particular instance of this value. Presented to users to identify the value. The short version may be used by consumers where space is at a premium. As with displayName units should not be included.

`timeout`

The `timeout` property tells the consumer how long it should consider the value valid. This value is specified in seconds, so for a high speed GPS sensor it may 0.1 or even 0.05.

The `displayScale` object provides information regarding the recommended type and extent of the scale used for displaying values. The `lower` and `upper` indicate the extent of the scale to be shown. Some values are better shown on a non linear scale, for example logarithmic for luminosity, depth, signal strength, etc. whilst others may be better on a squareroot scale eg. depth, windspeed. `type` has possible values of `linear` (default), `logarithmic` , `squareroot` or `power` . When `"type": "power"` is specified an additional property `power` must be present to define the power. Note that a power of 0.5 is equivalent to `squareroot` and a power of 1 is equivalent to linear. In using these scales the type defines the function which is applied to all values in order to calculate % scale deflection of the pointer/needle/plot:

Type	Formula for % deflection
linear	(V - L)/(U - L)
logarithmic	(log(V) - log(L) / (log(U) - log(L))
squareroot	(√V - √L) / (√U - √L)
power (P)	$(V^P - L^P) / (U^P - L^P)$

Where: V = value, L = lower bound of the gauge, U = upper bound of the gauge and P = power

Note that on a logarithmic scale neither L nor U can be zero.

The `alertMethod` , `warnMethod` , `alarmMethod` and `emergencyMethod` properties tell the consumer how it should respond to an abnormal data condition. Presently the values for these properties are `sound` and `visual` and the method is specified as an array containing one or both of these options. It is up to the consumer to decide how to convey these alerts.

`alertMethod` , etc

The `alertMethod` , `warnMethod` , `alarmMethod` and `emergencyMethod` properties tell the consumer how it should respond to an abnormal data condition. Presently the values for these properties are `sound` and `visual` and the method is specified as an array containing one or both of these options. It is up to the consumer to decide how to convey these alerts.

`zones`

The last property in the `meta` object is the `zones` array. This provides a series of hints to the consumer which can be used to properly set a range on a display gauge and also color sectors of a gauge to indicate normal or dangerous operating conditions. It also tells the consumer which state the data is in for a given range. Combined with the alert method properties, all Signal K consumers can react the same way to a given state.

The possible states in ascending order of severity are:

State/Zone	Description
nominal	this is a special type of normal state/zone (see below)
normal	the normal operating range for the value in question (default)
alert	Indicates a safe or normal condition which is brought to the operators attention to impart information for routine action purposes
warn	Indicates a condition that requires immediate attention but not immediate action
alarm	Indicates a condition which is outside the specified acceptable range. Immediate action is required to prevent loss of life or equipment damage
emergency	the value indicates a life-threatening condition

`nominal` : A example use of this is for engine monitoring eg. coolant temperature where there is a normal (no warnings) (green) zone between say 70C and 110C, but when the temperature is between 80C and 90C (`nominal`) the needle doesn't move at all (typically remains vertical or horizontal). This is really useful if you have many gauges (multiple motors with multiple sensors) where it is very easy to spot that every needle is pointing in exactly the same direction. Use of nominal will only be relevant if the gauge/display design permits it.

The `upper` and `lower` values in the zones do not need to be contiguous, they don't have to both be present in a zone, nor do they need to be within the bounds of the `upper` and `lower` specified in `displayScale` . When they are outside of the `displayScale` range they will still give rise to alerts. Both `upper` and `lower` values are considered to be inclusive.

If zones overlap each other the state/zone with the highest severity will take precedence. This is true for both alerts and gauge/display rendering. Any part of the range which is not explicitly within a zone is considered to be `normal` (the default). As such, zones with a state of `normal` have no effect and their removal would result in no changes to either displays or alerts.

There can be multiple zones with the same `state`, for example if a different message is required, or if they are on different parts of the scale.

Signal K servers will use the `zone` information to monitor any data which has a `meta` object and raise a generic alarm event. See the section on Alarm Handling for more.

Implicit Metadata

All keys in the Signal K specification must have a `description`, and where the key is a numeric value it must have `units`.

If a client requests the `meta` property for a valid Signal K key via the HTTP REST interface, the server must return the `description` and, if applicable, `units`, even if no value has ever been generated for that key.

If a key has values determined by an enum, the server should include the enum in the meta. NB. in future versions it is likely that this will become a mandatory requirement for the server.

```
// GET /signalk/v1/api/vessels/self/environment/depth/belowKeel/meta

{
  "units": "m",
  "description": "Depth below keel"
}
```

See keyswithmetadata.json

Default Configuration

Signal K does not provide a default set of metadata, it is up to the owner or their installer to configure their Signal K environment appropriately for their vessel. However, by centralizing this configuration they will only need to do it one time and any future consumers will automatically use this configuration.

Alarm Management

An alarm watch is set by setting the `meta.zones` array appropriately. A background process on the server checks for alarm conditions on any attribute with a `meta.zones` array. If the keys value is within a zone the server sets an alarm key similar to `vessels.self.notifications.[original_key_suffix]`, e.g. an alarm set on `vessels.self.navigation.courseOverGroundTrue` will become `vessels.self.notifications.navigation.courseOverGroundTrue`.

The object found at this key should contain the following:

```
{
  "message": "any text",
  "state": "[normal|alert|warn|alarm|emergency]"
}
```

Other Benefits

While not strictly part of the Signal K specification, metadata configuration could be shared between boats or even provided by manufacturers of production boats or by component suppliers such as engine or refrigerator manufacturers. Also, any device which implements Signal K should provide a baseline metadata configuration. As this standard becomes more widespread, less individual configuration will need to be performed.

Signal K Data Sources

Signal K provides a method to identify the specific device and—if available—the NMEA sentence or PGN which generated a particular value. This is handled in two ways. The first is with a pointer to a device in the `sources` section of the Signal K data model. When viewing a full Signal K data model, this is the method you will see. Every Signal K data object will have a `$source` property (note the dollar sign sigil which indicates the value is a pointer) which contains a dot separated path to an object relative to the top-level `sources` section. When receiving Signal K data via deltas, you may also see sources referenced this way. However, all current implementations of Signal K provide source data directly embedded in the delta message. This is done using the `source` property of the delta message.

Pointer Method

An example of the pointer method is shown below.

```
{
  "version": "1.0.0",
  "self": "vessels.urn:mrn:signalk:uuid:705f5f1a-efaf-44aa-9cb8-a0fd6305567c",
  "vessels": {
    "urn:mrn:signalk:uuid:705f5f1a-efaf-44aa-9cb8-a0fd6305567c": {
      "uuid": "urn:mrn:signalk:uuid:705f5f1a-efaf-44aa-9cb8-a0fd6305567c",
      "navigation": {
        "speedOverGround": {
          "value": 4.32693662,
          "$source": "ttyUSB0.GP",
          "sentence": "RMC",
          "timestamp": "2017-05-16T05:15:50.007Z"
        }
      }
    }
  },
  "sources": {
    "ttyUSB0": {
      "label": "ttyUSB0",
      "type": "NMEA0183",
      "GP": {
        "talker": "GP",
        "sentences": {
          "RMC": "2017-04-03T06:14:04.451Z"
        }
      }
    }
  }
}
```

This is a full Signal K model with a single data source and a single data object. The `navigation.speedOverGround` object references the `ttyUSB0.GP` device via the value of the `$source` property. In addition to `$source`, the `sentence` property is also included which identifies the specific NMEA 0183 sentence recieved from the data source which was converted to Signal K.

Within the `sources` section devices are conventionally organized by the physical connection to the Signal K device and an identifier for the specific source device. In the case above, the first level of the hierarchy is the UNIX device identifier for a USB serial port and the second level is the NMEA 0183 talker ID of the paddlewheel sensor.

Direct Inclusion Method

This method is only implemented in delta messages. An example delta with an inline source is shown below.

```
{
  "updates": [
    {
      "source": {
        "label": "ttyUSB0",
        "type": "NMEA2000",
        "pgn": 127251,
        "src": "204"
      },
      "timestamp": "2017-04-15T20:38:26.709Z",
      "values": [
        {
          "path": "navigation.rateOfTurn",
          "value": -0.000412469
        }
      ]
    }
  ],
  "context": "vessels.urn:mrn:imo:mmsi:338184312"
}
```

This particular source is an NMEA 2000 device so it has the `pgn` and `src` properties, but NMEA 0183 or other source types are similarly supported. An NMEA 0183 source would be structured like this:

```
{
  "label": "NMEA0183-0",
  "type": "NMEA0183",
  "sentence": "RMC",
  "talker": "GP"
}
```

A final example using a non-NMEA source, in this case an I²C sensor.

```
{
  "type": "I2C",
  "label": "I²C Bus #0",
  "src": "14"
}
```

Here, `src` is the address of the device on the I²C bus.

Sources Group

A Signal K device capable of generating a full data model will have a top level (i.e. at the same level as `vessels` , `version` and `self`) group called `sources` . This group provides detailed information about each network bus connected to the Signal K gateway or server. Sources are organized hierarchically, following a bus-source-type structure.

An example of multiple sources is shown below.

```
{
  "vhf": {
    "label": "AIS Receiver",
    "type": "VHF",
    "112334556": {
      "ais": {
        "aisType": 15
      }
    },
    "394299113": {
      "ais": {
        "aisType": 15
      }
    }
  },
  "ttyUSB0": {
    "label": "NMEA 0183",
    "type": "NMEA0183",
    "II": {
      "talker": "II",
      "sentences": {
        "VHW": "2018-04-16T01:34:03.881Z"
      }
    }
  },
  "ttyUSB1": {
    "label": "NMEA 2000",
    "type": "NMEA2000",
    "3": {
      "1": {},
      "2": {},
      "n2k": {
        "src": "3",
        "pgns": {
          "126992": "2017-04-15T18:44:59.006Z"
        }
      }
    }
  }
}
```

You may notice two odd lines in the NMEA 2000 source `"3"` : `"1": {}` and `"2": {}'` . These are placeholders for NMEA 2000 "instance" values. These are here to provide a valid schema for certain NMEA 2000 PGNs relating to temperature. Because temperature paths in the current versio of Signal K do not have the instance inline, consumers must look at the instance part of the `$source` or `source` property to determine which specific sensor provided the data.

URLs and Ports

While Signal K is a transport-agnistic protocol, there are certain conventions that have been established for use on the Web and by clients and servers using HTTP and WebSockets.

Ports

The Signal K HTTP and WebSocket services SHOULD be found on the usual HTTP/S ports (80 or 443). The services SHOULD be found on the same port, but may be configured for independent ports and MAY be configured for ports other than HTTP/S.

A Signal K server MAY offer Signal K over TCP or UDP, these services SHOULD be on port 8375[1].

If an alternate port is needed it SHOULD be an arbitrary high port in the range 49152–65535[2].

URL Prefix

The Signal K applications start from the `/signalk` root. This provides some protection against name collisions with other applications on the same server. Therefore the Signal K entry point will always be found by loading `http(s)://«host»:«port»/signalk`.

---

[1] This has not been registered with IANA yet. It is the ASCII decimal code for SK.
[2] This is the private use section of IP ports specified as reserved by IANA.

REST API

Signal K producers MAY implement an HTTP API which consumers can use to self-configure, poll for Signal K data or make configuration changes. As specified in the previous section, all URLs for interacting with Signal K are rooted at `/signalk` .

A Signal K server implementing the HTTP API may support http/2.

GET /signalk

Making a `GET` request to `/signalk` returns a JSON object which specifies the available Signal K endpoints and some information about the server. Also see versioning for details about `version` strings.

```
{
  "endpoints": {
    "v1": {
      "version": "1.0.0-alpha1",
      "signalk-http": "http://localhost:3000/signalk/v1/api/",
      "signalk-ws": "ws://localhost:3000/signalk/v1/stream"
    },
    "v3": {
      "version": "3.0.0",
      "signalk-http": "http://localhost/signalk/v3/api/",
      "signalk-ws": "ws://localhost/signalk/v3/stream",
      "signalk-tcp": "tcp://localhost:8367"
    }
  },
  "server": {
    "id": "signalk-server-node",
    "version": "0.1.33"
  }
}
```

This response is defined by the `discovery.json` schema. In this example, the server supports two versions of the specification: `1.alpha1` and `3.0` . For each version, the server indicates which transport protocols it supports and the URL that can be used to access that protocol‘s endpoint. Clients should use one of these published endpoints based on the protocol version they wish to use.

The server must only return valid URLs and should use IANA standard protocol names such as `http` . However, a server may support unofficial protocols and may return additional protocol names; for example, the response above indicates the server supports a `signalk-tcp` stream over TCP at on port 8367.

A server may return relative URIs that the client must resolve against the base of the original request.

A server MAY return information about itself in the `server` property. The id and version scheme is not defined as part of the specification and there is no registry for id values. If providfed, the `id` and `version` MUST be the same values as `swname` and `swvers` within the DNS-SD advertisement (if implemented), and also the `id` MUST provide the same value as `name` within the Websocket hello message (if implemented).

/signalk/«version»/api/

Note the trailing slash in the path

The base URL MUST provide a Signal K document that is valid according to the full Signal K schema specification. The contents SHOULD be all the current values of the data items the server knows in the Signal K full format as specified in Full and Delta Models.

/signalk/«version»/api/*

The Signal K data SHOULD be available via the REST API. For example, `GET /signalk/v1/api/vessels` should return all of the data under the `vessels` container in JSON format. Likewise, `GET /signalk/v1/api/vessels/urn:mrn:signalk:uuid:c0d79334-4e25-4245-8892-54e8ccc8021d` should return data for one specific vessel. In other words, the full Signal K data model SHOULD be traversable by any client making GET requests to an arbitrary depth.

History snapshot retrieval

A server MAY support retrieving historical data. The history snapshot retrieval endpoint is `/signalk/v1/snapshot` and functions like the full model endpoint at `/signalk/v1/api`. The client specifies the requested timestamp with request parameter `time`, for example `https://localhost:3443/signalk/v1/snapshot/vessels/self?time=2018-08-24T15:19:09Z`. The server will attempt to create the request part of the full model at the requested time.

A server MAY respond with `501 Not Implemented` status code if it does not support history snapshot retrieval and with `400 Bad Request` if it does not have data for the requested timestamp. A `404 Not Found` response is also acceptable to be backwards compatible.

Streaming API

WebSocket API: /signalk/«version»/stream

Initiates a WebSocket connection that will start streaming the server's updates as Signal K delta messages. You can specify the contents of the stream by using the `subscribe` query parameter.

- ws://hostname/signalk/«version»/stream?subscribe=self
- ws://hostname/signalk/«version»/stream?subscribe=all
- ws://hostname/signalk/«version»/stream?subscribe=none

With no query parameter the default is `self` , which will stream the data related to the `self` object. `all` will stream all the updates the server sees and `none` will stream only the heartbeat, until the client issues subscribe messages in the WebSocket stream.

A server may send the latest values it has cached when a client connects via WebSocket. A client can control this behavior with query parameter `sendCachedValues` . `false` will suppress sending the values and `true` force it. With no `sendCachedValues` parameter the server should send them.

If a server does not support some streaming options listed in here it must respond with HTTP status code `501 Not Implemented` .

See Subscription Protocol for more details.

Connection Hello

Upon connection the server MUST send a 'hello' JSON message, for example:

```
{
    "name": "foobar marine server",
    "version": "1.0.4",
    "timestamp": "2018-06-21T15:09:16.704Z",
    "self": "vessels.urn:mrn:signalk:uuid:c0d79334-4e25-4245-8892-54e8ccc8021d
    "roles": [
        "master",
        "main"
    ]
}
```

This response is defined by the `hello.json` schema.

The server MUST provide:

- `roles` which specifies which roles the server is capable of providing. See roles for details about possible server roles.
- `version` which specifies the version of the SignalK schema and APIs that the server is using. See versioning for details about `version` strings.

The server SHOULD provide:

- `timestamp` but only if the server is equipped with a time source and it has been set.

The server MAY provide:

- `self` is the unique identifier of the vessel using the URN format specified for the uuid field in the Signal K schema. It may also use the URN format specified for the mmsi field in the Signal K schema if it exists. This is only provided if the server relates to a specific vessel, aircraft, aid to navigation or sar.
- `name` is the name of the Signal K server software, e.g. signalk-server

`name` , `self` and `roles` MUST return the same values as provided in the `swname` , `self` and `roles` properties within the DNS-SD advertisement (if implemented).

`version` MUST be the same value as `version` within the associated endpoints list provided by the http `GET` request to `/signalk` within the REST API (if implemented).

History playback

The server MAY support history playback from a certain point in time with a specified rate.

To create a WebSocket connection that plays back data the client uses the request parameter `startTime` to specify the start timestamp and the optional request parameter `playbackRate` to specify the rate. Rate value parameter is a floating point value with value `1` equal to real time playback and for example `0.5` to half the real time rate and `5` to five times real time rate. Omitting the `playbackRate` will result in real time playback.

The playback api is located at `/signalk/v1/playback` . An example url for history playback streaming: `wss://localhost:3443/signalk/v1/playback?subscribe=self&startTime=2018-08-24T15:19:09Z&playbackRate=5` .

The hello message for a history playback stream MUST NOT contain the `timestamp` property and MUST include the properties `startTime` and `playbackRate` . The delta stream format for history playback is the normal streaming format. Timestamps indicate the time data was originally captured.

```
{
    "name": "foobar marine server",
    "version": "1.1.4",
    "startTime": "2018-08-24T15:19:09Z",
    "playbackRate": 1,
    "self": "vessels.urn:mrn:signalk:uuid:c0d79334-4e25-4245-8892-54e8ccc8021d
    "roles": [
        "master",
        "main"
    ]
}
```

A server MAY respond with `501 Not Implemented` status code if it does not support history playback and with `400 Bad Request` if it does not have data to play back for the given time period. A `404 Not Found` response is also acceptable to be backwards compatible.

Streaming over TCP

A server MAY provide streaming delta service over TCP. See Urls and Ports and Discovery and Connection Establishment for more details.

The messages MUST be serialised as JSON with one message per line using line terminator `\r\n` (carriage return and newline).

As there is no way to specify the subscription policy using url parameters as when opening a WebSocket connection the initial subscription policy is `none` , no active subscriptions. The client can modify the subscriptions after connection is established.

Connection `hello` is the same as when using WebSockets.

Subscription Protocol

Introduction

By default a Signal K server will provide a new WebSocket client with a delta stream of the `vessels.self` record, as updates are received from sources. E.g. `/signalk/v1/stream` will provide the following delta stream, every time the log value changes.

```
{
  "context": "vessels.urn:mrn:imo:mmsi:234567890",
  "updates": [
    {
      "source": {
        "label": "N2000-01",
        "type": "NMEA2000",
        "src": "115",
        "pgn": 128275
      },
      "values": [
        {
          "path": "navigation.trip.log",
          "value": 43374
        },
        {
          "path": "navigation.log",
          "value": 17404540
        }
      ]
    }
  ]
}
```

> Below we refer to WebSockets, but the same process works in the same way over any transport. E.g. for a raw TCP connection the connection causes the above message to be sent, and sending the subscribe messages will have the same effect as described here.

This can be a lot of messages, many you may not need, especially if your boat has many sensors, or other data sources. Often you will want to subscribe to a much smaller range of data. Especially for single value displays, it does not make sense to get the entire data stream when only a single value is wanted.

First you will want to unsubscribe from the current default (or you may have already connected with `ws://hostname/signalk/v1/stream?subscribe=none`). To unsubscribe all create an `unsubscribe` message with wildcards and send the message over the WebSocket connection:

```
{
  "context": "*",
  "unsubscribe": [
    {"path": "*"}
  ]
}
```

To subscribe to the required criteria send a suitable subscribe message:

```
{
  "context": "vessels.self",
  "subscribe": [
    {
      "path": "navigation.speedThroughWater",
      "period": 1000,
      "format": "delta",
      "policy": "ideal",
      "minPeriod": 200
    },
    {
      "path": "navigation.logTrip",
      "period": 10000
    }
  ]
}
```

- `path=[path.to.key]` is appended to the context to specify subsets of the context. The path value can use the wildcard `*`. A wildcard in the middle of a path (`propulsion/*/oilTemperature`) allows any value for that part and a wildcard at the end (`propulsion/port/*`) matches all paths beginning with the specified prefix.

The following are optional, included above only for example as it uses defaults anyway:

- `period=[millisecs]` becomes the transmission rate, e.g. every `period/1000` seconds. Default: 1000
- `format=[delta|full]` specifies delta or full format. Default: delta
- `policy=[instant|ideal|fixed]`. Default: ideal
  - `instant` means send all changes as fast as they are received, but no faster than `minPeriod`. With this policy the client has an immediate copy of the current state of the server.
  - `ideal` means use `instant` policy, but if no changes are received before `period`, then resend the last known values.
  - `fixed` means simply send the last known values every `period`.
- `minPeriod=[millisecs]` becomes the fastest message transmission rate allowed, e.g. every `minPeriod/1000` seconds. This is only relevant for policy='instant' to avoid swamping the client or network.

You can subscribe to multiple data keys multiple times, from multiple apps or devices. Each app or device simply subscribes to the data it requires, and the server and/or client implementation may combine subscriptions to avoid duplication as it prefers on a per connection basis. At the same time it is good practice to open the minimum connections necessary, for instance one WebSocket connection shared between an instrument panel with many gauges, rather then one WebSocket connection per gauge.

Multiple value handling in subscriptions

A subscription to a key is for all the updates to that key. If there are multiple sources generating data for that key the client will get all their updates.

If a client wants only the values of a single source it should subscribe to a path that includes the full path under `values` including the source reference key of the source. The source reference should be enclosed in square brackets: `navigation.speedThroughWater.values[n2kFromFile.43]` . The client can retrieve the relevant data via REST API. See Multiple Values for more information.

Single use, or intermittent data

When data is required once only, or upon request the `subscribe/unsubscribe` method should not be used. If the client is http capable the REST API is a good choice, or use `get/list/put` messages over WebSockets or TCP.

Use Cases and Proposed Solutions

Local boat individual instruments

A gauge-type display for just one or a few data items for the 'self' vessel should be able to specify that it only wants those items for the self vessel.

This can be achieved by a default WebSocket connection `/signalk/v1/stream?subcribe=none` , then sending a JSON message:

```
{
  "context": "vessels.self",
  "subscribe": [
    {
      "path": "environment.depth.belowTransducer"
    },
    {"path": "navigation.speedThroughWater"}
  ]
}
```

The JSON format is also viable over a simple TCP or serial transport, and is therefore supported as the primary subscription method.

Map display with all known vessel positions & directions, served over 3G cellular connection

```
{
  "context": "vessels.*",
  "subscribe": [
    {
      "path": "navigation.position",
      "period": 120000,
      "policy": "fixed"
    },
    {
      "path": "navigation.courseOverGround",
      "period": 120000,
      "policy": "fixed"
    }
  ]
}
```

The result is a delta message of the Signal K data with just `position` and `courseOverGround` branches for all known vessels, sent every 2 minutes (120 seconds) even if no data has been updated.

Position of a certain vessel, immediately it changes, but once per minute at most

```
{
  "context": "vessels.230029970",
  "subscribe": [
    {
      "path": "navigation.position",
      "minPeriod": 60000,
      "policy": "instant"
    }
  ]
}
```

The result will be delta position messages for vessel 230029970, broadcast whenever it changes, but with minimum interval of 60 seconds. Messages are delayed to meet the minimum interval with newer messages overriding the previous message in the buffer.

Discovery and Connection Establishment

Service Discovery

A Signal K server SHOULD advertise its services using DNS Service Discovery (DNS-SD) via Multicast DNS (mDNS); also known as Bonjour. The server MUST provide DNS Service (SRV) Records and Text (TXT) Records describing the Signal K interfaces it provides. These service identifiers are:

- `_http._tcp` for the server's web interface
- `_signalk-http._tcp` for the Signal K REST API
- `_signalk-ws._tcp` for the WebSocket data stream
- `_signalk-tcp._tcp` for the TCP data stream

If a server is providing Signal K via secure versions of HTTP or WebSockets then they MUST be able to provide a redirection to the secure versions of these protocols.

If a Signal K server is using DNS-SD, it MUST provide the following parameters (key/value pairs) in the TXT record portion of the DNS-SD advertisement:

- `txtvers` is a US-ASCII decimal number identifying the version of the DNS-SD record. Currently, this MUST have a value of 1
- `roles` specifies which roles the server is capable of providing. See Roles below for details

The server MAY provide the following values:

- `self` is the unique identifier of the vessel using the URN format specified for the `uuid` field in the Signal K schema. It may also use the URN format specified for the `mmsi` field in the Signal K schema if it exists.
- `swname` is the name of the Signal K server software, e.g. signalk-server-node
- `swvers` is the version of the Signal K server software

`swname` , `self` amd `roles` MUST be the same values as provided by the `name` , `self` and `roles` properties within the Websocket hello message (if implemented).

An example DNS-SD record set is shown below.

```
Service data for service 'signalk-http' of type '_signalk-http._tcp' in domain
    Host 10-1-1-40.local (10.1.1.40),
    port 80,
    TXT data: [
        'txtvers=1',
        'roles=master,main',
        'self=vessels.urn:mrn:signalk:uuid:c0d79334-4e25-4245-8892-54e8ccc8021
        'swname=signalk-server',
        'swvers=0.1.23'
        ]

Service data for service 'signalk-ws' of type '_signalk-ws._tcp' in domain 'lo
    Host 10-1-1-40.local (10.1.1.40),
    port 3000,
    TXT data: [
        'txtvers=1',
        'roles=master,main',
        'self=urn:mrn:signalk:uuid:c0d79334-4e25-4245-8892-54e8ccc8021d',
        'swname=signalk-server',
        'swvers=0.1.23'
        ]
```

These records are advertising a Signal K server with the HTTP REST API on port 80 and the WebSocket data stream on port

1. The server identifies as having the `master` and `main` roles and provides a `self` identifier as a UUID.

Roles

The four possible values for `roles` are `master` , `slave` , `main` , and `aux` . These are defined below.

Master

`master` is the canonical source for identity and configuration information for the entire vessel.

If there is only one master on the vessel, then it should also provide the main role. The combination of master and main informs a client that this server is actively providing identifying information.

Main and Aux

If there are more than one masters on the vessel, EXACTLY ONE server should advertise both master and main. All other masters should advertise master and aux. Clients should only use the master aux servers for identifying information if the master main is not available.

Any server identifying as master MUST be able to provide at a minimum the unique identifier (self) for the vessel.

Slave

Any server providing the `slave` role should retrieve identity and configuration information from the master server. Slave servers MAY provide configuration and identity information for themselves, but this identity MUST NOT be considered valid for the entire vesssel.

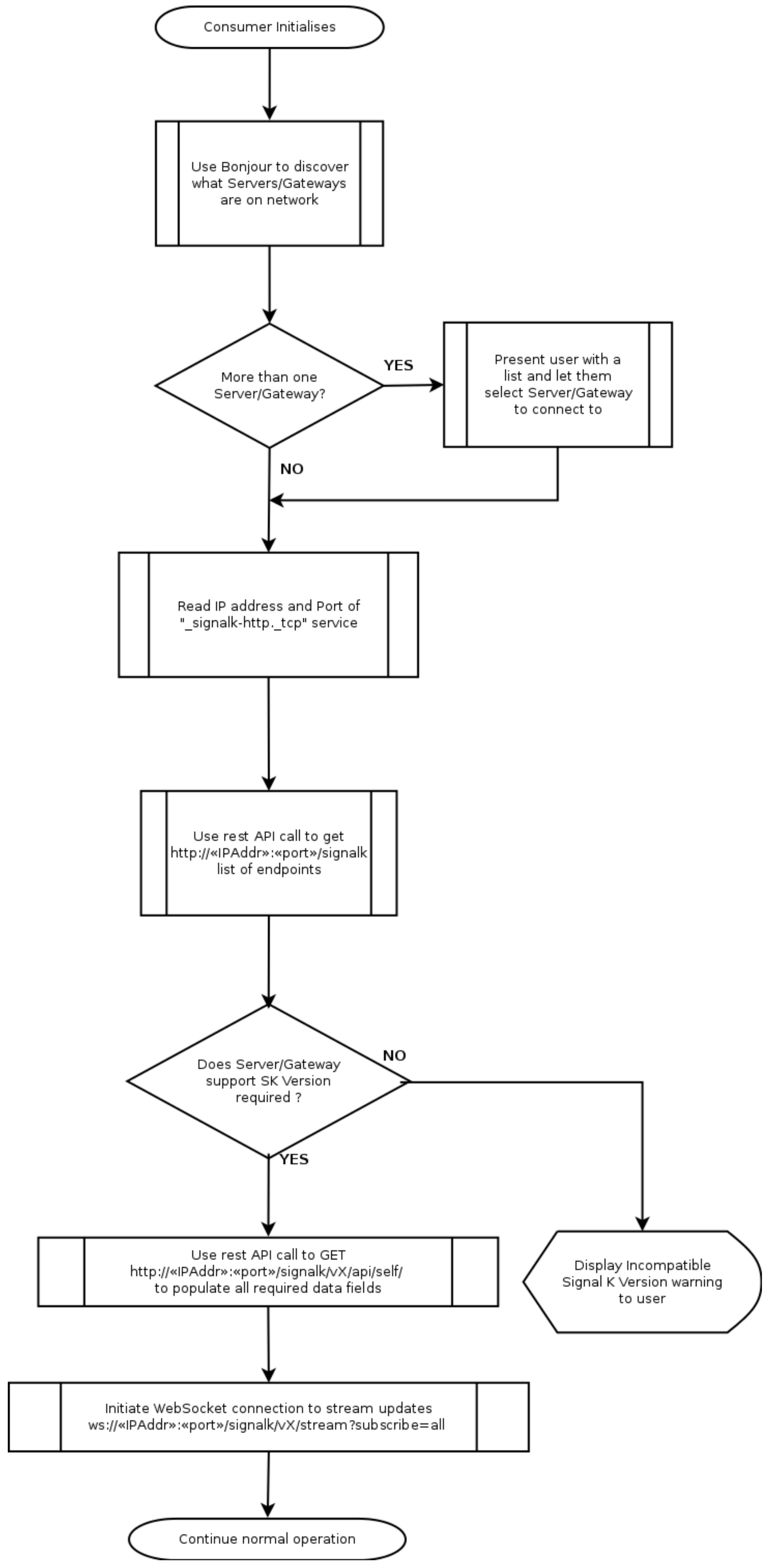
Main and Aux

The use of main and aux have not been defined for the slave role at this time.

Connection Establishment

Using the information above a web client or HTTP capable device can discover and connect to a Signal K server using the following process:

- Query for Signal K services using mDNS
- Connect to the host and port advertised as 'signalk-http' via HTTP (e.g. `http://10.1.1.40:80`)
- Per the Urls and Ports section, make a GET request for `/signalk` to retrieve a JSON object containing an `endpoints` JSON object
- Make further REST calls for more specific data, or open a websocket connection to start streaming updates.


Consumer Initialises
Use Bonjour to discover what Servers/Gateways are on network
More than one Server/Gateway?
YES
Present user with a list and let them select Server/Gateway to connect to
NO
Read IP address and Port of "_signalk-http._tcp" service
Use rest API call to get http://«IPAddr»:«port»/signalk list of endpoints
Does Server/Gateway support SK Version required ?
NO
Display Incompatible Signal K Version warning to user
YES
Use rest API call to GET http://«IPAddr»:«port»/signalk/vX/api/self/ to populate all required data fields
Initiate WebSocket connection to stream updates ws://«IPAddr»:«port»/signalk/vX/stream?subscribe=all
Continue normal operation

Alarm, Alert, and Notification Handling

Handling alarms, alerts, and notifications in Signal K is a multi-stage process. Alarms, alerts and notifications are all handled the same way, and are all referred to as alarms below.

We need a flexible model to define alarm conditions, and a standard way to announce and record them.

Alarm Process

- Define alarm states as zones in the meta object attached to any Signal K value. See [[Metadata for Data Values]]
- If the value is within an alarm zone raise the defined alarm.
- If the value goes out of the zone, remove the alarm by setting its value to null
- Alarms are raised by placing an alarm object in the `vessels.self.notifications` tree

Expected implementation behaviour

- The server (or device) should monitor the current value and compare it to the defined zones.
- If a value enters an alarm zone, then a key is written to `vessels.self.notifications..`
- If a value leaves an alarm zone, then the key is removed from `vessels.self.notifications..`
- Alarms raised are monitored by an alarm process on the server, which takes appropriate action, sounding alarms, or displaying messages.
- Clients interested in alarms can subcribe to the `vessels.self.notifications...` tree in the usual way and be informed of alarms in the same way as normal signalk keys.
- When an alarms is removed, a delta should be sent to subscribers with the path and a null value.

Example

eg If we exceed our anchor alarm radius:
`vessels.self.navigation.anchor.currentRadius` enters `vessels.self.navigation.anchor.currentRadius.meta.zones : [ {lower: "0", upper: maxRadius, state : "normal"}, {lower: maxRadius, upper: 999999, state: "alarm"}]`

The alarm is : `vessels.self.notifications.navigation.anchor.currentRadius`

The alarm object is

```
{
  "value": {
    "method": ["sound"],
    "state": "alarm",
    "message": "Dragging anchor!"
  },
  "timestamp": "...",
  "$source": "..."
}
```

The server alarm manager will see this new entry and turn on the alarm. Using a manager process allows flexibility in situations where multiple alarms are triggered and your vessel is a mass of flashing and beeping. eg A single 'Pause' button can give you 5-10 minutes to take action, stopping annoying noise, and removing popup messages from screens.

Since the `vessels.self.notifications` tree mirrors the other data in the signal k model, we can selectively watch or react to specific branches or keys. When displaying multiple alarms a screen can also sort and filter them.

Other Alarms

Above we have discussed monitoring existing values and raising alarms. There are other alarms that must be considered, eg MOB, fire, sinking etc, and misc alerts "GPS signal lost".etc.

The `vessels.[uuid].notifications` tree is the same as any other Signal k branch. Keys can be added and removed as required in the usual way. Since the branch is being monitored we only need to add a key of any sort to create a suitable alarm.

In the case of an emergency, create a unique key: The alarm is : `vessels.[uuid].notifications.[alarm.key]`

The alarm object is

```
{
  "value": {
    "method": ["visual", "sound"],
    "state": "emergency",
    "message": "Man Overboard!"
  },
  ...
}
```

Alarm objects that have been raised this way must be cleared manually, or by the process that created them. You can use any suitable path, keeping in mind the context of the alarm.

eg In the case of an alert, create a unique key by generating a path: The alarm is : `vessels.[uuid].notifications.navigation.gnss`

The alarm object is

```
{
  "value": {
    "method": ["visual"],
    "state": "alert",
    "message": "GPS signal lost!"
  },
  ...
}
```

Well Known Names

Some alarms are especially important, eg MOB. This is a list of keys for special alarms.

- `..notifications.mob.*`
- `..notifications.fire.*`
- `..notifications.sinking.*`
- `..notifications.flooding.*`
- `..notifications.collision.*`
- `..notifications.grounding.*`
- `..notifications.listing.*`
- `..notifications.adrift.*`
- `..notifications.piracy.*`
- `..notifications.abandon.*`

An example to send an MOB alarm from an N2K source, the gateway would convert and send something like:

```
{
  "context": "vessels.urn:mrn:signalk:uuid:c0d79334-4e25-4245-8892-54e8ccc8021
  "updates": [
    {
      "source": {...},
      "timestamp": "2017-08-15T16:00:05.200Z",
      "values": [
        {
          "path": "notifications.mob",
          "value": {
            "message": "MOB",
            "state": "emergency",
            "method": ["visual", "sound"]
          }
        }
      ]
    }
  ]
}
```

The resulting full signalk tree would be:

```
{
  "vessels": {
    "urn:mrn:signalk:uuid:c0d79334-4e25-4245-8892-54e8ccc8021d": {
      "uuid": "...",
      "notifications": {
        "mob": {
          "value": {
            "method": ["visual", "sound"],
            "state": "emergency",
            "message": "Man Overboard!"
          },
          "timestamp": "2017-04-10T08:33:53Z",
          "$source": "..."
        }
      }
    }
  },
  ...
}
```

To clear the alarm condition, send:

```
{
  "context": "vessels.urn:mrn:signalk:uuid:c0d79334-4e25-4245-8892-54e8ccc8021
  "updates": [
    {
      "source": {...},
      "timestamp": "2017-08-15T16:00:05.538Z",
      "values": [
        {
          "path": "notifications.mob",
          "value": null
        }
      ]
    }
  ]
}
```

Multiple cases of the same alarm

Should multiple cases of the same alarm occur (eg a gps loses signal, then a second gps loses signal) the alarms are handled the same as any other multiple values in signalk. However alarms will tend to be re-issued whenever the underlying data changes.

The servers alarm monitoring processes are expected to be smart enough to know that the anchor alarm is triggered, and its not necessary to raise a second copy of the same alarm, after all there is only one boat dragging!

This may be handled differently for notifications. It may be useful to know that your gps's are all failing intermittently, or that . Hence the handling of multiple copies of alarms is an implementation issue, and may vary.

The key should be unique

If we have an alarm `vessels.self.notifications.navigation.anchor.currentRadius` and we attempt to write another higher in the same tree at `vessels.self.notifications` it must not replace or remove the existing alarm. Since the `meta.zones` structure is only valid on signalk leaf values this occurs naturally in most circumstances. But it is possible to set an alarm value arbitrarily (eg MOB) and care should be taken in implementations that keys do not overwrite existing paths.

Security

Communications Security

For privacy and data integrity REST and WebSockets communications should be secured with Transport Layer Security (TLS). All communications over unsecure protocols like HTTP and WebSockets without TLS must be considered insecure even with authentication and access control mechanisms in place.

Authentication

Authentication for Signal K connections is based on a token carried in the message, or in a cookie or tokens carried in the HTTP `Authorization` header for a HTTP request. The tokens can be of any type.

Note: a Signal K server should never simply echo or redistribute a message received without removing or replacing the token. That would result in A's token being sent to B, which allows the B to impersonate A.

There are 3 authentication actions:

- authenticate - login and obtain a token
- logout - invalidate a token
- validate - validate a token with auto-renewal if valid.

All 3 actions can be done via HTTP requests or by sending Signal K messages for non HTTP clients

Authentication via HTTP

A device or a web client can authenticate with a Signal K server by providing a username and password via a standard HTTP POST request to `/signalk/«version»/auth/login` .

The `«version»` field is the endpoint version identifier chosen by the client from those offered by the server. See the REST API documentation for the structure of these identifiers.

The client may send the login request with a `Content-Type` of `application/json` with the properties `username` and `password` in the body OR with a `Content-Type` of `application/x-www-form-urlencoded` with the `username` and `password` fields.

```
{
  "username": "me@somecompany.com",
  "password": "my password"
}
```

In response to a valid login, the server shall respond with a 200 (OK) status, set an HTTP session cookie and include the token expiry in seconds and the token value in the body of the response. The response `Content-Type` must be `application/json`.

```
{
  "timeToLive": 86400,
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkZXZpY2UiOiIxMjM0LTQ1NjUz"
}
```

In response to invalid login information the server must return HTTP error code 401 (Unauthorized).

If the server does not implement this authentication mechanism it must return HTTP error code 501 (Not Implemented).

Authentication via WebSockets, TCP, and Similar Transports

The client should send a the following message

```
{
  "requestId": "1234-45653-343454",
  "login": {
    "username": "john_doe",
    "password": "password"
  }
}
```

If the login is successful, the server will send a response like the following:

```
{
  "requestId": "1234-45653-343454",
  "state": "COMPLETED",
  "result": 200,
  "login": {
      "timeToLive": 86400,
    "token": "eyJhbGciOiJIUzI1NiIsI...ibtv41fOnJObT4RdOyZ_UI9is8"
  }
}
```

If the login fails, the server will send a response like the following:

```
{
  "requestId": "1234-45653-343454",
  "state": "COMPLETED",
  "result": 401
}
```

The `result` codes are the same as normally used in HTTP.

Providing Authorization to the Server in Subsequent Requests

Web Based Clients

Web based clients should be sure to include the cookie or `Authorization` HTTP header obtained from the authentication response in all subsequent requests.

WebSockets Clients

Clients can include the authentication cookie with the initial request.

Clients can include the `Authorization` HTTP header with the initial connect request. The format of the header should be `Bearer {token}`, for example `Authorization: Bearer eyJhbGciOiJIUzI1NiIsI...ibtv41fOnJObT4RdOyZ_UI9is8`

Other Clients

Clients using other kinds of protocols must include the `token` in the Signal K messages they send.

```
{
  "context": "*",
  "token": "eyJhbGciOiJIUzI1NiIsI...ibtv41fOnJObT4RdOyZ_UI9is8"
  "unsubscribe": [
    {
      "path": "*"
    }
  ]
}
```

Token Validation

Tokens may have a short expiry time and need to be renewed periodically, or a token's current validity may be unknown.

HTTP Clients

To validate a token, a web based client should send an HTTP POST request to `/signalk/«version»/auth/validate` with the token in the cookie, or in the header. If the token is valid, a new token is created with new expiry time, and a new cookie or header set in the response. This effectively renews a token.

The reply message will be returned in any case.

Other Clients

Clients using other kinds of protocols can send the following message.

```
{
  "requestId": "1234-45653-343454",
  "validate": {
    "token": "eyJhbGciOiJIUzI1NiIsI...ibtv41fOnJObT4RdOyZ_UI9is8"
  }
}
```

Reply Messages

Any validation request results in one of the following messages

On success:

```
{
  "requestId": "1234-45653-343454",
  "state": "COMPLETED",
  "result": 200,
  "validate": {
    "token": "eyJhbGciOiJIUzI1NiIsI...ibtv41fOnJObT4RdOyZ_UI9is8"
  }
}
```

On error (`result` could be any HTTP code):

```
{
  "requestId": "1234-45653-343454",
  "state": "COMPLETED",
  "result": 401
}
```

Logout

HTTP Clients

To logout, an http based client should send an HTTP PUT request to `/signalk/«version»/auth/logout` with the token in the cookie or in the HTTP header.

Other Clients

Clients using other kinds of protocols should send the following message.

```
{
  "requestId": "1234-45653-343454",
  "logout": {
    "token": "eyJhbGciOiJIUzI1NiIsI...ibtv41fOnJObT4RdOyZ_UI9is8"
  }
}
```

Reply Messages

In both cases the reply will be

On success:

```
{
  "requestId": "1234-45653-343454",
  "state": "COMPLETED",
  "result": 200
}
```

On error (`result` could be any HTTP code):

```
{
  "requestId": "1234-45653-343454",
  "state": "COMPLETED",
  "result": 401
}
```

Device Access

Devices which don’t have any user interaction such as sensors with no input mechanisms should acquire a token using the Access Requests mechanism.

Request/Response

Requests are used to ask the server to take specific actions. This shows how requests should be formed by the client and how the reponses should look. The details of different request types are defined in other parts of the specification.

WebSockets and Other Full-duplex Protocols

The exact format of the message for a specific request is defined elsewhere in the specification.

A request must include a client generated `requestId`. The `requestId` is a string and it must be a version 4 UUID. It will always be included in any response to the request by the server.

For example, a request to PUT a value:

```
{
  "requestId": "123345-23232-232323",
  "put": {
    "path": "electrical.switches.anchorLight.state",
    "value": 1
  }
}
```

The server will respond with a message which includes the `requestId`, a `state`, and a `statusCode`.

The `state` can be `FAILED`, `PENDING` or `COMPLETED`.

The `statusCode` will be any standard HTTP code including the following.

- 200 - the request was successful
- 400 - something is wrong with the client's request
- 401 - the request has not been applied because it lacks valid authentication credentials
- 403 - the client does not have permission to make the request
- 405 - the server does not support the request
- 502 - something went wrong carrying out the request on the server side
- 504 - timeout on the server side trying to carry out the request

The message can optionally contain an informational, human oriented `message`.

The response object may contain other response data depending on the specific request being made. For example, a response to an authentication request could contain a `login` object.

```
{
  "requestId": "123345-23232-232323",
  "state": "COMPLETED",
  "statusCode": 200,
  "login": {
    "token": "....."
  }
}
```

A server may respond to a request multiple times depending on how it processes the request.

PENDING

When a server cannot process the request immediately, it will respond with the `state` PENDING:

```
{
  "requestId": "123345-23232-232323",
  "state": "PENDING",
  "statusCode": 202
}
```

FAILED

When a server fails read, or process the request (eg a server error), it will respond with the `state` FAILED:

```
{
  "requestId": "123345-23232-232323",
  "state": "FAILED",
  "statusCode": 500
}
```

COMPLETED

When processing is done, but it was not successful:

```
{
  "requestId": "123345-23232-232323",
  "state": "COMPLETED",
  "statusCode": 502,
  "message": "Unable to contact the light"
}
```

When processing completed successfully:

```
{
  "requestId": "123345-23232-232323",
  "state": "COMPLETED",
  "statusCode": 200
}
```

Query a Request

The state of a request can also be found by sending the following:

```
{
  "requestId": "123345-23232-232323",
  "query": true
}
```

This will result in a reply like the examples above.

HTTP

HTTP requests use REST API semantics and the responses are similar to the `response` object used above.

One difference is that the `statusCode` value is also sent as the HTTP response code.

The response when a server successfully processes a login request synchronously:

HTTP response code 200

```
{
  "requestId": "123345-23232-232323",
  "state": "COMPLETED",
  "token": "eyJhbGciOiJIUzI1NiI...aQ8sN1XBAP8bt3tNBT1WiIttm3qM",
  "statusCode": 200
}
```

When a request is PENDING, an HTTP 202 (Accepted) code will be returned and the body will include an `href` to use to check the status of the request. A client should then periodically poll the server to get the status. A client should not poll the server at a rate less than 500ms.

HTTP response code 202

```
{
  "requestId": "123345-23232-232323",
  "state": "PENDING",
  "href": "/signalk/v1/api/actions/12567",
  "statusCode": 202
}
```

The contents of the response message when checking the status will include the values defined above for the `result` object and may also include extra information related to the request.

For example, the result of a PUT request:

```
{
   "requestId": "123345-23232-232323",
   "state": "COMPLETED",
   "statusCode": 200
}
```

PUT Requests

PUT requests are sent to a server to request a change to a value. For example, a client would use PUT to switch the anchor light on or off, change the heading of the autopilot, or set position of the anchor.

See Request/Response for more details on request/response in Signal K.

Making a Request to Change a Value

To change a value, a PUT request should be sent via HTTP or using a Signal K **put** delta.

The `source` field is optional. If a request is sent without the source and there is more than one source for the value, the server should respond with a 400 (Bad Request) HTTP status code.

Via HTTP

```
PUT http://localhost:3000/signalk/v1/api/vessels/self/steering/autopilot/targe
{
  "value": 1.52,
  "source": "actisense.204",
}
```

Via a Delta

~~~
```
[<]: #

The `context` key is optional, and defaults to `vessels.self`, which is the us

#### NOTE ####
The above PUT request (v1) uses an array to allow multiple keys in a single PU
with the request/response semantics in cases of partial failures.  An alternat

[>]: # (mdpInsert ```json fsnip ../data/put-valid/delta-put-no-array.json)
```json
~~~

In the v2 API the array format will be removed. Implementors are recommended to support both in the interim.

Return states

A PUT request in the array format is only successful if *ALL* if the items are successful. It is up to the client to ascertain which were in error, and why.

This is a quick start for any-one that would like to contribute. Its roughly from technically unskilled to skilled, top to bottom. Dont be afraid to ask for help. Each task will probably start with a new thread for more details on the Google groups (https://groups.google.com/forum/#!forum/signalk). Be patient, civil, and persistent :-)

Completely unskilled at boat electronics:

- Join https://groups.google.com/forum/#!forum/signalk - as the user base grows, so does awareness.
- Tell others, spread the word
- Fly a Signal K flag from your boat
- If you have special skills (eg motors, batteries, navigation, etc) help us extend the Signal K protocol by identifying what we need to cover.
- Ask manufacturers about Signal K support
- Ask questions about what you dont understand, and collate the answers for us to put on the website.

Can do own installs, handyman, but not IT skilled.

- Try an install of Raspberry Pi and WIFI, document exactly how you did it, so others can follow.

Website or documentation skills

- Help us maintain the website, and improve the documents

Good computer skills, but not programming

- Download and try the java server (https://github.com/SignalK/signalk-server-java) and node server (https://github.com/SignalK/signalk-server-node) and the various apps and clients. Help test and identify issues, help improve documents so others can follow easier.
- Help with User manuals!

Systems engineer

- Help other users, help with scripts, develop and maintain install processes, managing our web sites, etc.
- Examples:
  - Create Debian packages of the Signal K software for easy installation to Raspbian

Software developer

- Download and test/fix our stuff, add improvements, join the team and help code, develop support in your own software.

Microprocessors

- Improve our Arduino stuff, add your own, incorporate Signal K into your products.

/vessels

Description: A wrapper object for vessel objects, each describing vessels in range, including this vessel.

/vessels/<RegExp>

Title: vessel

Description: This regex pattern is used for validation of an MMSI or Signal K UUID identifier for the vessel. Examples: urn:mrn:imo:mmsi:230099999 urn:mrn:signalk:uuid:c0d79334-4e25-4245-8892-54e8ccc8021d

---

/vessels/<RegExp>/url

Description: URL based identity of the vessel, if available.

/vessels/<RegExp>/mmsi

Description: MMSI number of the vessel, if available.

/vessels/<RegExp>/mothershipMmsi

Description: MMSI number of the mothership of this vessel, if available.

/vessels/<RegExp>/uuid

Description: A unique Signal K flavoured maritime resource identifier, assigned by the server.

/vessels/<RegExp>/name

Description: The common name of the vessel

/vessels/<RegExp>/flag

Description: The country of ship registration, or flag state of the vessel

---

/vessels/<RegExp>/port

Description: The home port of the vessel

/vessels/<RegExp>/registrations

Description: The various registrations of the vessel.

/vessels/<RegExp>/registrations/imo

Description: The IMO number of the vessel.

/vessels/<RegExp>/registrations/national

Description: The national registration number of the vessel.

/vessels/<RegExp>/registrations/national/<RegExp>

Description: This regex pattern is used for validating the identifier for the registration

/vessels/<RegExp>/registrations/national/<RegExp>/country

Description: The ISO 3166-2 country code.

/vessels/<RegExp>/registrations/national/<RegExp>/registration

Description: The registration code

/vessels/<RegExp>/registrations/national/<RegExp>/description

Description: The registration description

/vessels/<RegExp>/registrations/local

Description: A local or state registration number of the vessel.

/vessels/<RegExp>/registrations/local/<RegExp>

Description: This regex pattern is used for validating the identifier for the registration

---

/vessels/<RegExp>/registrations/local/<RegExp>/registration

Description: The registration code

/vessels/<RegExp>/registrations/local/<RegExp>/description

Description: The registration description

/vessels/<RegExp>/registrations/other

Description: Other registration or permits for the vessel.

/vessels/<RegExp>/registrations/other/<RegExp>

Description: This regex pattern is used for validating the identifier for the registration

---

/vessels/<RegExp>/registrations/other/<RegExp>/registration

Description: The registration code

/vessels/<RegExp>/registrations/other/<RegExp>/description

Description: The registration description

/vessels/<RegExp>/communication

Title: communication

Description: Communication data including Radio, Telephone, E-Mail, etc.

/vessels/<RegExp>/communication/callsignVhf

Description: Callsign for VHF communication

/vessels/<RegExp>/communication/callsignHf

Description: Callsign for HF communication

/vessels/<RegExp>/communication/phoneNumber

Description: Phone number of skipper

/vessels/<RegExp>/communication/emailHf

Description: Email address to be used for HF email (Winmail, Airmail, Sailmail)

/vessels/<RegExp>/communication/email

Description: Regular email for the skipper

/vessels/<RegExp>/communication/satPhoneNumber

Description: Satellite phone number for vessel.

/vessels/<RegExp>/communication/skipperName

Description: Full name of the skipper of the vessel.

/vessels/<RegExp>/communication/crewNames

Description: Array with the names of the crew

/vessels/<RegExp>/environment

Title: environment

Description: Environmental data measured locally including Depth, Wind, Temp, etc.

/vessels/<RegExp>/environment/outside

Description: Environmental conditions outside of the vessel's hull

---

/vessels/<RegExp>/environment/outside/temperature

Units: K (Kelvin)

Description: Current outside air temperature

/vessels/<RegExp>/environment/outside/dewPointTemperature

Units: K (Kelvin)

Description: Current outside dew point temperature

/vessels/<RegExp>/environment/outside/apparentWindChillTemperature

Units: K (Kelvin)

Description: Current outside apparent wind chill temperature

/vessels/<RegExp>/environment/outside/theoreticalWindChillTemperature

Units: K (Kelvin)

Description: Current outside theoretical wind chill temperature

---

/vessels/<RegExp>/environment/outside/heatIndexTemperature

Units: K (Kelvin)

Description: Current outside heat index temperature

/vessels/<RegExp>/environment/outside/pressure

Units: Pa (Pascal)

Description: Current outside air ambient pressure

/vessels/<RegExp>/environment/outside/humidity

Units: ratio (Ratio)

Description: Current outside air relative humidity

/vessels/<RegExp>/environment/outside/airDensity

Units: kg/m3 (undefined)

Description: Current outside air density

/vessels/<RegExp>/environment/outside/illuminance

Units: Lux (undefined)

Description: Current outside ambient light flux.

/vessels/<RegExp>/environment/inside

Description: Environmental conditions inside the vessel's hull

---

/vessels/<RegExp>/environment/inside/temperature

Units: K (Kelvin)

Description: Temperature

/vessels/<RegExp>/environment/inside/heatIndexTemperature

Units: K (Kelvin)

Description: Current heat index temperature in zone

/vessels/<RegExp>/environment/inside/pressure

Units: Pa (Pascal)

Description: Pressure in zone

/vessels/<RegExp>/environment/inside/relativeHumidity

Units: ratio (Ratio)

Description: Relative humidity in zone

/vessels/<RegExp>/environment/inside/dewPoint

Units: K (Kelvin)

Description: Dewpoint in zone

/vessels/<RegExp>/environment/inside/airDensity

Units: kg/m3 (undefined)

Description: Air density in zone

/vessels/<RegExp>/environment/inside/illuminance

Units: Lux (undefined)

Description: Illuminance in zone

/vessels/<RegExp>/environment/inside/[A-Za-z0-9]+

Description: This regex pattern is used for validation of the identifier for the environmental zone, eg. engineRoom, mainCabin, refrigerator

---

/vessels/<RegExp>/environment/inside/[A-Za-z0-9]+/temperature

Units: K (Kelvin)

Description: Temperature

/vessels/<RegExp>/environment/inside/[A-Za-z0-9]+/heatIndexTemperature

Units: K (Kelvin)

Description: Current heat index temperature in zone

/vessels/<RegExp>/environment/inside/[A-Za-z0-9]+/pressure

Units: Pa (Pascal)

Description: Pressure in zone

/vessels/<RegExp>/environment/inside/[A-Za-z0-9]+/relativeHumidity

Units: ratio (Ratio)

Description: Relative humidity in zone

/vessels/<RegExp>/environment/inside/[A-Za-z0-9]+/dewPoint

Units: K (Kelvin)

Description: Dewpoint in zone

/vessels/<RegExp>/environment/inside/[A-Za-z0-9]+/airDensity

Units: kg/m3 (undefined)

Description: Air density in zone

/vessels/<RegExp>/environment/inside/[A-Za-z0-9]+/illuminance

Units: Lux (undefined)

Description: Illuminance in zone

/vessels/<RegExp>/environment/water

Description: Environmental conditions of the water that the vessel is sailing in

/vessels/<RegExp>/environment/water/temperature

Units: K (Kelvin)

Description: Current water temperature

/vessels/<RegExp>/environment/water/salinity

Units: ratio (Ratio)

Description: Water salinity

/vessels/<RegExp>/environment/depth

Title: depth

Description: Depth related data

/vessels/<RegExp>/environment/depth/belowKeel

Units: m (Meter)

Description: Depth below keel

/vessels/<RegExp>/environment/depth/belowTransducer

Units: m (Meter)

Description: Depth below Transducer

/vessels/<RegExp>/environment/depth/belowSurface

Units: m (Meter)

Description: Depth from surface

/vessels/<RegExp>/environment/depth/transducerToKeel

Units: m (Meter)

Description: Depth from the transducer to the bottom of the keel

---

/vessels/<RegExp>/environment/depth/surfaceToTransducer

Units: m (Meter)

Description: Depth transducer is below the water surface

/vessels/<RegExp>/environment/current

Title: current

Description: Direction and strength of current affecting the vessel

Object value with properties

- drift (m/s)
- setTrue (rad)
- setMagnetic (rad)

/vessels/<RegExp>/environment/tide

Title: tide

Description: Tide data

/vessels/<RegExp>/environment/tide/heightHigh

Units: m (Meter)

Description: Next high tide height relative to lowest astronomical tide (LAT/Chart Datum)

/vessels/<RegExp>/environment/tide/heightNow

Units: m (Meter)

Description: The current tide height relative to lowest astronomical tide (LAT/Chart Datum)

/vessels/<RegExp>/environment/tide/heightLow

Units: m (Meter)

Description: The next low tide height relative to lowest astronomical tide (LAT/Chart Datum)

/vessels/<RegExp>/environment/tide/timeLow

Units: RFC 3339 (UTC) (undefined)

Description: Time of the next low tide in UTC

/vessels/<RegExp>/environment/tide/timeHigh

Units: RFC 3339 (UTC) (undefined)

Description: Time of next high tide in UTC

/vessels/<RegExp>/environment/heave

Units: m (Meter)

Description: Vertical movement of the vessel due to waves

/vessels/<RegExp>/environment/wind

Title: wind

Description: Wind data.

/vessels/<RegExp>/environment/wind/angleApparent

Units: rad (Radian)

Description: Apparent wind angle, negative to port

/vessels/<RegExp>/environment/wind/angleTrueGround

Units: rad (Radian)

Description: True wind angle based on speed over ground, negative to port

---

/vessels/<RegExp>/environment/wind/angleTrueWater

Units: rad (Radian)

Description: True wind angle based on speed through water, negative to port

---

/vessels/<RegExp>/environment/wind/directionChangeAlarm

Units: rad (Radian)

Description: The angle the wind needs to shift to raise an alarm

---

/vessels/<RegExp>/environment/wind/directionTrue

Units: rad (Radian)

Description: The wind direction relative to true north

/vessels/<RegExp>/environment/wind/directionMagnetic

Units: rad (Radian)

Description: The wind direction relative to magnetic north

/vessels/<RegExp>/environment/wind/speedTrue

Units: m/s (Meters per second)

Description: Wind speed over water (as calculated from speedApparent and vessel's speed through water)

/vessels/<RegExp>/environment/wind/speedOverGround

Units: m/s (Meters per second)

Description: Wind speed over ground (as calculated from speedApparent and vessel's speed over ground)

/vessels/<RegExp>/environment/wind/speedApparent

Units: m/s (Meters per second)

Description: Apparent wind speed

/vessels/<RegExp>/environment/time

Description: A time reference for the vessel. All clocks on the vessel dispaying local time should use the timezone offset here. If a timezoneRegion is supplied the timezone must also be supplied. If timezoneRegion is supplied that should be displayed by UIs in preference to simply timezone. ie 12:05 (Europe/London) should be displayed in preference to 12:05 (UTC+01:00)

/vessels/<RegExp>/environment/mode

Description: Mode of the vessel based on the current conditions. Can be combined with navigation.state to control vessel signals eg switch to night mode for instrumentation and lights, or make sound signals for fog.

/vessels/<RegExp>/navigation

Title: navigation

Description: Navigation data including Position, Course to next WP information, etc.

/vessels/<RegExp>/navigation/lights

Title: Navigation lights

Description: Current state of the vessels navigation lights

/vessels/<RegExp>/navigation/courseOverGroundMagnetic

Units: rad (Radian)

Description: Course over ground (magnetic)

/vessels/<RegExp>/navigation/courseOverGroundTrue

Units: rad (Radian)

Description: Course over ground (true)

/vessels/<RegExp>/navigation/courseRhumbline

Title: Course

Description: Course information computed with Rhumbline

/vessels/<RegExp>/navigation/courseRhumbline/crossTrackError

Units: m (Meter)

Description: The distance from the vessel's present position to the closest point on a line (track) between previousPoint and nextPoint. A negative number indicates that the vessel is currently to the left of this line (and thus must steer right to compensate), a positive number means the vessel is to the right of the line (steer left to compensate).

/vessels/<RegExp>/navigation/courseRhumbline/bearingTrackTrue

Units: rad (Radian)

Description: The bearing of a line between previousPoint and nextPoint, relative to true north.

/vessels/<RegExp>/navigation/courseRhumbline/bearingTrackMagnetic

Units: rad (Radian)

Description: The bearing of a line between previousPoint and nextPoint, relative to magnetic north.

/vessels/<RegExp>/navigation/courseRhumbline/activeRoute

Description: Data required if sailing to an active route, defined in resources.

/vessels/<RegExp>/navigation/courseRhumbline/activeRoute/href

Description: A reference (URL) to the presently active route, in resources.

/vessels/<RegExp>/navigation/courseRhumbline/activeRoute/estimatedTimeOfArrival

Description: The estimated time of arrival at the end of the current route

---

/vessels/<RegExp>/navigation/courseRhumbline/activeRoute/startTime

Description: The time this route was activated

/vessels/<RegExp>/navigation/courseRhumbline/nextPoint

Description: The point on earth the vessel's presently navigating towards

---

/vessels/<RegExp>/navigation/courseRhumbline/previousPoint

Description: The point on earth the vessel's presently navigating from

Object value with properties

- type
- href

/vessels/<RegExp>/navigation/courseRhumbline/previousPoint/distance

Units: m (Meter)

Description: The distance in meters between previousPoint and the vessel's present position

---

/vessels/<RegExp>/navigation/courseRhumbline/previousPoint/position

Title: position

Description: The position of lastPoint in two dimensions

Object value with properties

- longitude (deg)
- latitude (deg)
- altitude (m)

/vessels/<RegExp>/navigation/courseGreatCircle

Title: Course

Description: Course information computed with Great Circle

/vessels/<RegExp>/navigation/courseGreatCircle/crossTrackError

Units: m (Meter)

Description: The distance from the vessel's present position to the closest point on a line (track) between previousPoint and nextPoint. A negative number indicates that the vessel is currently to the left of this line (and thus must steer right to compensate), a positive number means the vessel is to the right of the line (steer left to compensate).

/vessels/<RegExp>/navigation/courseGreatCircle/bearingTrackTrue

Units: rad (Radian)

Description: The bearing of a line between previousPoint and nextPoint, relative to true north.

/vessels/<RegExp>/navigation/courseGreatCircle/bearingTrackMagnetic

Units: rad (Radian)

Description: The bearing of a line between previousPoint and nextPoint, relative to magnetic north.

/vessels/<RegExp>/navigation/courseGreatCircle/activeRoute

Description: Data required if sailing to an active route, defined in resources.

/vessels/<RegExp>/navigation/courseGreatCircle/activeRoute/href

Description: A reference (URL) to the presently active route, in resources.

/vessels/<RegExp>/navigation/courseGreatCircle/activeRoute/estimatedTimeOfArrival

Description: The estimated time of arrival at the end of the current route

/vessels/<RegExp>/navigation/courseGreatCircle/activeRoute/startTime

Description: The time this route was activated

/vessels/<RegExp>/navigation/courseGreatCircle/nextPoint

Description: The point on earth the vessel's presently navigating towards

/vessels/<RegExp>/navigation/courseGreatCircle/previousPoint

Description: The point on earth the vessel's presently navigating from

Object value with properties

- type
- href

/vessels/<RegExp>/navigation/courseGreatCircle/previousPoint/distance

Units: m (Meter)

Description: The distance in meters between previousPoint and the vessel's present position

---

/vessels/<RegExp>/navigation/courseGreatCircle/previousPoint/position

Title: position

Description: The position of lastPoint in two dimensions

Object value with properties

- longitude (deg)
- latitude (deg)
- altitude (m)

/vessels/<RegExp>/navigation/closestApproach

Description: Calculated values for other vessels, e.g. from AIS

Object value with properties

- distance (m)
- timeTo (s)

/vessels/<RegExp>/navigation/racing

Description: Specific navigational data related to yacht racing.

/vessels/<RegExp>/navigation/racing/startLineStb

Title: position

Description: Position of starboard start mark

Object value with properties

- longitude (deg)
- latitude (deg)
- altitude (m)

/vessels/<RegExp>/navigation/racing/startLinePort

Title: position

Description: Position of port start mark

Object value with properties

- longitude (deg)
- latitude (deg)
- altitude (m)

/vessels/<RegExp>/navigation/racing/distanceStartline

Units: m (Meter)

Description: The current distance to the start line

/vessels/<RegExp>/navigation/racing/timeToStart

Units: s (Second)

Description: Time left before start

/vessels/<RegExp>/navigation/racing/timePortDown

Units: s (Second)

Description: Time to arrive at the start line on port, turning downwind

---

/vessels/<RegExp>/navigation/racing/timePortUp

Units: s (Second)

Description: Time to arrive at the start line on port, turning upwind

---

/vessels/<RegExp>/navigation/racing/timeStbdDown

Units: s (Second)

Description: Time to arrive at the start line on starboard, turning downwind

---

/vessels/<RegExp>/navigation/racing/timeStbdUp

Units: s (Second)

Description: Time to arrive at the start line on starboard, turning upwind

---

/vessels/<RegExp>/navigation/racing/layline

Description: The layline crossing the current course

/vessels/<RegExp>/navigation/racing/layline/distance

Units: m (Meter)

Description: The current distance to the layline

/vessels/<RegExp>/navigation/racing/layline/time

Units: s (Second)

Description: The time to the layline at current speed and heading

---

/vessels/<RegExp>/navigation/racing/oppositeLayline

Description: The layline parallell to current course

/vessels/<RegExp>/navigation/racing/oppositeLayline/distance

Units: m (Meter)

Description: The current distance to the layline

/vessels/<RegExp>/navigation/racing/oppositeLayline/time

Units: s (Second)

Description: The time to the layline at current speed and heading

---

/vessels/<RegExp>/navigation/magneticVariation

Units: rad (Radian)

Description: The magnetic variation (declination) at the current position that must be added to the magnetic heading to derive the true heading. Easterly variations are positive and Westerly variations are negative (in Radians).

/vessels/<RegExp>/navigation/magneticVariationAgeOfService

Units: s (Second)

Description: Seconds since the 1st Jan 1970 that the variation calculation was made

/vessels/<RegExp>/navigation/destination

Title: destination

Description: The intended destination of this trip

/vessels/<RegExp>/navigation/destination/commonName

Description: Common name of the Destination, eg 'Fiji', also used in ais messages

/vessels/<RegExp>/navigation/destination/eta

Description: Expected time of arrival at destination waypoint

/vessels/<RegExp>/navigation/destination/waypoint

Description: UUID of destination waypoint

/vessels/<RegExp>/navigation/gnss

Title: gnss

Description: Global satellite navigation meta information

/vessels/<RegExp>/navigation/gnss/type

Description: Fix type

/vessels/<RegExp>/navigation/gnss/methodQuality

Description: Quality of the satellite fix

/vessels/<RegExp>/navigation/gnss/integrity

Description: Integrity of the satellite fix

/vessels/<RegExp>/navigation/gnss/satellites

Description: Number of satellites

/vessels/<RegExp>/navigation/gnss/antennaAltitude

Units: m (Meter)

Description: Altitude of antenna

/vessels/<RegExp>/navigation/gnss/horizontalDilution

Description: Horizontal Dilution of Precision

/vessels/<RegExp>/navigation/gnss/positionDilution

Description: Positional Dilution of Precision

/vessels/<RegExp>/navigation/gnss/geoidalSeparation

Description: Difference between WGS84 earth ellipsoid and mean sea level

---

/vessels/<RegExp>/navigation/gnss/differentialAge

Units: s (Second)

Description: Age of DGPS data

/vessels/<RegExp>/navigation/gnss/differentialReference

Description: ID of DGPS base station

/vessels/<RegExp>/navigation/headingMagnetic

Units: rad (Radian)

Description: Current magnetic heading of the vessel, equals 'headingCompass adjusted for magneticDeviation'

/vessels/<RegExp>/navigation/magneticDeviation

Units: rad (Radian)

Description: Magnetic deviation of the compass at the current headingCompass

---

/vessels/<RegExp>/navigation/headingCompass

Units: rad (Radian)

Description: Current magnetic heading received from the compass. This is not adjusted for magneticDeviation of the compass

---

/vessels/<RegExp>/navigation/headingTrue

Units: rad (Radian)

Description: The current true north heading of the vessel, equals 'headingMagnetic adjusted for magneticVariation'

/vessels/<RegExp>/navigation/position

Title: position

Description: The position of the vessel in 2 or 3 dimensions (WGS84 datum)

Object value with properties

- longitude (deg)
- latitude (deg)
- altitude (m)

/vessels/<RegExp>/navigation/attitude

Title: Attitude

Description: Vessel attitude: roll, pitch and yaw

Object value with properties

- roll (rad)
- pitch (rad)
- yaw (rad)

/vessels/<RegExp>/navigation/maneuver

Description: Special maneuver such as regional passing arrangement. (from ais)

/vessels/<RegExp>/navigation/rateOfTurn

Units: rad/s (Radian per second)

Description: Rate of turn (+ve is change to starboard). If the value is AIS RIGHT or LEFT, set to +-0.0206 rads and add warning in notifications

---

/vessels/<RegExp>/navigation/speedOverGround

Units: m/s (Meters per second)

Description: Vessel speed over ground. If converting from AIS 'HIGH' value, set to 102.2 (Ais max value) and add warning in notifications

---

/vessels/<RegExp>/navigation/speedThroughWater

Units: m/s (Meters per second)

Description: Vessel speed through the water

/vessels/<RegExp>/navigation/speedThroughWaterTransverse

Units: m/s (Meters per second)

Description: Transverse speed through the water (Leeway)

/vessels/<RegExp>/navigation/speedThroughWaterLongitudinal

Units: m/s (Meters per second)

Description: Longitudinal speed through the water

/vessels/<RegExp>/navigation/leewayAngle

Units: rad (Radian)

Description: Leeway Angle derived from the longitudinal and transverse speeds through the water

---

/vessels/<RegExp>/navigation/log

Units: m (Meter)

Description: Total distance traveled

/vessels/<RegExp>/navigation/trip

Description: Trip data

/vessels/<RegExp>/navigation/trip/log

Units: m (Meter)

Description: Total distance traveled on this trip / since trip reset

---

/vessels/<RegExp>/navigation/trip/lastReset

Description: Trip log reset time

/vessels/<RegExp>/navigation/state

Title: state

Description: Current navigational state of the vessel

/vessels/<RegExp>/navigation/anchor

Title: anchor

Description: The anchor data, for anchor watch etc

/vessels/<RegExp>/navigation/anchor/maxRadius

Units: m (Meter)

Description: Radius of anchor alarm boundary. The distance from anchor to the center of the boat

---

/vessels/<RegExp>/navigation/anchor/currentRadius

Units: m (Meter)

Description: Current distance to anchor

/vessels/<RegExp>/navigation/anchor/position

Title: position

Description: The actual anchor position of the vessel in 3 dimensions, probably an estimate at best

Object value with properties

- longitude (deg)
- latitude (deg)
- altitude (m)

/vessels/<RegExp>/navigation/datetime

Description: Time and Date from the GNSS Positioning System

/vessels/<RegExp>/propulsion

Title: propulsion

Description: Engine data, each engine identified by a unique name i.e. Port_Engine

---

/vessels/<RegExp>/propulsion/<RegExp>

Description: This regex pattern is used for validation of the identifier for the propulsion unit

---

/vessels/<RegExp>/propulsion/<RegExp>/label

Description: Human readable label for the propulsion unit

/vessels/<RegExp>/propulsion/<RegExp>/state

Description: The current state of the engine

/vessels/<RegExp>/propulsion/<RegExp>/revolutions

Units: Hz (Hertz)

Description: Engine revolutions (x60 for RPM)

/vessels/<RegExp>/propulsion/<RegExp>/temperature

Units: K (Kelvin)

Description: Engine temperature

/vessels/<RegExp>/propulsion/<RegExp>/oilTemperature

Units: K (Kelvin)

Description: Oil temperature

/vessels/<RegExp>/propulsion/<RegExp>/oilPressure

Units: Pa (Pascal)

Description: Oil pressure

/vessels/<RegExp>/propulsion/<RegExp>/alternatorVoltage

Units: V (Volt)

Description: Alternator voltage

/vessels/<RegExp>/propulsion/<RegExp>/runTime

Units: s (Second)

Description: Total running time for engine (Engine Hours in seconds)

/vessels/<RegExp>/propulsion/<RegExp>/coolantTemperature

Units: K (Kelvin)

Description: Coolant temperature

/vessels/<RegExp>/propulsion/<RegExp>/coolantPressure

Units: Pa (Pascal)

Description: Coolant pressure

/vessels/<RegExp>/propulsion/<RegExp>/boostPressure

Units: Pa (Pascal)

Description: Engine boost (turbo, supercharger) pressure

/vessels/<RegExp>/propulsion/<RegExp>/intakeManifoldTemperature

Units: K (Kelvin)

Description: Intake manifold temperature

/vessels/<RegExp>/propulsion/<RegExp>/engineLoad

Units: ratio (Ratio)

Description: Engine load ratio, 0<=ratio<=1, 1 is 100%

/vessels/<RegExp>/propulsion/<RegExp>/engineTorque

Units: ratio (Ratio)

Description: Engine torque ratio, 0<=ratio<=1, 1 is 100%

/vessels/<RegExp>/propulsion/<RegExp>/transmission

Description: The transmission (gear box) of the named engine

/vessels/<RegExp>/propulsion/<RegExp>/transmission/gear

Description: Currently selected gear the engine is in i.e. Forward, Reverse, etc.

/vessels/<RegExp>/propulsion/<RegExp>/transmission/gearRatio

Units: ratio (Ratio)

Description: Gear ratio, engine rotations per propeller shaft rotation

/vessels/<RegExp>/propulsion/<RegExp>/transmission/oilTemperature

Units: K (Kelvin)

Description: Oil temperature

/vessels/<RegExp>/propulsion/<RegExp>/transmission/oilPressure

Units: Pa (Pascal)

Description: Oil pressure

/vessels/<RegExp>/propulsion/<RegExp>/drive

Description: Data about the engine's drive.

/vessels/<RegExp>/propulsion/<RegExp>/drive/type

Description: The type of drive the boat has i.e Outboard, shaft, jet, etc.

Enum values:

- saildrive
- shaft
- outboard
- jet
- pod
- other

/vessels/<RegExp>/propulsion/<RegExp>/drive/trimState

Units: ratio (Ratio)

Description: Trim/tilt state, 0<=ratio<=1, 1 is 100% up

/vessels/<RegExp>/propulsion/<RegExp>/drive/thrustAngle

Units: rad (Radian)

Description: Current thrust angle for steerable drives, +ve is thrust to Starboard

---

/vessels/<RegExp>/propulsion/<RegExp>/drive/propeller

Description: Data about the drive's propeller (pitch and slip)

/vessels/<RegExp>/propulsion/<RegExp>/fuel

Description: Data about the engine's Fuel Supply

/vessels/<RegExp>/propulsion/<RegExp>/fuel/type

Description: Fuel type

Enum values:

- diesel
- petrol
- electric
- coal/wood
- other

/vessels/<RegExp>/propulsion/<RegExp>/fuel/used

Units: m3 (Cubic meter)

Description: Used fuel since last reset. Resetting is at user discretion

---

/vessels/<RegExp>/propulsion/<RegExp>/fuel/pressure

Units: Pa (Pascal)

Description: Fuel pressure

/vessels/<RegExp>/propulsion/<RegExp>/fuel/rate

Units: m3/s (Cubic meter per second)

Description: Fuel rate of consumption

/vessels/<RegExp>/propulsion/<RegExp>/fuel/economyRate

Units: m3/s (Cubic meter per second)

Description: Economy fuel rate of consumption

/vessels/<RegExp>/propulsion/<RegExp>/fuel/averageRate

Units: m3/s (Cubic meter per second)

Description: Average fuel rate of consumption

/vessels/<RegExp>/propulsion/<RegExp>/exhaustTemperature

Units: K (Kelvin)

Description: Exhaust temperature

/vessels/<RegExp>/electrical

Title: electrical

Description: Electrical data, each electrical device indentified by a unique name i.e. Battery_1

/vessels/<RegExp>/electrical/batteries

Description: Data about the vessel's batteries

/vessels/<RegExp>/electrical/batteries/<RegExp>

Title: Battery keyed by instance id

Description: Batteries, one or many, within the vessel

/vessels/<RegExp>/electrical/batteries/<RegExp>/name

Description: Unique ID of device (houseBattery, alternator, Generator, solar1, inverter, charger, combiner, etc.)

/vessels/<RegExp>/electrical/batteries/<RegExp>/location

Description: Installed location of device on vessel

/vessels/<RegExp>/electrical/batteries/<RegExp>/dateInstalled

Units: RFC 3339 (UTC) (undefined)

Description: Date device was installed

/vessels/<RegExp>/electrical/batteries/<RegExp>/manufacturer

Description: [missing]

/vessels/<RegExp>/electrical/batteries/<RegExp>/manufacturer/name

Description: Manufacturer's name

/vessels/<RegExp>/electrical/batteries/<RegExp>/manufacturer/model

Description: Model or part number

/vessels/<RegExp>/electrical/batteries/<RegExp>/manufacturer/URL

Description: Web referance / URL

/vessels/<RegExp>/electrical/batteries/<RegExp>/associatedBus

Description: Name of BUS device is associated with

/vessels/<RegExp>/electrical/batteries/<RegExp>/voltage

Units: V (Volt)

Description: Voltage measured at or as close as possible to the device

/vessels/<RegExp>/electrical/batteries/<RegExp>/voltage/ripple

Units: V (Volt)

Description: DC Ripple voltage

/vessels/<RegExp>/electrical/batteries/<RegExp>/current

Units: A (Ampere)

Description: Current flowing out (+ve) or in (-ve) to the device

/vessels/<RegExp>/electrical/batteries/<RegExp>/temperature

Title: temperature

Units: K (Kelvin)

Description: Temperature measured within or on the device

/vessels/<RegExp>/electrical/batteries/<RegExp>/chemistry

Description: Type of battery FLA, LiFePO4, etc.

/vessels/<RegExp>/electrical/batteries/<RegExp>/capacity

Title: capacity

Description: Data about the battery's capacity

/vessels/<RegExp>/electrical/batteries/<RegExp>/capacity/nominal

Units: J (Joule)

Description: The capacity of battery as specified by the manufacturer

---

/vessels/<RegExp>/electrical/batteries/<RegExp>/capacity/actual

Units: J (Joule)

Description: The measured capacity of battery. This may change over time and will likely deviate from the nominal capacity.

/vessels/<RegExp>/electrical/batteries/<RegExp>/capacity/remaining

Units: J (Joule)

Description: Capacity remaining in battery

/vessels/<RegExp>/electrical/batteries/<RegExp>/capacity/dischargeLimit

Units: J (Joule)

Description: Minimum capacity to be left in the battery while discharging

---

/vessels/<RegExp>/electrical/batteries/<RegExp>/capacity/stateOfCharge

Units: ratio (Ratio)

Description: State of charge, 1 = 100%

/vessels/<RegExp>/electrical/batteries/<RegExp>/capacity/stateOfHealth

Units: ratio (Ratio)

Description: State of Health, 1 = 100%

/vessels/<RegExp>/electrical/batteries/<RegExp>/capacity/dischargeSinceFull

Units: C (Coulomb)

Description: Cumulative discharge since battery was last full

---

/vessels/<RegExp>/electrical/batteries/<RegExp>/capacity/timeRemaining

Units: s (Second)

Description: Time to discharge to discharge limit at current rate

---

/vessels/<RegExp>/electrical/batteries/<RegExp>/lifetimeDischarge

Units: C (Coulomb)

Description: Cumulative charge discharged from battery over operational lifetime of battery

---

/vessels/<RegExp>/electrical/batteries/<RegExp>/lifetimeRecharge

Units: C (Coulomb)

Description: Cumulative charge recharged into battery over operational lifetime of battery

---

/vessels/<RegExp>/electrical/inverters

Description: Data about the Inverter that has both DC and AC qualities

---

/vessels/<RegExp>/electrical/inverters/<RegExp>

Title: Inverter

Description: DC to AC inverter, one or many, within the vessel

---

/vessels/<RegExp>/electrical/inverters/<RegExp>/name

Description: Unique ID of device (houseBattery, alternator, Generator, solar1, inverter, charger, combiner, etc.)

/vessels/<RegExp>/electrical/inverters/<RegExp>/location

Description: Installed location of device on vessel

/vessels/<RegExp>/electrical/inverters/<RegExp>/dateInstalled

Units: RFC 3339 (UTC) (undefined)

Description: Date device was installed

/vessels/<RegExp>/electrical/inverters/<RegExp>/manufacturer

Description: [missing]

/vessels/<RegExp>/electrical/inverters/<RegExp>/manufacturer/name

Description: Manufacturer's name

/vessels/<RegExp>/electrical/inverters/<RegExp>/manufacturer/model

Description: Model or part number

/vessels/<RegExp>/electrical/inverters/<RegExp>/manufacturer/URL

Description: Web referance / URL

/vessels/<RegExp>/electrical/inverters/<RegExp>/dc

Title: DC Qualities

Description: DC common qualities

/vessels/<RegExp>/electrical/inverters/<RegExp>/dc/associatedBus

Description: Name of BUS device is associated with

/vessels/<RegExp>/electrical/inverters/<RegExp>/dc/voltage

Units: V (Volt)

Description: Voltage measured at or as close as possible to the device

---

/vessels/<RegExp>/electrical/inverters/<RegExp>/dc/voltage/ripple

Units: V (Volt)

Description: DC Ripple voltage

/vessels/<RegExp>/electrical/inverters/<RegExp>/dc/current

Units: A (Ampere)

Description: Current flowing out (+ve) or in (-ve) to the device

---

/vessels/<RegExp>/electrical/inverters/<RegExp>/dc/temperature

Title: temperature

Units: K (Kelvin)

Description: Temperature measured within or on the device

/vessels/<RegExp>/electrical/inverters/<RegExp>/ac

Title: AC Qualities

Description: AC equipment common qualities

/vessels/<RegExp>/electrical/inverters/<RegExp>/ac/associatedBus

Description: Name of BUS device is associated with

/vessels/<RegExp>/electrical/inverters/<RegExp>/ac/lineNeutralVoltage

Units: V (Volt)

Description: RMS voltage measured between phase and neutral

/vessels/<RegExp>/electrical/inverters/<RegExp>/ac/lineLineVoltage

Units: V (Volt)

Description: RMS voltage measured between phases

/vessels/<RegExp>/electrical/inverters/<RegExp>/ac/current

Units: A (Ampere)

Description: RMS current

/vessels/<RegExp>/electrical/inverters/<RegExp>/ac/frequency

Units: Hz (Hertz)

Description: AC frequency.

/vessels/<RegExp>/electrical/inverters/<RegExp>/ac/reactivePower

Units: W (Watt)

Description: Reactive power

/vessels/<RegExp>/electrical/inverters/<RegExp>/ac/powerFactor

Description: Power factor

/vessels/<RegExp>/electrical/inverters/<RegExp>/ac/powerFactorLagging

Description: Lead/lag status.

Enum values:

- leading
- lagging
- error
- not available

/vessels/<RegExp>/electrical/inverters/<RegExp>/ac/realPower

Units: W (Watt)

Description: Real power.

/vessels/<RegExp>/electrical/inverters/<RegExp>/ac/apparentPower

Units: W (Watt)

Description: Apparent power.

/vessels/<RegExp>/electrical/inverters/<RegExp>/inverterMode

Description: Mode of inverter

/vessels/<RegExp>/electrical/chargers

Description: Data about AC sourced battery charger

/vessels/<RegExp>/electrical/chargers/<RegExp>

Title: Charger

Description: Battery charger

/vessels/<RegExp>/electrical/chargers/<RegExp>/name

Description: Unique ID of device (houseBattery, alternator, Generator, solar1, inverter, charger, combiner, etc.)

/vessels/<RegExp>/electrical/chargers/<RegExp>/location

Description: Installed location of device on vessel

/vessels/<RegExp>/electrical/chargers/<RegExp>/dateInstalled

Units: RFC 3339 (UTC) (undefined)

Description: Date device was installed

/vessels/<RegExp>/electrical/chargers/<RegExp>/manufacturer

Description: [missing]

/vessels/<RegExp>/electrical/chargers/<RegExp>/manufacturer/name

Description: Manufacturer's name

/vessels/<RegExp>/electrical/chargers/<RegExp>/manufacturer/model

Description: Model or part number

/vessels/<RegExp>/electrical/chargers/<RegExp>/manufacturer/URL

Description: Web referance / URL

/vessels/<RegExp>/electrical/chargers/<RegExp>/associatedBus

Description: Name of BUS device is associated with

/vessels/<RegExp>/electrical/chargers/<RegExp>/voltage

Units: V (Volt)

Description: Voltage measured at or as close as possible to the device

---

/vessels/<RegExp>/electrical/chargers/<RegExp>/voltage/ripple

Units: V (Volt)

Description: DC Ripple voltage

/vessels/<RegExp>/electrical/chargers/<RegExp>/current

Units: A (Ampere)

Description: Current flowing out (+ve) or in (-ve) to the device

---

/vessels/<RegExp>/electrical/chargers/<RegExp>/temperature

Title: temperature

Units: K (Kelvin)

Description: Temperature measured within or on the device

/vessels/<RegExp>/electrical/chargers/<RegExp>/chargingAlgorithm

Description: Algorithm being used by the charger

/vessels/<RegExp>/electrical/chargers/<RegExp>/chargerRole

Description: How is charging source configured? Standalone, or in sync with another charger?

/vessels/<RegExp>/electrical/chargers/<RegExp>/chargingMode

Description: Charging mode i.e. float, overcharge, etc.

/vessels/<RegExp>/electrical/chargers/<RegExp>/setpointVoltage

Units: V (Volt)

Description: Target regulation voltage

/vessels/<RegExp>/electrical/chargers/<RegExp>/setpointCurrent

Units: A (Ampere)

Description: Target current limit

/vessels/<RegExp>/electrical/alternators

Description: Data about an Alternator charging device

/vessels/<RegExp>/electrical/alternators/<RegExp>

Title: Alternator

Description: Mechanically driven alternator, includes dynamos

---

/vessels/<RegExp>/electrical/alternators/<RegExp>/name

Description: Unique ID of device (houseBattery, alternator, Generator, solar1, inverter, charger, combiner, etc.)

/vessels/<RegExp>/electrical/alternators/<RegExp>/location

Description: Installed location of device on vessel

/vessels/<RegExp>/electrical/alternators/<RegExp>/dateInstalled

Units: RFC 3339 (UTC) (undefined)

Description: Date device was installed

/vessels/<RegExp>/electrical/alternators/<RegExp>/manufacturer

Description: [missing]

/vessels/<RegExp>/electrical/alternators/<RegExp>/manufacturer/name

Description: Manufacturer's name

/vessels/<RegExp>/electrical/alternators/<RegExp>/manufacturer/model

Description: Model or part number

/vessels/<RegExp>/electrical/alternators/<RegExp>/manufacturer/URL

Description: Web referance / URL

/vessels/<RegExp>/electrical/alternators/<RegExp>/associatedBus

Description: Name of BUS device is associated with

/vessels/<RegExp>/electrical/alternators/<RegExp>/voltage

Units: V (Volt)

Description: Voltage measured at or as close as possible to the device

/vessels/<RegExp>/electrical/alternators/<RegExp>/voltage/ripple

Units: V (Volt)

Description: DC Ripple voltage

/vessels/<RegExp>/electrical/alternators/<RegExp>/current

Units: A (Ampere)

Description: Current flowing out (+ve) or in (-ve) to the device

/vessels/<RegExp>/electrical/alternators/<RegExp>/temperature

Title: temperature

Units: K (Kelvin)

Description: Temperature measured within or on the device

/vessels/<RegExp>/electrical/alternators/<RegExp>/chargingAlgorithm

Description: Algorithm being used by the charger

/vessels/<RegExp>/electrical/alternators/<RegExp>/chargerRole

Description: How is charging source configured? Standalone, or in sync with another charger?

/vessels/<RegExp>/electrical/alternators/<RegExp>/chargingMode

Description: Charging mode i.e. float, overcharge, etc.

/vessels/<RegExp>/electrical/alternators/<RegExp>/setpointVoltage

Units: V (Volt)

Description: Target regulation voltage

/vessels/<RegExp>/electrical/alternators/<RegExp>/setpointCurrent

Units: A (Ampere)

Description: Target current limit

/vessels/<RegExp>/electrical/alternators/<RegExp>/revolutions

Units: Hz (Hertz)

Description: Alternator revolutions per second (x60 for RPM)

/vessels/<RegExp>/electrical/alternators/<RegExp>/pulleyRatio

Units: ratio (Ratio)

Description: Mechanical pulley ratio of driving source (Used to back calculate engine RPMs)

/vessels/<RegExp>/electrical/alternators/<RegExp>/fieldDrive

Units: % (undefined)

Description: % (0..100) of field voltage applied

/vessels/<RegExp>/electrical/alternators/<RegExp>/regulatorTemperature

Units: K (Kelvin)

Description: Current temperature of critical regulator components

---

/vessels/<RegExp>/electrical/solar

Description: Data about Solar charging device(s)

/vessels/<RegExp>/electrical/solar/<RegExp>

Title: Solar

Description: Photovoltaic charging devices

/vessels/<RegExp>/electrical/solar/<RegExp>/name

Description: Unique ID of device (houseBattery, alternator, Generator, solar1, inverter, charger, combiner, etc.)

/vessels/<RegExp>/electrical/solar/<RegExp>/location

Description: Installed location of device on vessel

/vessels/<RegExp>/electrical/solar/<RegExp>/dateInstalled

Units: RFC 3339 (UTC) (undefined)

Description: Date device was installed

/vessels/<RegExp>/electrical/solar/<RegExp>/manufacturer

Description: [missing]

/vessels/<RegExp>/electrical/solar/<RegExp>/manufacturer/name

Description: Manufacturer's name

/vessels/<RegExp>/electrical/solar/<RegExp>/manufacturer/model

Description: Model or part number

/vessels/<RegExp>/electrical/solar/<RegExp>/manufacturer/URL

Description: Web referance / URL

/vessels/<RegExp>/electrical/solar/<RegExp>/associatedBus

Description: Name of BUS device is associated with

/vessels/<RegExp>/electrical/solar/<RegExp>/voltage

Units: V (Volt)

Description: Voltage measured at or as close as possible to the device

---

/vessels/<RegExp>/electrical/solar/<RegExp>/voltage/ripple

Units: V (Volt)

Description: DC Ripple voltage

/vessels/<RegExp>/electrical/solar/<RegExp>/current

Units: A (Ampere)

Description: Current flowing out (+ve) or in (-ve) to the device

---

/vessels/<RegExp>/electrical/solar/<RegExp>/temperature

Title: temperature

Units: K (Kelvin)

Description: Temperature measured within or on the device

/vessels/<RegExp>/electrical/solar/<RegExp>/chargingAlgorithm

Description: Algorithm being used by the charger

/vessels/<RegExp>/electrical/solar/<RegExp>/chargerRole

Description: How is charging source configured? Standalone, or in sync with another charger?

/vessels/<RegExp>/electrical/solar/<RegExp>/chargingMode

Description: Charging mode i.e. float, overcharge, etc.

/vessels/<RegExp>/electrical/solar/<RegExp>/setpointVoltage

Units: V (Volt)

Description: Target regulation voltage

/vessels/<RegExp>/electrical/solar/<RegExp>/setpointCurrent

Units: A (Ampere)

Description: Target current limit

/vessels/<RegExp>/electrical/solar/<RegExp>/controllerMode

Description: The current state of the engine

/vessels/<RegExp>/electrical/solar/<RegExp>/panelVoltage

Units: V (Volt)

Description: Voltage being supplied from Solar Panels to controller

---

/vessels/<RegExp>/electrical/solar/<RegExp>/panelCurrent

Units: A (Ampere)

Description: Amperage being supplied from Solar Panels to controller

---

/vessels/<RegExp>/electrical/solar/<RegExp>/panelTemperature

Units: K (Kelvin)

Description: Temperature of panels

/vessels/<RegExp>/electrical/solar/<RegExp>/load

Description: State of load port on controller (if applicable)

/vessels/<RegExp>/electrical/solar/<RegExp>/loadCurrent

Units: A (Ampere)

Description: Amperage being supplied to load directly connected to controller

---

/vessels/<RegExp>/electrical/ac

Description: AC buses

/vessels/<RegExp>/electrical/ac/<RegExp>

Title: AC Bus keyed by instance id

Description: AC Bus, one or many, within the vessel

/vessels/<RegExp>/electrical/ac/<RegExp>/name

Description: Unique ID of device (houseBattery, alternator, Generator, solar1, inverter, charger, combiner, etc.)

/vessels/<RegExp>/electrical/ac/<RegExp>/location

Description: Installed location of device on vessel

/vessels/<RegExp>/electrical/ac/<RegExp>/dateInstalled

Units: RFC 3339 (UTC) (undefined)

Description: Date device was installed

/vessels/<RegExp>/electrical/ac/<RegExp>/manufacturer

Description: [missing]

/vessels/<RegExp>/electrical/ac/<RegExp>/manufacturer/name

Description: Manufacturer's name

/vessels/<RegExp>/electrical/ac/<RegExp>/manufacturer/model

Description: Model or part number

/vessels/<RegExp>/electrical/ac/<RegExp>/manufacturer/URL

Description: Web referance / URL

/vessels/<RegExp>/electrical/ac/<RegExp>/phase

Description: Single or A,B or C in 3 Phase systems

/vessels/<RegExp>/electrical/ac/<RegExp>/phase/(single)|([A-C])

Title: AC Qualities

Description: AC equipment common qualities

/vessels/<RegExp>/electrical/ac/<RegExp>/phase/(single)|([A-C])/associatedBus

Description: Name of BUS device is associated with

/vessels/<RegExp>/electrical/ac/<RegExp>/phase/(single)|([A-C])/lineNeutralVoltage

Units: V (Volt)

Description: RMS voltage measured between phase and neutral

/vessels/<RegExp>/electrical/ac/<RegExp>/phase/(single)|([A-C])/lineLineVoltage

Units: V (Volt)

Description: RMS voltage measured between phases

/vessels/<RegExp>/electrical/ac/<RegExp>/phase/(single)|([A-C])/current

Units: A (Ampere)

Description: RMS current

/vessels/<RegExp>/electrical/ac/<RegExp>/phase/(single)|([A-C])/frequency

Units: Hz (Hertz)

Description: AC frequency.

/vessels/<RegExp>/electrical/ac/<RegExp>/phase/(single)|([A-C])/reactivePower

Units: W (Watt)

Description: Reactive power

/vessels/<RegExp>/electrical/ac/<RegExp>/phase/(single)|([A-C])/powerFactor

Description: Power factor

/vessels/<RegExp>/electrical/ac/<RegExp>/phase/(single)|([A-C])/powerFactorLagging

Description: Lead/lag status.

Enum values:

- leading
- lagging
- error
- not available

/vessels/<RegExp>/electrical/ac/<RegExp>/phase/(single)|([A-C])/realPower

Units: W (Watt)

Description: Real power.

/vessels/<RegExp>/electrical/ac/<RegExp>/phase/(single)|([A-C])/apparentPower

Units: W (Watt)

Description: Apparent power.

/vessels/<RegExp>/notifications

Title: notifications

Description: Notifications currently raised. Major categories have well-defined names, but the tree can be extended by any hierarchical structure

---

/vessels/<RegExp>/notifications/mob

Description: Man overboard

Object value with properties

- method
- state
- message

/vessels/<RegExp>/notifications/fire

Description: Fire onboard

Object value with properties

- method
- state
- message

/vessels/<RegExp>/notifications/sinking

Description: Vessel is sinking

Object value with properties

- method
- state
- message

/vessels/<RegExp>/notifications/flooding

Description: Vessel is flooding

Object value with properties

- method
- state
- message

/vessels/<RegExp>/notifications/collision

Description: In collision with another vessel or object

Object value with properties

- method
- state
- message

/vessels/<RegExp>/notifications/grounding

Description: Vessel grounding

Object value with properties

- method
- state
- message

/vessels/<RegExp>/notifications/listing

Description: Vessel is listing

Object value with properties

- method
- state
- message

/vessels/<RegExp>/notifications/adrift

Description: Vessel is adrift

Object value with properties

- method
- state
- message

/vessels/<RegExp>/notifications/piracy

Description: Under attack or danger from pirates

Object value with properties

- method
- state
- message

/vessels/<RegExp>/notifications/abandon

Description: Abandon ship

Object value with properties

- method
- state
- message

/vessels/<RegExp>/notifications/<RegExp>

Description: This regex pattern is used for validation of the path of the alarm

---

/vessels/<RegExp>/steering

Title: steering

Description: Vessel steering data for steering controls (not Autopilot 'Nav Data')

/vessels/<RegExp>/steering/rudderAngle

Units: rad (Radian)

Description: Current rudder angle, +ve is rudder to Starboard

---

/vessels/<RegExp>/steering/rudderAngleTarget

Units: rad (Radian)

Description: The angle the rudder should move to, +ve is rudder to Starboard

---

/vessels/<RegExp>/steering/autopilot

Title: autopilot

Description: Autopilot data

/vessels/<RegExp>/steering/autopilot/state

Description: Autopilot state

/vessels/<RegExp>/steering/autopilot/mode

Description: Operational mode

/vessels/<RegExp>/steering/autopilot/target

Title: target

Description: Autopilot target

/vessels/<RegExp>/steering/autopilot/target/windAngleApparent

Units: rad (Radian)

Description: Target angle to steer, relative to Apparent wind +port -starboard

/vessels/<RegExp>/steering/autopilot/target/headingTrue

Units: rad (Radian)

Description: Target heading for autopilot, relative to North

/vessels/<RegExp>/steering/autopilot/target/headingMagnetic

Units: rad (Radian)

Description: Target heading for autopilot, relative to Magnetic North

/vessels/<RegExp>/steering/autopilot/deadZone

Units: rad (Radian)

Description: Dead zone to ignore for rudder corrections

/vessels/<RegExp>/steering/autopilot/backlash

Units: rad (Radian)

Description: Slack in the rudder drive mechanism

/vessels/<RegExp>/steering/autopilot/gain

Description: Auto-pilot gain, higher number equals more rudder movement for a given turn

---

/vessels/<RegExp>/steering/autopilot/maxDriveCurrent

Units: A (Ampere)

Description: Maximum current to use to drive servo

/vessels/<RegExp>/steering/autopilot/maxDriveRate

Units: rad/s (Radian per second)

Description: Maximum rudder rotation speed

/vessels/<RegExp>/steering/autopilot/portLock

Units: rad (Radian)

Description: Position of servo on port lock

/vessels/<RegExp>/steering/autopilot/starboardLock

Units: rad (Radian)

Description: Position of servo on starboard lock

/vessels/<RegExp>/tanks

Title: tanks

Description: Tank data, each tank indentified by a unique name i.e. FreshWater_2

/vessels/<RegExp>/tanks/freshWater

Description: Fresh water tank (drinking)

/vessels/<RegExp>/tanks/freshWater/<RegExp>

Description: Tank, one or many, within the vessel

/vessels/<RegExp>/tanks/freshWater/<RegExp>/name

Description: The name of the tank. Useful if multiple tanks of a certain type are on board

/vessels/<RegExp>/tanks/freshWater/<RegExp>/type

Description: The type of tank

Enum values:

- petrol
- fresh water
- greywater
- blackwater
- holding
- lpg
- diesel
- liveWell
- baitWell
- ballast
- rum

/vessels/<RegExp>/tanks/freshWater/<RegExp>/capacity

Units: m3 (Cubic meter)

Description: Total capacity

/vessels/<RegExp>/tanks/freshWater/<RegExp>/currentLevel

Units: ratio (Ratio)

Description: Level of fluid in tank 0-100%

/vessels/<RegExp>/tanks/freshWater/<RegExp>/currentVolume

Units: m3 (Cubic meter)

Description: Volume of fluid in tank

/vessels/<RegExp>/tanks/freshWater/<RegExp>/pressure

Units: Pa (Pascal)

Description: Pressure of contents in tank, especially LPG/gas

---

/vessels/<RegExp>/tanks/freshWater/<RegExp>/temperature

Units: K (Kelvin)

Description: Temperature of tank, especially cryogenic or LPG/gas

---

/vessels/<RegExp>/tanks/freshWater/<RegExp>/viscosity

Units: Pa/s (undefined)

Description: Viscosity of the fluid, if applicable

/vessels/<RegExp>/tanks/freshWater/<RegExp>/extinguishant

Description: The preferred extinguishant to douse a fire in this tank

---

/vessels/<RegExp>/tanks/wasteWater

Description: Waste water tank (grey water)

/vessels/<RegExp>/tanks/wasteWater/<RegExp>

Description: Tank, one or many, within the vessel

/vessels/<RegExp>/tanks/wasteWater/<RegExp>/name

Description: The name of the tank. Useful if multiple tanks of a certain type are on board

/vessels/<RegExp>/tanks/wasteWater/<RegExp>/type

Description: The type of tank

Enum values:

- petrol
- fresh water
- greywater
- blackwater
- holding
- lpg
- diesel
- liveWell
- baitWell
- ballast
- rum

/vessels/<RegExp>/tanks/wasteWater/<RegExp>/capacity

Units: m3 (Cubic meter)

Description: Total capacity

/vessels/<RegExp>/tanks/wasteWater/<RegExp>/currentLevel

Units: ratio (Ratio)

Description: Level of fluid in tank 0-100%

/vessels/<RegExp>/tanks/wasteWater/<RegExp>/currentVolume

Units: m3 (Cubic meter)

Description: Volume of fluid in tank

/vessels/<RegExp>/tanks/wasteWater/<RegExp>/pressure

Units: Pa (Pascal)

Description: Pressure of contents in tank, especially LPG/gas

/vessels/<RegExp>/tanks/wasteWater/<RegExp>/temperature

Units: K (Kelvin)

Description: Temperature of tank, especially cryogenic or LPG/gas

---

/vessels/<RegExp>/tanks/wasteWater/<RegExp>/viscosity

Units: Pa/s (undefined)

Description: Viscosity of the fluid, if applicable

/vessels/<RegExp>/tanks/wasteWater/<RegExp>/extinguishant

Description: The preferred extinguishant to douse a fire in this tank

---

/vessels/<RegExp>/tanks/blackWater

Description: Black water tank (sewage)

/vessels/<RegExp>/tanks/blackWater/<RegExp>

Description: Tank, one or many, within the vessel

/vessels/<RegExp>/tanks/blackWater/<RegExp>/name

Description: The name of the tank. Useful if multiple tanks of a certain type are on board

---

/vessels/<RegExp>/tanks/blackWater/<RegExp>/type

Description: The type of tank

Enum values:

- petrol
- fresh water
- greywater
- blackwater
- holding
- lpg

- diesel
- liveWell
- baitWell
- ballast
- rum

/vessels/<RegExp>/tanks/blackWater/<RegExp>/capacity

Units: m3 (Cubic meter)

Description: Total capacity

/vessels/<RegExp>/tanks/blackWater/<RegExp>/currentLevel

Units: ratio (Ratio)

Description: Level of fluid in tank 0-100%

/vessels/<RegExp>/tanks/blackWater/<RegExp>/currentVolume

Units: m3 (Cubic meter)

Description: Volume of fluid in tank

/vessels/<RegExp>/tanks/blackWater/<RegExp>/pressure

Units: Pa (Pascal)

Description: Pressure of contents in tank, especially LPG/gas

---

/vessels/<RegExp>/tanks/blackWater/<RegExp>/temperature

Units: K (Kelvin)

Description: Temperature of tank, especially cryogenic or LPG/gas

---

/vessels/<RegExp>/tanks/blackWater/<RegExp>/viscosity

Units: Pa/s (undefined)

Description: Viscosity of the fluid, if applicable

/vessels/<RegExp>/tanks/blackWater/<RegExp>/extinguishant

Description: The preferred extinguishant to douse a fire in this tank

/vessels/<RegExp>/tanks/fuel

Description: Fuel tank (petrol or diesel)

/vessels/<RegExp>/tanks/fuel/<RegExp>

Description: Tank, one or many, within the vessel

/vessels/<RegExp>/tanks/fuel/<RegExp>/name

Description: The name of the tank. Useful if multiple tanks of a certain type are on board

/vessels/<RegExp>/tanks/fuel/<RegExp>/type

Description: The type of tank

Enum values:

- petrol
- fresh water
- greywater
- blackwater
- holding
- lpg
- diesel
- liveWell
- baitWell
- ballast
- rum

/vessels/<RegExp>/tanks/fuel/<RegExp>/capacity

Units: m3 (Cubic meter)

Description: Total capacity

/vessels/<RegExp>/tanks/fuel/<RegExp>/currentLevel

Units: ratio (Ratio)

Description: Level of fluid in tank 0-100%

/vessels/<RegExp>/tanks/fuel/<RegExp>/currentVolume

Units: m3 (Cubic meter)

Description: Volume of fluid in tank

/vessels/<RegExp>/tanks/fuel/<RegExp>/pressure

Units: Pa (Pascal)

Description: Pressure of contents in tank, especially LPG/gas

/vessels/<RegExp>/tanks/fuel/<RegExp>/temperature

Units: K (Kelvin)

Description: Temperature of tank, especially cryogenic or LPG/gas

/vessels/<RegExp>/tanks/fuel/<RegExp>/viscosity

Units: Pa/s (undefined)

Description: Viscosity of the fluid, if applicable

/vessels/<RegExp>/tanks/fuel/<RegExp>/extinguishant

Description: The preferred extinguishant to douse a fire in this tank

/vessels/<RegExp>/tanks/lubrication

Description: Lubrication tank (oil or grease)

/vessels/<RegExp>/tanks/lubrication/<RegExp>

Description: Tank, one or many, within the vessel

/vessels/<RegExp>/tanks/lubrication/<RegExp>/name

Description: The name of the tank. Useful if multiple tanks of a certain type are on board

---

/vessels/<RegExp>/tanks/lubrication/<RegExp>/type

Description: The type of tank

Enum values:

- petrol
- fresh water
- greywater
- blackwater
- holding
- lpg
- diesel
- liveWell
- baitWell
- ballast
- rum

/vessels/<RegExp>/tanks/lubrication/<RegExp>/capacity

Units: m3 (Cubic meter)

Description: Total capacity

/vessels/<RegExp>/tanks/lubrication/<RegExp>/currentLevel

Units: ratio (Ratio)

Description: Level of fluid in tank 0-100%

/vessels/<RegExp>/tanks/lubrication/<RegExp>/currentVolume

Units: m3 (Cubic meter)

Description: Volume of fluid in tank

/vessels/<RegExp>/tanks/lubrication/<RegExp>/pressure

Units: Pa (Pascal)

Description: Pressure of contents in tank, especially LPG/gas

---

/vessels/<RegExp>/tanks/lubrication/<RegExp>/temperature

Units: K (Kelvin)

Description: Temperature of tank, especially cryogenic or LPG/gas

---

/vessels/<RegExp>/tanks/lubrication/<RegExp>/viscosity

Units: Pa/s (undefined)

Description: Viscosity of the fluid, if applicable

/vessels/<RegExp>/tanks/lubrication/<RegExp>/extinguishant

Description: The preferred extinguishant to douse a fire in this tank

---

/vessels/<RegExp>/tanks/liveWell

Description: Live tank (fish)

/vessels/<RegExp>/tanks/liveWell/<RegExp>

Description: Tank, one or many, within the vessel

/vessels/<RegExp>/tanks/liveWell/<RegExp>/name

Description: The name of the tank. Useful if multiple tanks of a certain type are on board

---

/vessels/<RegExp>/tanks/liveWell/<RegExp>/type

Description: The type of tank

Enum values:

- petrol
- fresh water
- greywater
- blackwater
- holding
- lpg
- diesel
- liveWell
- baitWell
- ballast
- rum

/vessels/<RegExp>/tanks/liveWell/<RegExp>/capacity

Units: m3 (Cubic meter)

Description: Total capacity

/vessels/<RegExp>/tanks/liveWell/<RegExp>/currentLevel

Units: ratio (Ratio)

Description: Level of fluid in tank 0-100%

/vessels/<RegExp>/tanks/liveWell/<RegExp>/currentVolume

Units: m3 (Cubic meter)

Description: Volume of fluid in tank

/vessels/<RegExp>/tanks/liveWell/<RegExp>/pressure

Units: Pa (Pascal)

Description: Pressure of contents in tank, especially LPG/gas

---

/vessels/<RegExp>/tanks/liveWell/<RegExp>/temperature

Units: K (Kelvin)

Description: Temperature of tank, especially cryogenic or LPG/gas

/vessels/<RegExp>/tanks/liveWell/<RegExp>/viscosity

Units: Pa/s (undefined)

Description: Viscosity of the fluid, if applicable

/vessels/<RegExp>/tanks/liveWell/<RegExp>/extinguishant

Description: The preferred extinguishant to douse a fire in this tank

/vessels/<RegExp>/tanks/baitWell

Description: Bait tank

/vessels/<RegExp>/tanks/baitWell/<RegExp>

Description: Tank, one or many, within the vessel

/vessels/<RegExp>/tanks/baitWell/<RegExp>/name

Description: The name of the tank. Useful if multiple tanks of a certain type are on board

/vessels/<RegExp>/tanks/baitWell/<RegExp>/type

Description: The type of tank

Enum values:

- petrol
- fresh water
- greywater
- blackwater
- holding
- lpg
- diesel
- liveWell
- baitWell

- ballast
- rum

/vessels/<RegExp>/tanks/baitWell/<RegExp>/capacity

Units: m3 (Cubic meter)

Description: Total capacity

/vessels/<RegExp>/tanks/baitWell/<RegExp>/currentLevel

Units: ratio (Ratio)

Description: Level of fluid in tank 0-100%

/vessels/<RegExp>/tanks/baitWell/<RegExp>/currentVolume

Units: m3 (Cubic meter)

Description: Volume of fluid in tank

/vessels/<RegExp>/tanks/baitWell/<RegExp>/pressure

Units: Pa (Pascal)

Description: Pressure of contents in tank, especially LPG/gas

---

/vessels/<RegExp>/tanks/baitWell/<RegExp>/temperature

Units: K (Kelvin)

Description: Temperature of tank, especially cryogenic or LPG/gas

---

/vessels/<RegExp>/tanks/baitWell/<RegExp>/viscosity

Units: Pa/s (undefined)

Description: Viscosity of the fluid, if applicable

/vessels/<RegExp>/tanks/baitWell/<RegExp>/extinguishant

Description: The preferred extinguishant to douse a fire in this tank

---

/vessels/<RegExp>/tanks/gas

Description: Lpg/propane and other gases

/vessels/<RegExp>/tanks/gas/<RegExp>

Description: Tank, one or many, within the vessel

/vessels/<RegExp>/tanks/gas/<RegExp>/name

Description: The name of the tank. Useful if multiple tanks of a certain type are on board

---

/vessels/<RegExp>/tanks/gas/<RegExp>/type

Description: The type of tank

Enum values:

- petrol
- fresh water
- greywater
- blackwater
- holding
- lpg
- diesel
- liveWell
- baitWell
- ballast
- rum

/vessels/<RegExp>/tanks/gas/<RegExp>/capacity

Units: m3 (Cubic meter)

Description: Total capacity

/vessels/<RegExp>/tanks/gas/<RegExp>/currentLevel

Units: ratio (Ratio)

Description: Level of fluid in tank 0-100%

/vessels/<RegExp>/tanks/gas/<RegExp>/currentVolume

Units: m3 (Cubic meter)

Description: Volume of fluid in tank

/vessels/<RegExp>/tanks/gas/<RegExp>/pressure

Units: Pa (Pascal)

Description: Pressure of contents in tank, especially LPG/gas

---

/vessels/<RegExp>/tanks/gas/<RegExp>/temperature

Units: K (Kelvin)

Description: Temperature of tank, especially cryogenic or LPG/gas

---

/vessels/<RegExp>/tanks/gas/<RegExp>/viscosity

Units: Pa/s (undefined)

Description: Viscosity of the fluid, if applicable

/vessels/<RegExp>/tanks/gas/<RegExp>/extinguishant

Description: The preferred extinguishant to douse a fire in this tank

---

/vessels/<RegExp>/tanks/ballast

Description: Ballast tanks

/vessels/<RegExp>/tanks/ballast/<RegExp>

Description: Tank, one or many, within the vessel

/vessels/<RegExp>/tanks/ballast/<RegExp>/name

Description: The name of the tank. Useful if multiple tanks of a certain type are on board

---

/vessels/<RegExp>/tanks/ballast/<RegExp>/type

Description: The type of tank

Enum values:

- petrol
- fresh water
- greywater
- blackwater
- holding
- lpg
- diesel
- liveWell
- baitWell
- ballast
- rum

/vessels/<RegExp>/tanks/ballast/<RegExp>/capacity

Units: m3 (Cubic meter)

Description: Total capacity

/vessels/<RegExp>/tanks/ballast/<RegExp>/currentLevel

Units: ratio (Ratio)

Description: Level of fluid in tank 0-100%

/vessels/<RegExp>/tanks/ballast/<RegExp>/currentVolume

Units: m3 (Cubic meter)

Description: Volume of fluid in tank

/vessels/<RegExp>/tanks/ballast/<RegExp>/pressure

Units: Pa (Pascal)

Description: Pressure of contents in tank, especially LPG/gas

/vessels/<RegExp>/tanks/ballast/<RegExp>/temperature

Units: K (Kelvin)

Description: Temperature of tank, especially cryogenic or LPG/gas

/vessels/<RegExp>/tanks/ballast/<RegExp>/viscosity

Units: Pa/s (undefined)

Description: Viscosity of the fluid, if applicable

/vessels/<RegExp>/tanks/ballast/<RegExp>/extinguishant

Description: The preferred extinguishant to douse a fire in this tank

/vessels/<RegExp>/design

Title: design

Description: Design/dimensional data of this vessel

/vessels/<RegExp>/design/displacement

Units: kg (Kilogram)

Description: The displacement of the vessel

/vessels/<RegExp>/design/aisShipType

Description: The ais ship type see http://www.bosunsmate.org/ais/message5.php

Object value with properties

- id
- name

/vessels/<RegExp>/design/draft

Title: draft

Description: The draft of the vessel

Object value with properties

- minimum (m)
- maximum (m)
- current (m)
- canoe (m)

/vessels/<RegExp>/design/length

Title: length

Description: The various lengths of the vessel

Object value with properties

- overall (m)
- hull (m)
- waterline (m)

/vessels/<RegExp>/design/keel

Title: keel

Description: Information about the vessel's keel

/vessels/<RegExp>/design/keel/angle

Units: rad (Radian)

Description: A number indicating at which angle the keel currently is (in case of a canting keel), negative to port.

/vessels/<RegExp>/design/keel/lift

Units: ratio (Ratio)

Description: In the case of a lifting keel, centreboard or daggerboard, the part of the keel which is extended. 0 is 'all the way up' and 1 is 'all the way down'. 0.8 would be 80% down.

/vessels/<RegExp>/design/beam

Units: m (Meter)

Description: Beam length

/vessels/<RegExp>/design/airHeight

Units: m (Meter)

Description: Total height of the vessel

/vessels/<RegExp>/design/rigging

Title: rigging

Description: Information about the vessel's rigging

/vessels/<RegExp>/sails

Title: sails

Description: Sails data

/vessels/<RegExp>/sails/inventory

Description: An object containing a description of each sail available to the vessel crew

---

/vessels/<RegExp>/sails/inventory/<RegExp>

Description: 'sail' data type.

/vessels/<RegExp>/sails/area

Description: An object containing information about the vessels' sails.

/vessels/<RegExp>/sails/area/total

Units: m2 (Square meter)

Description: The total area of all sails on the vessel

/vessels/<RegExp>/sails/area/active

Units: m2 (Square meter)

Description: The total area of the sails currently in use on the vessel

/vessels/<RegExp>/sensors

Title: sensors

Description: Sensors, their state, and data.

/vessels/<RegExp>/sensors/<RegExp>

Title: sensor

Description: This regex pattern is used for validation UUID identifier for the sensor

/vessels/<RegExp>/sensors/<RegExp>/name

Description: The common name of the sensor

/vessels/<RegExp>/sensors/<RegExp>/sensorType

Description: The datamodel definition of the sensor data. FIXME - need to create a definitions lib of sensor datamodel types

/vessels/<RegExp>/sensors/<RegExp>/sensorData

Description: The data of the sensor data. FIXME - need to ref the definitions of sensor types

/vessels/<RegExp>/sensors/<RegExp>/fromBow

Description: The distance from the bow to the sensor location

/vessels/<RegExp>/sensors/<RegExp>/fromCenter

Description: The distance from the centerline to the sensor location, -ve to starboard, +ve to port

/vessels/<RegExp>/sensors/<RegExp>/class

Description: AIS transponder class in sensors.ais.class, A or B

/vessels/<RegExp>/performance

Title: performance

Description: Performance Sailing data including VMG, Polar Speed, tack angle, etc.

/vessels/<RegExp>/performance/polarSpeed

Units: m/s (Meters per second)

Description: The current polar speed based on current polar diagram, WindSpeedTrue and angleTrueWater.

/vessels/<RegExp>/performance/polarSpeedRatio

Units: ratio (Ratio)

Description: The ratio of current speed through water to the polar speed.

/vessels/<RegExp>/performance/velocityMadeGood

Units: m/s (Meters per second)

Description: The current velocity made good derived from the speed through water and appearant wind angle. A positive value is heading to upwind, negative to downwind.

/vessels/<RegExp>/performance/velocityMadeGoodToWaypoint

Units: m/s (Meters per second)

Description: The current velocity made good to the next waypoint derived from the speedOverGround, courseOverGround.

/vessels/<RegExp>/performance/beatAngle

Units: rad (Radian)

Description: The true wind beat angle for the best velocity made good based on current current polar diagram and WindSpeedTrue.

/vessels/<RegExp>/performance/beatAngleVelocityMadeGood

Units: m/s (Meters per second)

Description: The velocity made good for the beat angle.

/vessels/<RegExp>/performance/beatAngleTargetSpeed

Units: m/s (Meters per second)

Description: The target speed for the beat angle.

/vessels/<RegExp>/performance/gybeAngle

Units: rad (Radian)

Description: The true wind gybe angle for the best velocity made good downwind based on current polar diagram and WindSpeedTrue.

/vessels/<RegExp>/performance/gybeAngleVelocityMadeGood

Units: m/s (Meters per second)

Description: The velocity made good for the gybe angle

/vessels/<RegExp>/performance/gybeAngleTargetSpeed

Units: m/s (Meters per second)

Description: The target speed for the gybe angle.

/vessels/<RegExp>/performance/targetAngle

Units: rad (Radian)

Description: The true wind gybe or beat angle for the best velocity made good downwind or upwind based on current polar diagram and WindSpeedTrue.

/vessels/<RegExp>/performance/targetSpeed

Units: m/s (Meters per second)

Description: The target speed for the beat angle or gybe angle, which ever is applicable.

/vessels/<RegExp>/performance/leeway

Units: rad (Radian)

Description: Current leeway

/vessels/<RegExp>/performance/tackMagnetic

Units: rad (Radian)

Description: Magnetic heading on opposite tack.

/vessels/<RegExp>/performance/tackTrue

Units: rad (Radian)

Description: True heading on opposite tack.

/self

Description: This holds the context (prefix + UUID, MMSI or URL in dot notation) of the server's self object.

/aircraft

Description: A wrapper object for aircraft, primarily intended for SAR aircraft in relation to marine search and rescue. For clarity about seaplanes etc, if it CAN fly, its an aircraft.

/aircraft/<RegExp>

Title: aircraft

Description: This regex pattern is used for validation of an MMSI or Signal K UUID identifier for the aircraft. Examples: urn:mrn:imo:mmsi:111099999 urn:mrn:signalk:uuid:c0d79334-4e25-4245-8892-54e8ccc8021d

/aircraft/<RegExp>/url

Description: URL based identity of the aircraft, if available.

/aircraft/<RegExp>/mmsi

Description: MMSI number of the aircraft, if available.

/aircraft/<RegExp>/uuid

Description: A unique Signal K flavoured maritime resource identifier, assigned by the server.

/aircraft/<RegExp>/flag

Description: The country of aircraft registration, or flag state of the aircraft

/aircraft/<RegExp>/base

Description: The home base of the aircraft

/aircraft/<RegExp>/registrations

Description: The various registrations of the aircraft.

/aircraft/<RegExp>/registrations/imo

Description: The IMO number of the aircraft.

/aircraft/<RegExp>/registrations/national

Description: The national registration number of the aircraft.

/aircraft/<RegExp>/registrations/national/<RegExp>

Description: This regex pattern is used for validating the identifier for the registration

---

/aircraft/<RegExp>/registrations/national/<RegExp>/country

Description: The ISO 3166-2 country code.

/aircraft/<RegExp>/registrations/national/<RegExp>/registration

Description: The registration code

/aircraft/<RegExp>/registrations/national/<RegExp>/description

Description: The registration description

/aircraft/<RegExp>/registrations/other

Description: Other registration or permits for the aircraft.

/aircraft/<RegExp>/registrations/other/<RegExp>

Description: This regex pattern is used for validating the identifier for the registration

/aircraft/<RegExp>/registrations/other/<RegExp>/registration

Description: The registration code

/aircraft/<RegExp>/registrations/other/<RegExp>/description

Description: The registration description

/aircraft/<RegExp>/communication

Title: communication

Description: Communication data including Radio, Telephone, E-Mail, etc.

/aircraft/<RegExp>/communication/callsignVhf

Description: Callsign for VHF communication

/aircraft/<RegExp>/communication/callsignHf

Description: Callsign for HF communication

/aircraft/<RegExp>/communication/phoneNumber

Description: Phone number of skipper

/aircraft/<RegExp>/communication/emailHf

Description: Email address to be used for HF email (Winmail, Airmail, Sailmail)

/aircraft/<RegExp>/communication/email

Description: Regular email for the skipper

/aircraft/<RegExp>/communication/satPhoneNumber

Description: Satellite phone number for vessel.

/aircraft/<RegExp>/communication/skipperName

Description: Full name of the skipper of the vessel.

/aircraft/<RegExp>/communication/crewNames

Description: Array with the names of the crew

/aircraft/<RegExp>/environment

Title: environment

Description: Environmental data measured locally including Depth, Wind, Temp, etc.

/aircraft/<RegExp>/environment/outside

Description: Environmental conditions outside of the vessel's hull

/aircraft/<RegExp>/environment/outside/temperature

Units: K (Kelvin)

Description: Current outside air temperature

/aircraft/<RegExp>/environment/outside/dewPointTemperature

Units: K (Kelvin)

Description: Current outside dew point temperature

/aircraft/<RegExp>/environment/outside/apparentWindChillTemperature

Units: K (Kelvin)

Description: Current outside apparent wind chill temperature

/aircraft/<RegExp>/environment/outside/theoreticalWindChillTemperature

Units: K (Kelvin)

Description: Current outside theoretical wind chill temperature

---

/aircraft/<RegExp>/environment/outside/heatIndexTemperature

Units: K (Kelvin)

Description: Current outside heat index temperature

/aircraft/<RegExp>/environment/outside/pressure

Units: Pa (Pascal)

Description: Current outside air ambient pressure

/aircraft/<RegExp>/environment/outside/humidity

Units: ratio (Ratio)

Description: Current outside air relative humidity

/aircraft/<RegExp>/environment/outside/airDensity

Units: kg/m3 (undefined)

Description: Current outside air density

/aircraft/<RegExp>/environment/outside/illuminance

Units: Lux (undefined)

Description: Current outside ambient light flux.

/aircraft/<RegExp>/environment/inside

Description: Environmental conditions inside the vessel's hull

---

/aircraft/<RegExp>/environment/inside/temperature

Units: K (Kelvin)

Description: Temperature

/aircraft/<RegExp>/environment/inside/heatIndexTemperature

Units: K (Kelvin)

Description: Current heat index temperature in zone

/aircraft/<RegExp>/environment/inside/pressure

Units: Pa (Pascal)

Description: Pressure in zone

/aircraft/<RegExp>/environment/inside/relativeHumidity

Units: ratio (Ratio)

Description: Relative humidity in zone

/aircraft/<RegExp>/environment/inside/dewPoint

Units: K (Kelvin)

Description: Dewpoint in zone

/aircraft/<RegExp>/environment/inside/airDensity

Units: kg/m3 (undefined)

Description: Air density in zone

/aircraft/<RegExp>/environment/inside/illuminance

Units: Lux (undefined)

Description: Illuminance in zone

/aircraft/<RegExp>/environment/inside/[A-Za-z0-9]+

Description: This regex pattern is used for validation of the identifier for the environmental zone, eg. engineRoom, mainCabin, refrigerator

/aircraft/<RegExp>/environment/inside/[A-Za-z0-9]+/temperature

Units: K (Kelvin)

Description: Temperature

/aircraft/<RegExp>/environment/inside/[A-Za-z0-9]+/heatIndexTemperature

Units: K (Kelvin)

Description: Current heat index temperature in zone

/aircraft/<RegExp>/environment/inside/[A-Za-z0-9]+/pressure

Units: Pa (Pascal)

Description: Pressure in zone

/aircraft/<RegExp>/environment/inside/[A-Za-z0-9]+/relativeHumidity

Units: ratio (Ratio)

Description: Relative humidity in zone

/aircraft/<RegExp>/environment/inside/[A-Za-z0-9]+/dewPoint

Units: K (Kelvin)

Description: Dewpoint in zone

/aircraft/<RegExp>/environment/inside/[A-Za-z0-9]+/airDensity

Units: kg/m3 (undefined)

Description: Air density in zone

/aircraft/<RegExp>/environment/inside/[A-Za-z0-9]+/illuminance

Units: Lux (undefined)

Description: Illuminance in zone

/aircraft/<RegExp>/environment/water

Description: Environmental conditions of the water that the vessel is sailing in

/aircraft/<RegExp>/environment/water/temperature

Units: K (Kelvin)

Description: Current water temperature

/aircraft/<RegExp>/environment/water/salinity

Units: ratio (Ratio)

Description: Water salinity

/aircraft/<RegExp>/environment/depth

Title: depth

Description: Depth related data

/aircraft/<RegExp>/environment/depth/belowKeel

Units: m (Meter)

Description: Depth below keel

/aircraft/<RegExp>/environment/depth/belowTransducer

Units: m (Meter)

Description: Depth below Transducer

/aircraft/<RegExp>/environment/depth/belowSurface

Units: m (Meter)

Description: Depth from surface

/aircraft/<RegExp>/environment/depth/transducerToKeel

Units: m (Meter)

Description: Depth from the transducer to the bottom of the keel

/aircraft/<RegExp>/environment/depth/surfaceToTransducer

Units: m (Meter)

Description: Depth transducer is below the water surface

/aircraft/<RegExp>/environment/current

Title: current

Description: Direction and strength of current affecting the vessel

Object value with properties

- drift (m/s)
- setTrue (rad)
- setMagnetic (rad)

/aircraft/<RegExp>/environment/tide

Title: tide

Description: Tide data

/aircraft/<RegExp>/environment/tide/heightHigh

Units: m (Meter)

Description: Next high tide height relative to lowest astronomical tide (LAT/Chart Datum)

/aircraft/<RegExp>/environment/tide/heightNow

Units: m (Meter)

Description: The current tide height relative to lowest astronomical tide (LAT/Chart Datum)

/aircraft/<RegExp>/environment/tide/heightLow

Units: m (Meter)

Description: The next low tide height relative to lowest astronomical tide (LAT/Chart Datum)

/aircraft/<RegExp>/environment/tide/timeLow

Units: RFC 3339 (UTC) (undefined)

Description: Time of the next low tide in UTC

/aircraft/<RegExp>/environment/tide/timeHigh

Units: RFC 3339 (UTC) (undefined)

Description: Time of next high tide in UTC

/aircraft/<RegExp>/environment/heave

Units: m (Meter)

Description: Vertical movement of the vessel due to waves

/aircraft/<RegExp>/environment/wind

Title: wind

Description: Wind data.

/aircraft/<RegExp>/environment/wind/angleApparent

Units: rad (Radian)

Description: Apparent wind angle, negative to port

/aircraft/<RegExp>/environment/wind/angleTrueGround

Units: rad (Radian)

Description: True wind angle based on speed over ground, negative to port

---

/aircraft/<RegExp>/environment/wind/angleTrueWater

Units: rad (Radian)

Description: True wind angle based on speed through water, negative to port

---

/aircraft/<RegExp>/environment/wind/directionChangeAlarm

Units: rad (Radian)

Description: The angle the wind needs to shift to raise an alarm

---

/aircraft/<RegExp>/environment/wind/directionTrue

Units: rad (Radian)

Description: The wind direction relative to true north

/aircraft/<RegExp>/environment/wind/directionMagnetic

Units: rad (Radian)

Description: The wind direction relative to magnetic north

/aircraft/<RegExp>/environment/wind/speedTrue

Units: m/s (Meters per second)

Description: Wind speed over water (as calculated from speedApparent and vessel's speed through water)

/aircraft/<RegExp>/environment/wind/speedOverGround

Units: m/s (Meters per second)

Description: Wind speed over ground (as calculated from speedApparent and vessel's speed over ground)

/aircraft/<RegExp>/environment/wind/speedApparent

Units: m/s (Meters per second)

Description: Apparent wind speed

/aircraft/<RegExp>/environment/time

Description: A time reference for the vessel. All clocks on the vessel dispaying local time should use the timezone offset here. If a timezoneRegion is supplied the timezone must also be supplied. If timezoneRegion is supplied that should be displayed by UIs in preference to simply timezone. ie 12:05 (Europe/London) should be displayed in preference to 12:05 (UTC+01:00)

/aircraft/<RegExp>/environment/mode

Description: Mode of the vessel based on the current conditions. Can be combined with navigation.state to control vessel signals eg switch to night mode for instrumentation and lights, or make sound signals for fog.

/aircraft/<RegExp>/navigation

Title: navigation

Description: Navigation data including Position, Course to next WP information, etc.

/aircraft/<RegExp>/navigation/lights

Title: Navigation lights

Description: Current state of the vessels navigation lights

/aircraft/<RegExp>/navigation/courseOverGroundMagnetic

Units: rad (Radian)

Description: Course over ground (magnetic)

/aircraft/<RegExp>/navigation/courseOverGroundTrue

Units: rad (Radian)

Description: Course over ground (true)

/aircraft/<RegExp>/navigation/courseRhumbline

Title: Course

Description: Course information computed with Rhumbline

/aircraft/<RegExp>/navigation/courseRhumbline/crossTrackError

Units: m (Meter)

Description: The distance from the vessel's present position to the closest point on a line (track) between previousPoint and nextPoint. A negative number indicates that the vessel is currently to the left of this line (and thus must steer right to compensate), a positive number means the vessel is to the right of the line (steer left to compensate).

/aircraft/<RegExp>/navigation/courseRhumbline/bearingTrackTrue

Units: rad (Radian)

Description: The bearing of a line between previousPoint and nextPoint, relative to true north.

/aircraft/<RegExp>/navigation/courseRhumbline/bearingTrackMagnetic

Units: rad (Radian)

Description: The bearing of a line between previousPoint and nextPoint, relative to magnetic north.

/aircraft/<RegExp>/navigation/courseRhumbline/activeRoute

Description: Data required if sailing to an active route, defined in resources.

/aircraft/<RegExp>/navigation/courseRhumbline/activeRoute/href

Description: A reference (URL) to the presently active route, in resources.

/aircraft/<RegExp>/navigation/courseRhumbline/activeRoute/estimatedTimeOfArrival

Description: The estimated time of arrival at the end of the current route

---

/aircraft/<RegExp>/navigation/courseRhumbline/activeRoute/startTime

Description: The time this route was activated

/aircraft/<RegExp>/navigation/courseRhumbline/nextPoint

Description: The point on earth the vessel's presently navigating towards

---

/aircraft/<RegExp>/navigation/courseRhumbline/previousPoint

Description: The point on earth the vessel's presently navigating from

Object value with properties

- type
- href

/aircraft/<RegExp>/navigation/courseRhumbline/previousPoint/distance

Units: m (Meter)

Description: The distance in meters between previousPoint and the vessel's present position

---

/aircraft/<RegExp>/navigation/courseRhumbline/previousPoint/position

Title: position

Description: The position of lastPoint in two dimensions

Object value with properties

- longitude (deg)
- latitude (deg)
- altitude (m)

/aircraft/<RegExp>/navigation/courseGreatCircle

Title: Course

Description: Course information computed with Great Circle

/aircraft/<RegExp>/navigation/courseGreatCircle/crossTrackError

Units: m (Meter)

Description: The distance from the vessel's present position to the closest point on a line (track) between previousPoint and nextPoint. A negative number indicates that the vessel is currently to the left of this line (and thus must steer right to compensate), a positive number means the vessel is to the right of the line (steer left to compensate).

/aircraft/<RegExp>/navigation/courseGreatCircle/bearingTrackTrue

Units: rad (Radian)

Description: The bearing of a line between previousPoint and nextPoint, relative to true north.

/aircraft/<RegExp>/navigation/courseGreatCircle/bearingTrackMagnetic

Units: rad (Radian)

Description: The bearing of a line between previousPoint and nextPoint, relative to magnetic north.

/aircraft/<RegExp>/navigation/courseGreatCircle/activeRoute

Description: Data required if sailing to an active route, defined in resources.

/aircraft/<RegExp>/navigation/courseGreatCircle/activeRoute/href

Description: A reference (URL) to the presently active route, in resources.

/aircraft/<RegExp>/navigation/courseGreatCircle/activeRoute/estimatedTimeOfArrival

Description: The estimated time of arrival at the end of the current route

---

/aircraft/<RegExp>/navigation/courseGreatCircle/activeRoute/startTime

Description: The time this route was activated

/aircraft/<RegExp>/navigation/courseGreatCircle/nextPoint

Description: The point on earth the vessel's presently navigating towards

---

/aircraft/<RegExp>/navigation/courseGreatCircle/previousPoint

Description: The point on earth the vessel's presently navigating from

Object value with properties

- type
- href

/aircraft/<RegExp>/navigation/courseGreatCircle/previousPoint/distance

Units: m (Meter)

Description: The distance in meters between previousPoint and the vessel's present position

---

/aircraft/<RegExp>/navigation/courseGreatCircle/previousPoint/position

Title: position

Description: The position of lastPoint in two dimensions

Object value with properties

- longitude (deg)
- latitude (deg)
- altitude (m)

/aircraft/<RegExp>/navigation/closestApproach

Description: Calculated values for other vessels, e.g. from AIS

Object value with properties

- distance (m)
- timeTo (s)

/aircraft/<RegExp>/navigation/racing

Description: Specific navigational data related to yacht racing.

/aircraft/<RegExp>/navigation/racing/startLineStb

Title: position

Description: Position of starboard start mark

Object value with properties

- longitude (deg)
- latitude (deg)
- altitude (m)

/aircraft/<RegExp>/navigation/racing/startLinePort

Title: position

Description: Position of port start mark

Object value with properties

- longitude (deg)
- latitude (deg)
- altitude (m)

/aircraft/<RegExp>/navigation/racing/distanceStartline

Units: m (Meter)

Description: The current distance to the start line

/aircraft/<RegExp>/navigation/racing/timeToStart

Units: s (Second)

Description: Time left before start

/aircraft/<RegExp>/navigation/racing/timePortDown

Units: s (Second)

Description: Time to arrive at the start line on port, turning downwind

---

/aircraft/<RegExp>/navigation/racing/timePortUp

Units: s (Second)

Description: Time to arrive at the start line on port, turning upwind

---

/aircraft/<RegExp>/navigation/racing/timeStbdDown

Units: s (Second)

Description: Time to arrive at the start line on starboard, turning downwind

---

/aircraft/<RegExp>/navigation/racing/timeStbdUp

Units: s (Second)

Description: Time to arrive at the start line on starboard, turning upwind

---

/aircraft/<RegExp>/navigation/racing/layline

Description: The layline crossing the current course

/aircraft/<RegExp>/navigation/racing/layline/distance

Units: m (Meter)

Description: The current distance to the layline

/aircraft/<RegExp>/navigation/racing/layline/time

Units: s (Second)

Description: The time to the layline at current speed and heading

---

/aircraft/<RegExp>/navigation/racing/oppositeLayline

Description: The layline parallell to current course

/aircraft/<RegExp>/navigation/racing/oppositeLayline/distance

Units: m (Meter)

Description: The current distance to the layline

/aircraft/<RegExp>/navigation/racing/oppositeLayline/time

Units: s (Second)

Description: The time to the layline at current speed and heading

---

/aircraft/<RegExp>/navigation/magneticVariation

Units: rad (Radian)

Description: The magnetic variation (declination) at the current position that must be added to the magnetic heading to derive the true heading. Easterly variations are positive and Westerly variations are negative (in Radians).

/aircraft/<RegExp>/navigation/magneticVariationAgeOfService

Units: s (Second)

Description: Seconds since the 1st Jan 1970 that the variation calculation was made

---

/aircraft/<RegExp>/navigation/destination

Title: destination

Description: The intended destination of this trip

/aircraft/<RegExp>/navigation/destination/commonName

Description: Common name of the Destination, eg 'Fiji', also used in ais messages

---

/aircraft/<RegExp>/navigation/destination/eta

Description: Expected time of arrival at destination waypoint

/aircraft/<RegExp>/navigation/destination/waypoint

Description: UUID of destination waypoint

/aircraft/<RegExp>/navigation/gnss

Title: gnss

Description: Global satellite navigation meta information

/aircraft/<RegExp>/navigation/gnss/type

Description: Fix type

/aircraft/<RegExp>/navigation/gnss/methodQuality

Description: Quality of the satellite fix

/aircraft/<RegExp>/navigation/gnss/integrity

Description: Integrity of the satellite fix

/aircraft/<RegExp>/navigation/gnss/satellites

Description: Number of satellites

/aircraft/<RegExp>/navigation/gnss/antennaAltitude

Units: m (Meter)

Description: Altitude of antenna

/aircraft/<RegExp>/navigation/gnss/horizontalDilution

Description: Horizontal Dilution of Precision

/aircraft/<RegExp>/navigation/gnss/positionDilution

Description: Positional Dilution of Precision

/aircraft/<RegExp>/navigation/gnss/geoidalSeparation

Description: Difference between WGS84 earth ellipsoid and mean sea level

---

/aircraft/<RegExp>/navigation/gnss/differentialAge

Units: s (Second)

Description: Age of DGPS data

/aircraft/<RegExp>/navigation/gnss/differentialReference

Description: ID of DGPS base station

/aircraft/<RegExp>/navigation/headingMagnetic

Units: rad (Radian)

Description: Current magnetic heading of the vessel, equals 'headingCompass adjusted for magneticDeviation'

/aircraft/<RegExp>/navigation/magneticDeviation

Units: rad (Radian)

Description: Magnetic deviation of the compass at the current headingCompass

---

/aircraft/<RegExp>/navigation/headingCompass

Units: rad (Radian)

Description: Current magnetic heading received from the compass. This is not adjusted for magneticDeviation of the compass

---

/aircraft/<RegExp>/navigation/headingTrue

Units: rad (Radian)

Description: The current true north heading of the vessel, equals 'headingMagnetic adjusted for magneticVariation'

/aircraft/<RegExp>/navigation/position

Title: position

Description: The position of the vessel in 2 or 3 dimensions (WGS84 datum)

Object value with properties

- longitude (deg)
- latitude (deg)
- altitude (m)

/aircraft/<RegExp>/navigation/attitude

Title: Attitude

Description: Vessel attitude: roll, pitch and yaw

Object value with properties

- roll (rad)
- pitch (rad)
- yaw (rad)

/aircraft/<RegExp>/navigation/maneuver

Description: Special maneuver such as regional passing arrangement. (from ais)

/aircraft/<RegExp>/navigation/rateOfTurn

Units: rad/s (Radian per second)

Description: Rate of turn (+ve is change to starboard). If the value is AIS RIGHT or LEFT, set to +-0.0206 rads and add warning in notifications

---

/aircraft/<RegExp>/navigation/speedOverGround

Units: m/s (Meters per second)

Description: Vessel speed over ground. If converting from AIS 'HIGH' value, set to 102.2 (Ais max value) and add warning in notifications

---

/aircraft/<RegExp>/navigation/speedThroughWater

Units: m/s (Meters per second)

Description: Vessel speed through the water

/aircraft/<RegExp>/navigation/speedThroughWaterTransverse

Units: m/s (Meters per second)

Description: Transverse speed through the water (Leeway)

/aircraft/<RegExp>/navigation/speedThroughWaterLongitudinal

Units: m/s (Meters per second)

Description: Longitudinal speed through the water

/aircraft/<RegExp>/navigation/leewayAngle

Units: rad (Radian)

Description: Leeway Angle derived from the longitudinal and transverse speeds through the water

/aircraft/<RegExp>/navigation/log

Units: m (Meter)

Description: Total distance traveled

/aircraft/<RegExp>/navigation/trip

Description: Trip data

/aircraft/<RegExp>/navigation/trip/log

Units: m (Meter)

Description: Total distance traveled on this trip / since trip reset

/aircraft/<RegExp>/navigation/trip/lastReset

Description: Trip log reset time

/aircraft/<RegExp>/navigation/state

Title: state

Description: Current navigational state of the vessel

/aircraft/<RegExp>/navigation/anchor

Title: anchor

Description: The anchor data, for anchor watch etc

/aircraft/<RegExp>/navigation/anchor/maxRadius

Units: m (Meter)

Description: Radius of anchor alarm boundary. The distance from anchor to the center of the boat

/aircraft/<RegExp>/navigation/anchor/currentRadius

Units: m (Meter)

Description: Current distance to anchor

/aircraft/<RegExp>/navigation/anchor/position

Title: position

Description: The actual anchor position of the vessel in 3 dimensions, probably an estimate at best

Object value with properties

- longitude (deg)
- latitude (deg)
- altitude (m)

/aircraft/<RegExp>/navigation/datetime

Description: Time and Date from the GNSS Positioning System

/aircraft/<RegExp>/notifications

Title: notifications

Description: Notifications currently raised. Major categories have well-defined names, but the tree can be extended by any hierarchical structure

---

/aircraft/<RegExp>/notifications/<RegExp>

Description: This regex pattern is used for validation of the path of the alarm

---

/aircraft/<RegExp>/design

Title: design

Description: Design/dimensional data of this aircraft

/aircraft/<RegExp>/design/displacement

Units: kg (Kilogram)

Description: The displacement of the vessel

/aircraft/<RegExp>/design/aisShipType

Description: The ais ship type see http://www.bosunsmate.org/ais/message5.php

Object value with properties

- id
- name

/aircraft/<RegExp>/design/draft

Title: draft

Description: The draft of the vessel

Object value with properties

- minimum (m)
- maximum (m)
- current (m)
- canoe (m)

/aircraft/<RegExp>/design/length

Title: length

Description: The various lengths of the vessel

Object value with properties

- overall (m)
- hull (m)
- waterline (m)

/aircraft/<RegExp>/design/keel

Title: keel

Description: Information about the vessel's keel

/aircraft/<RegExp>/design/keel/angle

Units: rad (Radian)

Description: A number indicating at which angle the keel currently is (in case of a canting keel), negative to port.

/aircraft/<RegExp>/design/keel/lift

Units: ratio (Ratio)

Description: In the case of a lifting keel, centreboard or daggerboard, the part of the keel which is extended. 0 is 'all the way up' and 1 is 'all the way down'. 0.8 would be 80% down.

/aircraft/<RegExp>/design/beam

Units: m (Meter)

Description: Beam length

/aircraft/<RegExp>/design/airHeight

Units: m (Meter)

Description: Total height of the vessel

/aircraft/<RegExp>/design/rigging

Title: rigging

Description: Information about the vessel's rigging

/aircraft/<RegExp>/sensors

Title: sensors

Description: Sensors, their state, and data.

/aircraft/<RegExp>/sensors/<RegExp>

Title: sensor

Description: This regex pattern is used for validation UUID identifier for the sensor

/aircraft/<RegExp>/sensors/<RegExp>/name

Description: The common name of the sensor

/aircraft/<RegExp>/sensors/<RegExp>/sensorType

Description: The datamodel definition of the sensor data. FIXME - need to create a definitions lib of sensor datamodel types

/aircraft/<RegExp>/sensors/<RegExp>/sensorData

Description: The data of the sensor data. FIXME - need to ref the definitions of sensor types

/aircraft/<RegExp>/sensors/<RegExp>/fromBow

Description: The distance from the bow to the sensor location

/aircraft/<RegExp>/sensors/<RegExp>/fromCenter

Description: The distance from the centerline to the sensor location, -ve to starboard, +ve to port

/aircraft/<RegExp>/sensors/<RegExp>/class

Description: AIS transponder class in sensors.ais.class, A or B

/aton

Description: A wrapper object for Aids to Navigation (aton's)

/aton/<RegExp>

Title: aid to navigation

Description: This regex pattern is used for validation of an MMSI or Signal K UUID identifier for the aid to navigation. Examples: urn:mrn:imo:mmsi:991099999 urn:mrn:signalk:uuid:c0d79334-4e25-4245-8892-54e8ccc8021d

---

/aton/<RegExp>/url

Description: URL based identity of the aid to navigation, if available.

/aton/<RegExp>/mmsi

Description: MMSI number of the aid to navigation, if available.

/aton/<RegExp>/uuid

Description: A unique Signal K flavoured maritime resource identifier, assigned by the server.

/aton/<RegExp>/atonType

Description: The aton type

Object value with properties

- id
- name

/aton/<RegExp>/name

Description: The aton name

/aton/<RegExp>/communication

Title: communication

Description: Communication data including Radio, Telephone, E-Mail, etc.

/aton/<RegExp>/communication/callsignVhf

Description: Callsign for VHF communication

/aton/<RegExp>/communication/callsignHf

Description: Callsign for HF communication

/aton/<RegExp>/communication/phoneNumber

Description: Phone number of skipper

/aton/<RegExp>/communication/emailHf

Description: Email address to be used for HF email (Winmail, Airmail, Sailmail)

/aton/<RegExp>/communication/email

Description: Regular email for the skipper

/aton/<RegExp>/communication/satPhoneNumber

Description: Satellite phone number for vessel.

/aton/<RegExp>/communication/skipperName

Description: Full name of the skipper of the vessel.

/aton/<RegExp>/communication/crewNames

Description: Array with the names of the crew

/aton/<RegExp>/environment

Title: environment

Description: Environmental data measured locally including Depth, Wind, Temp, etc.

/aton/<RegExp>/environment/outside

Description: Environmental conditions outside of the vessel's hull

/aton/<RegExp>/environment/outside/temperature

Units: K (Kelvin)

Description: Current outside air temperature

/aton/<RegExp>/environment/outside/dewPointTemperature

Units: K (Kelvin)

Description: Current outside dew point temperature

/aton/<RegExp>/environment/outside/apparentWindChillTemperature

Units: K (Kelvin)

Description: Current outside apparent wind chill temperature

/aton/<RegExp>/environment/outside/theoreticalWindChillTemperature

Units: K (Kelvin)

Description: Current outside theoretical wind chill temperature

/aton/<RegExp>/environment/outside/heatIndexTemperature

Units: K (Kelvin)

Description: Current outside heat index temperature

/aton/<RegExp>/environment/outside/pressure

Units: Pa (Pascal)

Description: Current outside air ambient pressure

/aton/<RegExp>/environment/outside/humidity

Units: ratio (Ratio)

Description: Current outside air relative humidity

/aton/<RegExp>/environment/outside/airDensity

Units: kg/m3 (undefined)

Description: Current outside air density

/aton/<RegExp>/environment/outside/illuminance

Units: Lux (undefined)

Description: Current outside ambient light flux.

/aton/<RegExp>/environment/inside

Description: Environmental conditions inside the vessel's hull

---

/aton/<RegExp>/environment/inside/temperature

Units: K (Kelvin)

Description: Temperature

/aton/<RegExp>/environment/inside/heatIndexTemperature

Units: K (Kelvin)

Description: Current heat index temperature in zone

/aton/<RegExp>/environment/inside/pressure

Units: Pa (Pascal)

Description: Pressure in zone

/aton/<RegExp>/environment/inside/relativeHumidity

Units: ratio (Ratio)

Description: Relative humidity in zone

/aton/<RegExp>/environment/inside/dewPoint

Units: K (Kelvin)

Description: Dewpoint in zone

/aton/<RegExp>/environment/inside/airDensity

Units: kg/m3 (undefined)

Description: Air density in zone

/aton/<RegExp>/environment/inside/illuminance

Units: Lux (undefined)

Description: Illuminance in zone

/aton/<RegExp>/environment/inside/[A-Za-z0-9]+

Description: This regex pattern is used for validation of the identifier for the environmental zone, eg. engineRoom, mainCabin, refrigerator

---

/aton/<RegExp>/environment/inside/[A-Za-z0-9]+/temperature

Units: K (Kelvin)

Description: Temperature

/aton/<RegExp>/environment/inside/[A-Za-z0-9]+/heatIndexTemperature

Units: K (Kelvin)

Description: Current heat index temperature in zone

/aton/<RegExp>/environment/inside/[A-Za-z0-9]+/pressure

Units: Pa (Pascal)

Description: Pressure in zone

/aton/<RegExp>/environment/inside/[A-Za-z0-9]+/relativeHumidity

Units: ratio (Ratio)

Description: Relative humidity in zone

/aton/<RegExp>/environment/inside/[A-Za-z0-9]+/dewPoint

Units: K (Kelvin)

Description: Dewpoint in zone

/aton/<RegExp>/environment/inside/[A-Za-z0-9]+/airDensity

Units: kg/m3 (undefined)

Description: Air density in zone

/aton/<RegExp>/environment/inside/[A-Za-z0-9]+/illuminance

Units: Lux (undefined)

Description: Illuminance in zone

/aton/<RegExp>/environment/water

Description: Environmental conditions of the water that the vessel is sailing in

/aton/<RegExp>/environment/water/temperature

Units: K (Kelvin)

Description: Current water temperature

/aton/<RegExp>/environment/water/salinity

Units: ratio (Ratio)

Description: Water salinity

/aton/<RegExp>/environment/depth

Title: depth

Description: Depth related data

/aton/<RegExp>/environment/depth/belowKeel

Units: m (Meter)

Description: Depth below keel

/aton/<RegExp>/environment/depth/belowTransducer

Units: m (Meter)

Description: Depth below Transducer

/aton/<RegExp>/environment/depth/belowSurface

Units: m (Meter)

Description: Depth from surface

/aton/<RegExp>/environment/depth/transducerToKeel

Units: m (Meter)

Description: Depth from the transducer to the bottom of the keel

---

/aton/<RegExp>/environment/depth/surfaceToTransducer

Units: m (Meter)

Description: Depth transducer is below the water surface

/aton/<RegExp>/environment/current

Title: current

Description: Direction and strength of current affecting the vessel

Object value with properties

- drift (m/s)
- setTrue (rad)
- setMagnetic (rad)

/aton/<RegExp>/environment/tide

Title: tide

Description: Tide data

/aton/<RegExp>/environment/tide/heightHigh

Units: m (Meter)

Description: Next high tide height relative to lowest astronomical tide (LAT/Chart Datum)

/aton/<RegExp>/environment/tide/heightNow

Units: m (Meter)

Description: The current tide height relative to lowest astronomical tide (LAT/Chart Datum)

/aton/<RegExp>/environment/tide/heightLow

Units: m (Meter)

Description: The next low tide height relative to lowest astronomical tide (LAT/Chart Datum)

/aton/<RegExp>/environment/tide/timeLow

Units: RFC 3339 (UTC) (undefined)

Description: Time of the next low tide in UTC

/aton/<RegExp>/environment/tide/timeHigh

Units: RFC 3339 (UTC) (undefined)

Description: Time of next high tide in UTC

/aton/<RegExp>/environment/heave

Units: m (Meter)

Description: Vertical movement of the vessel due to waves

/aton/<RegExp>/environment/wind

Title: wind

Description: Wind data.

/aton/<RegExp>/environment/wind/angleApparent

Units: rad (Radian)

Description: Apparent wind angle, negative to port

/aton/<RegExp>/environment/wind/angleTrueGround

Units: rad (Radian)

Description: True wind angle based on speed over ground, negative to port

---

/aton/<RegExp>/environment/wind/angleTrueWater

Units: rad (Radian)

Description: True wind angle based on speed through water, negative to port

---

/aton/<RegExp>/environment/wind/directionChangeAlarm

Units: rad (Radian)

Description: The angle the wind needs to shift to raise an alarm

---

/aton/<RegExp>/environment/wind/directionTrue

Units: rad (Radian)

Description: The wind direction relative to true north

/aton/<RegExp>/environment/wind/directionMagnetic

Units: rad (Radian)

Description: The wind direction relative to magnetic north

/aton/<RegExp>/environment/wind/speedTrue

Units: m/s (Meters per second)

Description: Wind speed over water (as calculated from speedApparent and vessel's speed through water)

/aton/<RegExp>/environment/wind/speedOverGround

Units: m/s (Meters per second)

Description: Wind speed over ground (as calculated from speedApparent and vessel's speed over ground)

/aton/<RegExp>/environment/wind/speedApparent

Units: m/s (Meters per second)

Description: Apparent wind speed

/aton/<RegExp>/environment/time

Description: A time reference for the vessel. All clocks on the vessel dispaying local time should use the timezone offset here. If a timezoneRegion is supplied the timezone must also be supplied. If timezoneRegion is supplied that should be displayed by UIs in preference to simply timezone. ie 12:05 (Europe/London) should be displayed in preference to 12:05 (UTC+01:00)

/aton/<RegExp>/environment/mode

Description: Mode of the vessel based on the current conditions. Can be combined with navigation.state to control vessel signals eg switch to night mode for instrumentation and lights, or make sound signals for fog.

/aton/<RegExp>/navigation

Title: navigation

Description: Navigation data including Position, Course to next WP information, etc.

/aton/<RegExp>/navigation/lights

Title: Navigation lights

Description: Current state of the vessels navigation lights

/aton/<RegExp>/navigation/courseOverGroundMagnetic

Units: rad (Radian)

Description: Course over ground (magnetic)

/aton/<RegExp>/navigation/courseOverGroundTrue

Units: rad (Radian)

Description: Course over ground (true)

/aton/<RegExp>/navigation/courseRhumbline

Title: Course

Description: Course information computed with Rhumbline

/aton/<RegExp>/navigation/courseRhumbline/crossTrackError

Units: m (Meter)

Description: The distance from the vessel's present position to the closest point on a line (track) between previousPoint and nextPoint. A negative number indicates that the vessel is currently to the left of this line (and thus must steer right to compensate), a positive number means the vessel is to the right of the line (steer left to compensate).

/aton/<RegExp>/navigation/courseRhumbline/bearingTrackTrue

Units: rad (Radian)

Description: The bearing of a line between previousPoint and nextPoint, relative to true north.

/aton/<RegExp>/navigation/courseRhumbline/bearingTrackMagnetic

Units: rad (Radian)

Description: The bearing of a line between previousPoint and nextPoint, relative to magnetic north.

/aton/<RegExp>/navigation/courseRhumbline/activeRoute

Description: Data required if sailing to an active route, defined in resources.

/aton/<RegExp>/navigation/courseRhumbline/activeRoute/href

Description: A reference (URL) to the presently active route, in resources.

/aton/<RegExp>/navigation/courseRhumbline/activeRoute/estimatedTimeOfArrival

Description: The estimated time of arrival at the end of the current route

---

/aton/<RegExp>/navigation/courseRhumbline/activeRoute/startTime

Description: The time this route was activated

/aton/<RegExp>/navigation/courseRhumbline/nextPoint

Description: The point on earth the vessel's presently navigating towards

---

/aton/<RegExp>/navigation/courseRhumbline/previousPoint

Description: The point on earth the vessel's presently navigating from

Object value with properties

- type
- href

/aton/<RegExp>/navigation/courseRhumbline/previousPoint/distance

Units: m (Meter)

Description: The distance in meters between previousPoint and the vessel's present position

---

/aton/<RegExp>/navigation/courseRhumbline/previousPoint/position

Title: position

Description: The position of lastPoint in two dimensions

Object value with properties

- longitude (deg)
- latitude (deg)
- altitude (m)

/aton/<RegExp>/navigation/courseGreatCircle

Title: Course

Description: Course information computed with Great Circle

/aton/<RegExp>/navigation/courseGreatCircle/crossTrackError

Units: m (Meter)

Description: The distance from the vessel's present position to the closest point on a line (track) between previousPoint and nextPoint. A negative number indicates that the vessel is currently to the left of this line (and thus must steer right to compensate), a positive number means the vessel is to the right of the line (steer left to compensate).

/aton/<RegExp>/navigation/courseGreatCircle/bearingTrackTrue

Units: rad (Radian)

Description: The bearing of a line between previousPoint and nextPoint, relative to true north.

/aton/<RegExp>/navigation/courseGreatCircle/bearingTrackMagnetic

Units: rad (Radian)

Description: The bearing of a line between previousPoint and nextPoint, relative to magnetic north.

/aton/<RegExp>/navigation/courseGreatCircle/activeRoute

Description: Data required if sailing to an active route, defined in resources.

/aton/<RegExp>/navigation/courseGreatCircle/activeRoute/href

Description: A reference (URL) to the presently active route, in resources.

/aton/<RegExp>/navigation/courseGreatCircle/activeRoute/estimatedTimeOfArrival

Description: The estimated time of arrival at the end of the current route

---

/aton/<RegExp>/navigation/courseGreatCircle/activeRoute/startTime

Description: The time this route was activated

/aton/<RegExp>/navigation/courseGreatCircle/nextPoint

Description: The point on earth the vessel's presently navigating towards

---

/aton/<RegExp>/navigation/courseGreatCircle/previousPoint

Description: The point on earth the vessel's presently navigating from

Object value with properties

- type
- href

/aton/<RegExp>/navigation/courseGreatCircle/previousPoint/distance

Units: m (Meter)

Description: The distance in meters between previousPoint and the vessel's present position

---

/aton/<RegExp>/navigation/courseGreatCircle/previousPoint/position

Title: position

Description: The position of lastPoint in two dimensions

Object value with properties

- longitude (deg)
- latitude (deg)
- altitude (m)

/aton/<RegExp>/navigation/closestApproach

Description: Calculated values for other vessels, e.g. from AIS

Object value with properties

- distance (m)
- timeTo (s)

/aton/<RegExp>/navigation/racing

Description: Specific navigational data related to yacht racing.

/aton/<RegExp>/navigation/racing/startLineStb

Title: position

Description: Position of starboard start mark

Object value with properties

- longitude (deg)
- latitude (deg)
- altitude (m)

/aton/<RegExp>/navigation/racing/startLinePort

Title: position

Description: Position of port start mark

Object value with properties

- longitude (deg)
- latitude (deg)
- altitude (m)

/aton/<RegExp>/navigation/racing/distanceStartline

Units: m (Meter)

Description: The current distance to the start line

/aton/<RegExp>/navigation/racing/timeToStart

Units: s (Second)

Description: Time left before start

/aton/<RegExp>/navigation/racing/timePortDown

Units: s (Second)

Description: Time to arrive at the start line on port, turning downwind

---

/aton/<RegExp>/navigation/racing/timePortUp

Units: s (Second)

Description: Time to arrive at the start line on port, turning upwind

---

/aton/<RegExp>/navigation/racing/timeStbdDown

Units: s (Second)

Description: Time to arrive at the start line on starboard, turning downwind

---

/aton/<RegExp>/navigation/racing/timeStbdUp

Units: s (Second)

Description: Time to arrive at the start line on starboard, turning upwind

---

/aton/<RegExp>/navigation/racing/layline

Description: The layline crossing the current course

/aton/<RegExp>/navigation/racing/layline/distance

Units: m (Meter)

Description: The current distance to the layline

/aton/<RegExp>/navigation/racing/layline/time

Units: s (Second)

Description: The time to the layline at current speed and heading

---

/aton/<RegExp>/navigation/racing/oppositeLayline

Description: The layline parallell to current course

/aton/<RegExp>/navigation/racing/oppositeLayline/distance

Units: m (Meter)

Description: The current distance to the layline

/aton/<RegExp>/navigation/racing/oppositeLayline/time

Units: s (Second)

Description: The time to the layline at current speed and heading

---

/aton/<RegExp>/navigation/magneticVariation

Units: rad (Radian)

Description: The magnetic variation (declination) at the current position that must be added to the magnetic heading to derive the true heading. Easterly variations are positive and Westerly variations are negative (in Radians).

/aton/<RegExp>/navigation/magneticVariationAgeOfService

Units: s (Second)

Description: Seconds since the 1st Jan 1970 that the variation calculation was made

---

/aton/<RegExp>/navigation/destination

Title: destination

Description: The intended destination of this trip

/aton/<RegExp>/navigation/destination/commonName

Description: Common name of the Destination, eg 'Fiji', also used in ais messages

---

/aton/<RegExp>/navigation/destination/eta

Description: Expected time of arrival at destination waypoint

---

/aton/<RegExp>/navigation/destination/waypoint

Description: UUID of destination waypoint

/aton/<RegExp>/navigation/gnss

Title: gnss

Description: Global satellite navigation meta information

/aton/<RegExp>/navigation/gnss/type

Description: Fix type

/aton/<RegExp>/navigation/gnss/methodQuality

Description: Quality of the satellite fix

/aton/<RegExp>/navigation/gnss/integrity

Description: Integrity of the satellite fix

/aton/<RegExp>/navigation/gnss/satellites

Description: Number of satellites

/aton/<RegExp>/navigation/gnss/antennaAltitude

Units: m (Meter)

Description: Altitude of antenna

/aton/<RegExp>/navigation/gnss/horizontalDilution

Description: Horizontal Dilution of Precision

/aton/<RegExp>/navigation/gnss/positionDilution

Description: Positional Dilution of Precision

/aton/<RegExp>/navigation/gnss/geoidalSeparation

Description: Difference between WGS84 earth ellipsoid and mean sea level

---

/aton/<RegExp>/navigation/gnss/differentialAge

Units: s (Second)

Description: Age of DGPS data

/aton/<RegExp>/navigation/gnss/differentialReference

Description: ID of DGPS base station

/aton/<RegExp>/navigation/headingMagnetic

Units: rad (Radian)

Description: Current magnetic heading of the vessel, equals 'headingCompass adjusted for magneticDeviation'

/aton/<RegExp>/navigation/magneticDeviation

Units: rad (Radian)

Description: Magnetic deviation of the compass at the current headingCompass

---

/aton/<RegExp>/navigation/headingCompass

Units: rad (Radian)

Description: Current magnetic heading received from the compass. This is not adjusted for magneticDeviation of the compass

---

/aton/<RegExp>/navigation/headingTrue

Units: rad (Radian)

Description: The current true north heading of the vessel, equals 'headingMagnetic adjusted for magneticVariation'

/aton/<RegExp>/navigation/position

Title: position

Description: The position of the vessel in 2 or 3 dimensions (WGS84 datum)

Object value with properties

- longitude (deg)
- latitude (deg)
- altitude (m)

/aton/<RegExp>/navigation/attitude

Title: Attitude

Description: Vessel attitude: roll, pitch and yaw

Object value with properties

- roll (rad)
- pitch (rad)
- yaw (rad)

/aton/<RegExp>/navigation/maneuver

Description: Special maneuver such as regional passing arrangement. (from ais)

/aton/<RegExp>/navigation/rateOfTurn

Units: rad/s (Radian per second)

Description: Rate of turn (+ve is change to starboard). If the value is AIS RIGHT or LEFT, set to +-0.0206 rads and add warning in notifications

---

/aton/<RegExp>/navigation/speedOverGround

Units: m/s (Meters per second)

Description: Vessel speed over ground. If converting from AIS 'HIGH' value, set to 102.2 (Ais max value) and add warning in notifications

---

/aton/<RegExp>/navigation/speedThroughWater

Units: m/s (Meters per second)

Description: Vessel speed through the water

/aton/<RegExp>/navigation/speedThroughWaterTransverse

Units: m/s (Meters per second)

Description: Transverse speed through the water (Leeway)

/aton/<RegExp>/navigation/speedThroughWaterLongitudinal

Units: m/s (Meters per second)

Description: Longitudinal speed through the water

/aton/<RegExp>/navigation/leewayAngle

Units: rad (Radian)

Description: Leeway Angle derived from the longitudinal and transverse speeds through the water

---

/aton/<RegExp>/navigation/log

Units: m (Meter)

Description: Total distance traveled

/aton/<RegExp>/navigation/trip

Description: Trip data

/aton/<RegExp>/navigation/trip/log

Units: m (Meter)

Description: Total distance traveled on this trip / since trip reset

/aton/<RegExp>/navigation/trip/lastReset

Description: Trip log reset time

/aton/<RegExp>/navigation/state

Title: state

Description: Current navigational state of the vessel

/aton/<RegExp>/navigation/anchor

Title: anchor

Description: The anchor data, for anchor watch etc

/aton/<RegExp>/navigation/anchor/maxRadius

Units: m (Meter)

Description: Radius of anchor alarm boundary. The distance from anchor to the center of the boat

/aton/<RegExp>/navigation/anchor/currentRadius

Units: m (Meter)

Description: Current distance to anchor

/aton/<RegExp>/navigation/anchor/position

Title: position

Description: The actual anchor position of the vessel in 3 dimensions, probably an estimate at best

Object value with properties

- longitude (deg)
- latitude (deg)
- altitude (m)

/aton/<RegExp>/navigation/datetime

Description: Time and Date from the GNSS Positioning System

/aton/<RegExp>/notifications

Title: notifications

Description: Notifications currently raised. Major categories have well-defined names, but the tree can be extended by any hierarchical structure

/aton/<RegExp>/notifications/<RegExp>

Description: This regex pattern is used for validation of the path of the alarm

/aton/<RegExp>/design

Title: design

Description: Design/dimensional data of this

/aton/<RegExp>/design/displacement

Units: kg (Kilogram)

Description: The displacement of the vessel

/aton/<RegExp>/design/aisShipType

Description: The ais ship type see http://www.bosunsmate.org/ais/message5.php

Object value with properties

- id
- name

/aton/<RegExp>/design/draft

Title: draft

Description: The draft of the vessel

Object value with properties

- minimum (m)
- maximum (m)
- current (m)
- canoe (m)

/aton/<RegExp>/design/length

Title: length

Description: The various lengths of the vessel

Object value with properties

- overall (m)
- hull (m)
- waterline (m)

/aton/<RegExp>/design/keel

Title: keel

Description: Information about the vessel's keel

/aton/<RegExp>/design/keel/angle

Units: rad (Radian)

Description: A number indicating at which angle the keel currently is (in case of a canting keel), negative to port.

/aton/<RegExp>/design/keel/lift

Units: ratio (Ratio)

Description: In the case of a lifting keel, centreboard or daggerboard, the part of the keel which is extended. 0 is 'all the way up' and 1 is 'all the way down'. 0.8 would be 80% down.

/aton/<RegExp>/design/beam

Units: m (Meter)

Description: Beam length

/aton/<RegExp>/design/airHeight

Units: m (Meter)

Description: Total height of the vessel

/aton/<RegExp>/design/rigging

Title: rigging

Description: Information about the vessel's rigging

/aton/<RegExp>/sensors

Title: sensors

Description: Sensors, their state, and data.

/aton/<RegExp>/sensors/<RegExp>

Title: sensor

Description: This regex pattern is used for validation UUID identifier for the sensor

---

/aton/<RegExp>/sensors/<RegExp>/name

Description: The common name of the sensor

/aton/<RegExp>/sensors/<RegExp>/sensorType

Description: The datamodel definition of the sensor data. FIXME - need to create a definitions lib of sensor datamodel types

/aton/<RegExp>/sensors/<RegExp>/sensorData

Description: The data of the sensor data. FIXME - need to ref the definitions of sensor types

/aton/<RegExp>/sensors/<RegExp>/fromBow

Description: The distance from the bow to the sensor location

/aton/<RegExp>/sensors/<RegExp>/fromCenter

Description: The distance from the centerline to the sensor location, -ve to starboard, +ve to port

/aton/<RegExp>/sensors/<RegExp>/class

Description: AIS transponder class in sensors.ais.class, A or B

/sar

Description: A wrapper object for Search And Rescue (SAR) MMSI's usied in transponders. MOB, EPIRBS etc

/sar/<RegExp>

Title: Search and rescue beacons

Description: This regex pattern is used for validation of an MMSI or Signal K UUID identifier for the aid to navigation. Examples: urn:mrn:imo:mmsi:972099999 urn:mrn:signalk:uuid:c0d79334-4e25-4245-8892-54e8ccc8021d

/sar/<RegExp>/url

Description: URL based identity of the aid to navigation, if available.

/sar/<RegExp>/mmsi

Description: MMSI number of the aid to navigation, if available.

/sar/<RegExp>/uuid

Description: A unique Signal K flavoured maritime resource identifier, assigned by the server.

/sar/<RegExp>/communication

Title: communication

Description: Communication data including Radio, Telephone, E-Mail, etc.

/sar/<RegExp>/communication/callsignVhf

Description: Callsign for VHF communication

/sar/<RegExp>/communication/callsignHf

Description: Callsign for HF communication

/sar/<RegExp>/communication/phoneNumber

Description: Phone number of skipper

/sar/<RegExp>/communication/emailHf

Description: Email address to be used for HF email (Winmail, Airmail, Sailmail)

/sar/<RegExp>/communication/email

Description: Regular email for the skipper

/sar/<RegExp>/communication/satPhoneNumber

Description: Satellite phone number for vessel.

/sar/<RegExp>/communication/skipperName

Description: Full name of the skipper of the vessel.

/sar/<RegExp>/communication/crewNames

Description: Array with the names of the crew

/sar/<RegExp>/navigation

Title: navigation

Description: Navigation data including Position, Course to next WP information, etc.

/sar/<RegExp>/navigation/lights

Title: Navigation lights

Description: Current state of the vessels navigation lights

/sar/<RegExp>/navigation/courseOverGroundMagnetic

Units: rad (Radian)

Description: Course over ground (magnetic)

/sar/<RegExp>/navigation/courseOverGroundTrue

Units: rad (Radian)

Description: Course over ground (true)

/sar/<RegExp>/navigation/courseRhumbline

Title: Course

Description: Course information computed with Rhumbline

/sar/<RegExp>/navigation/courseRhumbline/crossTrackError

Units: m (Meter)

Description: The distance from the vessel's present position to the closest point on a line (track) between previousPoint and nextPoint. A negative number indicates that the vessel is currently to the left of this line (and thus must steer right to compensate), a positive number means the vessel is to the right of the line (steer left to compensate).

/sar/<RegExp>/navigation/courseRhumbline/bearingTrackTrue

Units: rad (Radian)

Description: The bearing of a line between previousPoint and nextPoint, relative to true north.

/sar/<RegExp>/navigation/courseRhumbline/bearingTrackMagnetic

Units: rad (Radian)

Description: The bearing of a line between previousPoint and nextPoint, relative to magnetic north.

/sar/<RegExp>/navigation/courseRhumbline/activeRoute

Description: Data required if sailing to an active route, defined in resources.

/sar/<RegExp>/navigation/courseRhumbline/activeRoute/href

Description: A reference (URL) to the presently active route, in resources.

/sar/<RegExp>/navigation/courseRhumbline/activeRoute/estimatedTimeOfArrival

Description: The estimated time of arrival at the end of the current route

---

/sar/<RegExp>/navigation/courseRhumbline/activeRoute/startTime

Description: The time this route was activated

/sar/<RegExp>/navigation/courseRhumbline/nextPoint

Description: The point on earth the vessel's presently navigating towards

---

/sar/<RegExp>/navigation/courseRhumbline/previousPoint

Description: The point on earth the vessel's presently navigating from

Object value with properties

- type
- href

/sar/<RegExp>/navigation/courseRhumbline/previousPoint/distance

Units: m (Meter)

Description: The distance in meters between previousPoint and the vessel's present position

---

/sar/<RegExp>/navigation/courseRhumbline/previousPoint/position

Title: position

Description: The position of lastPoint in two dimensions

Object value with properties

- longitude (deg)
- latitude (deg)
- altitude (m)

/sar/<RegExp>/navigation/courseGreatCircle

Title: Course

Description: Course information computed with Great Circle

/sar/<RegExp>/navigation/courseGreatCircle/crossTrackError

Units: m (Meter)

Description: The distance from the vessel's present position to the closest point on a line (track) between previousPoint and nextPoint. A negative number indicates that the vessel is currently to the left of this line (and thus must steer right to compensate), a positive number means the vessel is to the right of the line (steer left to compensate).

/sar/<RegExp>/navigation/courseGreatCircle/bearingTrackTrue

Units: rad (Radian)

Description: The bearing of a line between previousPoint and nextPoint, relative to true north.

/sar/<RegExp>/navigation/courseGreatCircle/bearingTrackMagnetic

Units: rad (Radian)

Description: The bearing of a line between previousPoint and nextPoint, relative to magnetic north.

/sar/<RegExp>/navigation/courseGreatCircle/activeRoute

Description: Data required if sailing to an active route, defined in resources.

/sar/<RegExp>/navigation/courseGreatCircle/activeRoute/href

Description: A reference (URL) to the presently active route, in resources.

/sar/<RegExp>/navigation/courseGreatCircle/activeRoute/estimatedTimeOfArrival

Description: The estimated time of arrival at the end of the current route

---

/sar/<RegExp>/navigation/courseGreatCircle/activeRoute/startTime

Description: The time this route was activated

/sar/<RegExp>/navigation/courseGreatCircle/nextPoint

Description: The point on earth the vessel's presently navigating towards

---

/sar/<RegExp>/navigation/courseGreatCircle/previousPoint

Description: The point on earth the vessel's presently navigating from

Object value with properties

- type
- href

/sar/<RegExp>/navigation/courseGreatCircle/previousPoint/distance

Units: m (Meter)

Description: The distance in meters between previousPoint and the vessel's present position

---

/sar/<RegExp>/navigation/courseGreatCircle/previousPoint/position

Title: position

Description: The position of lastPoint in two dimensions

Object value with properties

- longitude (deg)
- latitude (deg)
- altitude (m)

/sar/<RegExp>/navigation/closestApproach

Description: Calculated values for other vessels, e.g. from AIS

Object value with properties

- distance (m)
- timeTo (s)

/sar/<RegExp>/navigation/racing

Description: Specific navigational data related to yacht racing.

/sar/<RegExp>/navigation/racing/startLineStb

Title: position

Description: Position of starboard start mark

Object value with properties

- longitude (deg)
- latitude (deg)
- altitude (m)

/sar/<RegExp>/navigation/racing/startLinePort

Title: position

Description: Position of port start mark

Object value with properties

- longitude (deg)
- latitude (deg)
- altitude (m)

/sar/<RegExp>/navigation/racing/distanceStartline

Units: m (Meter)

Description: The current distance to the start line

/sar/<RegExp>/navigation/racing/timeToStart

Units: s (Second)

Description: Time left before start

/sar/<RegExp>/navigation/racing/timePortDown

Units: s (Second)

Description: Time to arrive at the start line on port, turning downwind

---

/sar/<RegExp>/navigation/racing/timePortUp

Units: s (Second)

Description: Time to arrive at the start line on port, turning upwind

---

/sar/<RegExp>/navigation/racing/timeStbdDown

Units: s (Second)

Description: Time to arrive at the start line on starboard, turning downwind

---

/sar/<RegExp>/navigation/racing/timeStbdUp

Units: s (Second)

Description: Time to arrive at the start line on starboard, turning upwind

---

/sar/<RegExp>/navigation/racing/layline

Description: The layline crossing the current course

/sar/<RegExp>/navigation/racing/layline/distance

Units: m (Meter)

Description: The current distance to the layline

/sar/<RegExp>/navigation/racing/layline/time

Units: s (Second)

Description: The time to the layline at current speed and heading

---

/sar/<RegExp>/navigation/racing/oppositeLayline

Description: The layline parallell to current course

/sar/<RegExp>/navigation/racing/oppositeLayline/distance

Units: m (Meter)

Description: The current distance to the layline

/sar/<RegExp>/navigation/racing/oppositeLayline/time

Units: s (Second)

Description: The time to the layline at current speed and heading

/sar/<RegExp>/navigation/magneticVariation

Units: rad (Radian)

Description: The magnetic variation (declination) at the current position that must be added to the magnetic heading to derive the true heading. Easterly variations are positive and Westerly variations are negative (in Radians).

/sar/<RegExp>/navigation/magneticVariationAgeOfService

Units: s (Second)

Description: Seconds since the 1st Jan 1970 that the variation calculation was made

/sar/<RegExp>/navigation/destination

Title: destination

Description: The intended destination of this trip

/sar/<RegExp>/navigation/destination/commonName

Description: Common name of the Destination, eg 'Fiji', also used in ais messages

/sar/<RegExp>/navigation/destination/eta

Description: Expected time of arrival at destination waypoint

/sar/<RegExp>/navigation/destination/waypoint

Description: UUID of destination waypoint

/sar/<RegExp>/navigation/gnss

Title: gnss

Description: Global satellite navigation meta information

/sar/<RegExp>/navigation/gnss/type

Description: Fix type

/sar/<RegExp>/navigation/gnss/methodQuality

Description: Quality of the satellite fix

/sar/<RegExp>/navigation/gnss/integrity

Description: Integrity of the satellite fix

/sar/<RegExp>/navigation/gnss/satellites

Description: Number of satellites

/sar/<RegExp>/navigation/gnss/antennaAltitude

Units: m (Meter)

Description: Altitude of antenna

/sar/<RegExp>/navigation/gnss/horizontalDilution

Description: Horizontal Dilution of Precision

/sar/<RegExp>/navigation/gnss/positionDilution

Description: Positional Dilution of Precision

/sar/<RegExp>/navigation/gnss/geoidalSeparation

Description: Difference between WGS84 earth ellipsoid and mean sea level

---

/sar/<RegExp>/navigation/gnss/differentialAge

Units: s (Second)

Description: Age of DGPS data

/sar/<RegExp>/navigation/gnss/differentialReference

Description: ID of DGPS base station

/sar/<RegExp>/navigation/headingMagnetic

Units: rad (Radian)

Description: Current magnetic heading of the vessel, equals 'headingCompass adjusted for magneticDeviation'

/sar/<RegExp>/navigation/magneticDeviation

Units: rad (Radian)

Description: Magnetic deviation of the compass at the current headingCompass

---

/sar/<RegExp>/navigation/headingCompass

Units: rad (Radian)

Description: Current magnetic heading received from the compass. This is not adjusted for magneticDeviation of the compass

---

/sar/<RegExp>/navigation/headingTrue

Units: rad (Radian)

Description: The current true north heading of the vessel, equals 'headingMagnetic adjusted for magneticVariation'

/sar/<RegExp>/navigation/position

Title: position

Description: The position of the vessel in 2 or 3 dimensions (WGS84 datum)

Object value with properties

- longitude (deg)
- latitude (deg)
- altitude (m)

/sar/<RegExp>/navigation/attitude

Title: Attitude

Description: Vessel attitude: roll, pitch and yaw

Object value with properties

- roll (rad)
- pitch (rad)
- yaw (rad)

/sar/<RegExp>/navigation/maneuver

Description: Special maneuver such as regional passing arrangement. (from ais)

/sar/<RegExp>/navigation/rateOfTurn

Units: rad/s (Radian per second)

Description: Rate of turn (+ve is change to starboard). If the value is AIS RIGHT or LEFT, set to +-0.0206 rads and add warning in notifications

---

/sar/<RegExp>/navigation/speedOverGround

Units: m/s (Meters per second)

Description: Vessel speed over ground. If converting from AIS 'HIGH' value, set to 102.2 (Ais max value) and add warning in notifications

---

/sar/<RegExp>/navigation/speedThroughWater

Units: m/s (Meters per second)

Description: Vessel speed through the water

/sar/<RegExp>/navigation/speedThroughWaterTransverse

Units: m/s (Meters per second)

Description: Transverse speed through the water (Leeway)

/sar/<RegExp>/navigation/speedThroughWaterLongitudinal

Units: m/s (Meters per second)

Description: Longitudinal speed through the water

/sar/<RegExp>/navigation/leewayAngle

Units: rad (Radian)

Description: Leeway Angle derived from the longitudinal and transverse speeds through the water

/sar/<RegExp>/navigation/log

Units: m (Meter)

Description: Total distance traveled

/sar/<RegExp>/navigation/trip

Description: Trip data

/sar/<RegExp>/navigation/trip/log

Units: m (Meter)

Description: Total distance traveled on this trip / since trip reset

/sar/<RegExp>/navigation/trip/lastReset

Description: Trip log reset time

/sar/<RegExp>/navigation/state

Title: state

Description: Current navigational state of the vessel

/sar/<RegExp>/navigation/anchor

Title: anchor

Description: The anchor data, for anchor watch etc

/sar/<RegExp>/navigation/anchor/maxRadius

Units: m (Meter)

Description: Radius of anchor alarm boundary. The distance from anchor to the center of the boat

---

/sar/<RegExp>/navigation/anchor/currentRadius

Units: m (Meter)

Description: Current distance to anchor

/sar/<RegExp>/navigation/anchor/position

Title: position

Description: The actual anchor position of the vessel in 3 dimensions, probably an estimate at best

Object value with properties

- longitude (deg)
- latitude (deg)
- altitude (m)

/sar/<RegExp>/navigation/datetime

Description: Time and Date from the GNSS Positioning System

/sar/<RegExp>/notifications

Title: notifications

Description: Notifications currently raised. Major categories have well-defined names, but the tree can be extended by any hierarchical structure

---

/sar/<RegExp>/notifications/mob

Description: Man overboard

Object value with properties

- method
- state
- message

/sar/<RegExp>/notifications/fire

Description: Fire onboard

Object value with properties

- method
- state
- message

/sar/<RegExp>/notifications/sinking

Description: Vessel is sinking

Object value with properties

- method
- state
- message

/sar/<RegExp>/notifications/flooding

Description: Vessel is flooding

Object value with properties

- method
- state
- message

/sar/<RegExp>/notifications/collision

Description: In collision with another vessel or object

Object value with properties

- method
- state
- message

/sar/<RegExp>/notifications/grounding

Description: Vessel grounding

Object value with properties

- method
- state
- message

/sar/<RegExp>/notifications/listing

Description: Vessel is listing

Object value with properties

- method
- state
- message

/sar/<RegExp>/notifications/adrift

Description: Vessel is adrift

Object value with properties

- method
- state
- message

/sar/<RegExp>/notifications/piracy

Description: Under attack or danger from pirates

Object value with properties

- method
- state
- message

/sar/<RegExp>/notifications/abandon

Description: Abandon ship

Object value with properties

- method
- state
- message

/sar/<RegExp>/notifications/<RegExp>

Description: This regex pattern is used for validation of the path of the alarm

---

/resources

Title: resources

Description: Resources to aid in navigation and operation of the vessel including waypoints, routes, notes, etc.

/resources/charts

Title: chart

Description: A holder for charts, each named with their chart code

---

/resources/charts/<RegExp>

Description: A chart

/resources/routes

Title: route

Description: A holder for routes, each named with a UUID

/resources/routes/<RegExp>

Description: A route, named with a UUID

/resources/notes

Title: notes

Description: A holder for notes about regions, each named with a UUID. Notes might include navigation or cruising info, images, or anything

---

/resources/notes/<RegExp>

Description: A note about a region, named with a UUID. Notes might include navigation or cruising info, images, or anything

---

/resources/notes/<RegExp>/position

Title: position

Description: Position related to note. Alternative to region or geohash

Object value with properties

- longitude (deg)

- latitude (deg)
- altitude (m)

/resources/regions

Title: region

Description: A holder for regions, each named with UUID

/resources/regions/<RegExp>

Description: A region of interest, each named with a UUID

/resources/waypoints

Title: waypoints

Description: A holder for waypoints, each named with a UUID

/resources/waypoints/<RegExp>

Description: A waypoint, named with a UUID

/resources/waypoints/<RegExp>/position

Title: position

Description: The position in 3 dimensions

Object value with properties

- longitude (deg)
- latitude (deg)
- altitude (m)

/resources/waypoints/<RegExp>/feature

Title: Feature

Description: A Geo JSON feature object

/resources/waypoints/<RegExp>/feature/type

Description: [missing]

Enum values:

- Feature

/resources/waypoints/<RegExp>/feature/geometry

Title: Point

Description: [missing]

/resources/waypoints/<RegExp>/feature/geometry/type

Description: [missing]

Enum values:

- Point

/resources/waypoints/<RegExp>/feature/geometry/coordinates

Description: A single position, in x,y order (Lon, Lat)

/resources/waypoints/<RegExp>/feature/properties

Description: Additional data of any type

/resources/waypoints/<RegExp>/feature/id

Description: [missing]

/version

Description: Version of the schema and APIs that this data is using in Canonical format i.e. V1.0.0.

Change Log

v1.5.0 (2020/11/01 16:29 +00:00)

- #579 build(deps): bump lodash from 4.17.13 to 4.17.19 (@dependabot[bot])
- #568 feature: add sendCachedValues ws query parameter (@tkurki)
- #558 build(deps): bump fstream from 1.0.11 to 1.0.12 (@dependabot[bot])
- #556 build(deps): bump js-yaml from 3.10.0 to 3.13.1 (@dependabot[bot])
- #557 build(deps): bump tar from 2.2.1 to 2.2.2 (@dependabot[bot])
- #576 feature: add NMEA can name to `source` (@sbender9)
- #562 feature: add sensors.ais.class (@tkurki)
- #565 fix: merge meta from spec with meta from defaults (@sbender9)
- #567 fix: add commonValueFields to propulsion.*.transmission.gear (@sbender9)

v1.4.0 (2020/02/05 19:18 +00:00)

- #561 Fix typo (@nkarstens)
- #559 feature: specify streaming over TCP in more detail (@tkurki)
- #551 build(deps): bump lodash from 4.17.11 to 4.17.13 (@dependabot[bot])
- #535 Update security doc with full functionality and support for other transports (@rob42)
- #542 Add Apps API page (@rob42)

v1.3.1 (2019/06/06 11:12 +00:00)

- #490 corrected error in description of anchor alarm notification object de… (@RBerliner)
- #548 feature: include all source information under `sources` (@sbender9)
- #546 fix: get wording (@sbender9)
- #537 Revert "Fix PUT default context and warn about paths" (@rob42)
- #514 [WIP] Create openapi definitions for REST API's (@rob42)
- #536 Fix PUT default context and warn about paths (@rob42)
- #523 Fix delta PUT message format (@rob42)
- #528 Fix message schemas (@tkurki)
- #531 Added headingRaw, and magneticDeviation, adjust descriptions to be clearer (@rob42)
- #471 docs: improve documentation for displayName, longName and shortName (@bkp7)
- #529 chore: update to lodash 4 (@sarfata)
- #526 Fix: reducedState as object (@joabakk)
- #527 Corrected typos (@Krillle)

v1.3.0 (2018/11/11 16:03 +00:00)

- #506 feature: define Signal K security protocols (@sbender9)

- #505 feature: define process for devices to request access to a server (@sbender9)

v1.2.0 (2018/10/28 07:48 +00:00)

- #508 feature: define request reponse semantics (@sbender9)
- #507 feature: PUT Requests (@sbender9)

v1.1.1 (2018/10/20 17:06 +00:00)

- #511 feature: add baitWell as a type of tank (@sarfata)

v1.1.0 (2018/10/18 19:19 +00:00)

- #512 fix: units should not be a required field on metadata (@sarfata)
- #510 History playback and snapshot retrieval APIs (@tkurki)
- #495 docs: fix readme validation error (@bkp7)
- #503 chore: cleanup docson related files (@tkurki)
- #501 feature: update tv4-formats to latest version (@sarfata)
- #497 doc: add MAY re: http/2 (@tkurki)
- #500 fix: Remove unintentional space in propeller.pitch path (@joabakk)
- #475 docs: clarify websockets hello message (@bkp7)
- #476 fix: discrepancy between docs and sample json (@bkp7)
- #486 fix: accommodate any server software versioning scheme (@bkp7)
- #484 chore: fix broken link in documentation (@bkp7)
- #483 chore: remove orphaned service_discovery.md document (@bkp7)
- #469 chore: mark gaugeType as deprecated, update documentation and samples (@bkp7)
- #367 Add reef and furl state to sails (@joabakk)
- #463 feature: add 'nominal' state to meta zones (@bkp7)
- #376 Add opposite layline, reorganize layline, typo (@joabakk)
- #466 fix: change file names to remove spaces (@bkp7)
- #461 feature: add type to meta/displayScale (@bkp7)
- #404 feature: remove unsupported i18n (@tkurki)
- #445 Fix Environment Schema and add tests (@bkp7)
- #447 feature: add heatIndexTemperature to environment inside zones (@bkp7)
- #462 fix: modify schema to enforce `description` and `units` (@bkp7)
- #436 Add displayScale lower and upper limits to metadata (@bkp7)
- #460 docs: clarify timeout definition/usage (@bkp7)

v1.0.4 (2018/04/22 12:55 +00:00)

- #432 Add Documentation for Sources (@timmathews)
- #458 Put descriptive source properties where they belong (@timmathews)
- #451 WIP: docs: add notes covering naming conventions for pull requests (@bkp7)
- #449 Removed spaces from filenames. (@bkp7)
- #433 Convert docs to use verified json samples (@bkp7)

- #448 Modify description of Magnetic Variation (@bkp7)
- #438 fix: invalid electrical schema (@bkp7)
- #419 fix: modify package.json to make it cross platform compatible. (@bkp7)
- #422 refactor: improve usability of json validation tests (@bkp7)
- #429 chore: use node lts version (@tkurki)
- #424 polish: improvements to documentation: versioning.md (@bkp7)
- #421 Update Travis tests to use latest node-js v8.x.x release (@bkp7)
- #415 Additional units with corrections (@bkp7)
- #414 Add the samples to test scripts (@bkp7)

v1.0.3 (2018/01/08 19:40 +00:00)

- #416 fix: make self relative to root in full (@tkurki)
- #413 Fixed: bug in messages tests (@bkp7)
- #412 fix npm install failing on windows platforms (@bkp7)

v1.0.0 (2018/01/01 08:58 +00:00)

- #396 Root relative self (@tkurki)
- #358 Documentation and schema fixes (@rob42)
- #403 Fix typo in connection.md (@gilesvangruisen)

v0.0.1-12 (2017/11/04 19:48 +00:00)

- #315 Object valued properties under value in full model (@tkurki)
- #395 feature: specify WGS84 for coordinates (@tkurki)
- #392 spec: remove sparse format (@tkurki)
- #393 feature: specify WGS84 for coordinates (@tkurki)
- #372 Improve, remove logic error in prop slip description (@joabakk)
- #366 Add CPA and TCPA (@joabakk)
- #390 feature: Add server information to discovery (@tkurki)
- #385 Feature: make /self required (@tkurki)

v0.0.1-11 (2017/10/17 18:46 +00:00)

- #388 feature: add schema api to get ship and aton type names from the id (@sbender9)
- #386 fix: better aisShipType and atonType (@sbender9)

v0.0.1-9 (2017/10/07 14:29 +00:00)

- #383 fix: don't create undefined meta properties (@tkurki)

v0.0.1-7 (2017/09/27 17:42 +00:00)

- #332 feature: add metadata handling to js lib (@tkurki)
- #382 feature: include enums in keyswithmetadata (@tkurki)
- #381 Add outside air density (@joabakk)

v0.0.1-6 (2017/08/09 20:02 +00:00)

- #368 Transpile distributed npm module to ES2015 (@tkurki)

v0.0.1-5 (2017/08/09 19:24 +00:00)

- #375 Add Alternator and Solar devices, create chargerQualitiies + cleanups (@thomasonw)

v0.0.1-4 (2017/08/09 17:16 +00:00)

- #374 Add commonValueFields to notifications (@sbender9)
- #369 Minor fixes (@parsley72)
- #365 Add intakeManifoldTemperature to propulsion schema (@anajavi)
- #364 Change from schemas/version to version/schemas (@rob42)
- #361 Fix description typos in performance (@joabakk)
- #355 [WIP] Convert json id from https://signalk.github.io/specification/schemas… (@rob42)
- #357 Split gitbook doc keys list (@tkurki)
- #353 Fixed typos and text formatting (@aplathan)
- #342 Rewrite multiple sources/values documentation (@tkurki)

v0.0.1-3 (2017/04/19 04:41 +00:00)

- #350 Add page specifying versioning in Signal K (@rob42)
- #337 Ais additions (@rob42)
- #351 Add note on data accuracy (@rob42)
- #349 fix: fix dollarsource references in full tree (@tkurki)
- #347 Multiple value subscriptions documentation (@tkurki)
- #344 Added SAR aircraft, updates tests to handle aircraft, add vessel.mmsi… (@rob42)
- #346 Tanks (@rob42)

v0.0.1-1 (2017/03/19 16:07 +00:00)

- #348 Add note that meta.units MUST be returned for valid keys (@rob42)
- #345 Added trip.log, and trip.lastReset (@rob42)
- #340 Add slack badge (@bkp7)
- #336 Pattern for version (@bkp7)
- #328 Added timezoneRegion to environment.time (@bkp7)
- #321 Fix notifications schema (@bkp7)
- #335 Added illuminance (@rob42)
- #325 Improvements to the chart model (@emilecantin)
- #319 Add changelog generation (@tkurki)
- #327 Updated Schema so that date-time must be in UTC (@bkp7)
- #324 Define timestamp as JSON Schema date-time (@bkp7)
- #320 Fix schema files to be valid against http://json-schema.org/draft-04/schema# (@bkp7)
- #299 Generate documentation for object types from local files (@joux3)

- #312 Specify vessel context and leaf path for delta (@tkurki)
- #313 Itemize course properties (@tkurki)

v0.0.1-0 (2016/12/27 19:54 +00:00)

- #309 Improve Top level signalk overview (@sumps)
- #272 Update multiple values documentation (@tkurki)
- #301 Add support for dollarpath in FullSignalK (@tkurki)
- #306 Removed refererence to Boundary Layer (@sumps)
- #283 Add Transverse Water Speed and Leeway Angle (@sumps)
- #300 Expand on the Discovery Section (@timmathews)
- #290 Make processSchemaFiles produce keyswithmetadata.json (@tkurki)
- #284 Steering group in line with autopilot communication (@joabakk)
- #278 Validate missing schema references & fix missing references (@joux3)
- #270 Use wildcard context in example, mention wildcard (@tkurki)
- #265 Add Changelog (@tkurki)
- #274 Resolve relative references in the same file (@joux3)
- #269 RFC0004 :Replace JsonPath with wildcard in subscription paths (@tkurki)
- #225 Add defaults overlay (@timmathews)
- #257 Reorganized steering group, added test (@joabakk)
- #266 Cleanup of sources schema (@thomasonw)
- #260 Clarify gitbook-docs/README.md (@timmathews)
- #261 Clean ups (@thomasonw)
- #255 Gitbook Documentation mvp (@tkurki)
- #245 Make tank capacity numberValue (@tkurki)
- #238 Source handling for NMEA and non-NMEA sources (@tkurki)
- #232 Added missing message types (@rob42)
- #233 Added chart scale, and some new charts types (@rob42)
- #230 Added examples, and tidied descriptions (@rob42)
- #231 add description to enum values (@sailoog)
- #228 Include enums in keyswithmetadata.json (@tkurki)
- #221 Add GC/RL distinction (@tkurki)
- #223 Consistent Use of JSON-schema draft 04 Format (@timmathews)
- #222 Tank Senders only provide tank level value (@sumps)
- #211 Alternative proposal for "course" object in navigation (@fabdrol)
- #217 Reorganise temperatures a bit (@tkurki)
- #202 Add general & n2k specific info to sources (@tkurki)
- #193 Added racing parameters (@joabakk)
- #204 Renamed refridgeration to refrigerator (@joabakk)
- #201 Fix keys with metadata (@tkurki)
- #200 Added State of Health to Batteries (@sumps)
- #185 Reorganise electrical: remove ac/dc distinction, branches for equipment types (@tkurki)
- #186 Temp, humidity, pressure reorganisation (@tkurki)
- #195 Refactor tanks (@tkurki)
- #194 Unit cleanup and added prop slip dependancies (@tkurki)
- #190 Improve descriptions (@joabakk)
- #182 Replace maxRevolutions with zones usage (@tkurki)

- #175 Added agnostic target speed and target angle (@joabakk)
- #176 Remove legacy Alarm objects from Environment Tree (@sumps)
- #165 Add engine load & torque (@tkurki)
- #118 Multiple values (take 2) (@tkurki)
- #161 Convert zone in meta to an object (@rob42)
- #162 Rename alarms branch to notifications (@rob42)
- #142 Master waypoints ref (@rob42)
- #160 Remove alarm uuid from path as its not needed. (@rob42)
- #159 Update vessel.json (@joabakk)
- #152 Improve alarms - add alert and emergency (@rob42)
- #120 Improve propulsion (@tkurki)
- #119 Rework electrical dc: batteries (@tkurki)
- #151 Use SPDX abbreviation of the license (@tkurki)
- #148 Add schema and tests for endpoint discovery (@jboynes)
- #149 Remove unneeded references to lodash (@jboynes)
- #145 Added registration structure (@rob42)
- #139 Added lights to navigation (@rob42)
- #143 Add env.mode - take2 - Add source/timestamp and test (@rob42)
- #141 Added environment.mode (@rob42)
- #140 Added IMO, flag, port, and reg number (@rob42)
- #136 Add datetime with source information to the Signal K model (@tkurki)
- #137 Master add systime (@rob42)
- #132 Express air pressure change rate as a rate (@jboynes)
- #129 Replace last use of floatValue with numberValue (@jboynes)
- #128 Removed CallsignDsc (@sumps)
- #121 Add Heave to environment.json (@sumps)
- #117 Additional GNSS field added (@sumps)
- #114 Change all units to (derived) SI units (fix issue #30) (@keesverruijt)
- #113 Implement my own suggestion in issue #112 (@keesverruijt)
- #110 Identities, take two (@fabdrol)
- #109 Add NMEA0183 sentence and talker to source (@tkurki)
- #107 Fix schema references and add all subschemas for validation (@tkurki)
- #105 Housekeeping: cleanup of all schemas (@fabdrol)
- #103 Relocate current from navigation to environment (@timmathews)
- #101 Split angleTrue (@tkurki)
- #87 Improve validation (@tkurki)
- #88 Change pgn type to number (@tkurki)
- #81 Format the design group (@MariusVolkhart)
- #78 Format the propulsion group (@MariusVolkhart)
- #79 Format the electrical_dc group (@MariusVolkhart)
- #77 Format the resources group (@MariusVolkhart)
- #80 Format the navigation group (@MariusVolkhart)
- #82 Format the alarms group (@MariusVolkhart)
- #83 Added an UUID property to vessel.json (@fabdrol)
- #75 Format all enums to the same style (@MariusVolkhart)
- #70 Format vessel.json (@MariusVolkhart)
- #54 Pull of restructured ELECTRICAL JSON schema (@thomasonw)
- #52 Electrical, AC (@timmathews)

- #64 Message schemas (@rob42)
- #51 Group roll pitch and yaw (@timmathews)
- #65 Updates to the "design" group, in order to accommodate vessels with a variable keel or centerboard and some other changes (@fabdrol)
- #67 Separate group for sails (@fabdrol)
- #55 Labels for SignalK data items (@tkurki)
- #63 More lenient delta schema (@tkurki)
- #62 Change subschema loading (@tkurki)
- #61 Added n2kMappings (@tkurki)
- #59 Added performance group to vessel.json (@zapfware)
- #57 Define specification for Performance data (@zapfware)
- #58 Add waypoint.distanceActual, correct units for log & logTrip (@zapfware)
- #50 Fix propulsion and sensors and some cleanup (@rob42)
- #44 Reorganize current group (@timmathews)
- #43 Adds source and timestamp to activeRoute (@timmathews)
- #42 Rename navigation.currentRoute (@timmathews)
- #34 Update layout (@timmathews)
- #29 Delta schema & validation (@tkurki)
- #31 Standardize formatting in schema JSON files (@timmathews)
- #27 Changes related to n2k integration (@tkurki)
- #26 Move source/timestamp so it only occurs on leaf nodes (@rob42)
- #24 Indent with 4 spaces (@tkurki)
- #23 Wind angle & direction (@tkurki)
- #21 Added sensors, design data, and anchor data (@rob42)
- #17 Added definition for GNSS object (@fabdrol)
- #15 Resources - add headers etc (@rob42)
- #14 Try to add headers (@rob42)
- #12 Add signalk header to docsun page (@rob42)
- #11 Replace experimental JavaScript calls (@timmathews)
- #9 Resources (@rob42)
- #5 Added alarms group (@rob42)
- #6 Added basic principles (@rob42)
- #4 Fix to docson.js (@timmathews)
- #3 Separated schemas (@fabdrol)
- #1 Added instructions for publishing spec to GH Pages (@timmathews)

Versioning in Signal K specification

Our versioning is based on http://snowplowanalytics.com/blog/2014/05/13/introducing-schemaver-for-semantic-versioning-of-schemas/

Most of this document is reproduced from there.

When versioning a data schema, we are concerned with the backwards-compatibility between the new schema and existing data represented in earlier versions of the schema. This is the fundamental building block of SchemaVer, and explains the divergence from Semantic Versioning.

Heres a simple formula for our SchemaVer:

Given a version number MODEL.REVISION.ADDITION-SUFFIX, increment the:

- MODEL - when you make a breaking schema change which will prevent interaction with any historical data
- REVISION - when you make a schema change which may prevent interaction with some historical data
- ADDITION - when you make a schema change that is compatible with all historical data
- SUFFIX - optional - denotes special versions or active development eg `alpha-1` , `SNAPSHOT`

- The first released version of Signal K will be `1.0.0` .

- The current development version will then move to be `1.0.1-SNAPSHOT`
- The next release candidate might then be `1.0.1-alpha-1`
- The current development version will then move to be `1.0.2-SNAPSHOT`

`SNAPSHOT` denotes a version under active change. If you depend on the `SNAPSHOT` version then every time you build your project it will have changed with what-ever was committed since last time you checked.

Let's make SchemaVer more concrete with some examples using some (truncated and contrived) Signal K Schemas, in reverse order:

Addition

We have an existing JSON Schema, let's call this `1.0.0` :

```
{
  "type": "object",
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "https://signalk.org/demospecification/1.0.0/schemas/groups/communicat
  "description": "Schema describing the communication child-object of a Vessel
  "title": "communication",
  "properties": {
      "dscAddress": {
      "type": "string",
      "description": "MMSI Callsign for VHF communication"
    }
  },
  "required": ["dscAddress"],
  "additionalProperties": false
}
```

Now we want to add an additional field to our schema:

```
{
  "type": "object",
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "https://signalk.org/demospecification/1.0.1/schemas/groups/communicat
  "description": "Schema describing the communication child-object of a Vessel
  "title": "communication",
  "properties": {
    "dscAddress": {
      "type": "string",
      "description": "MMSI Callsign for VHF communication"
    },
    "phoneNumber": {
      "type": "string",
      "description": "Phone number of skipper",
      "example": "+64xxxxxx"
    }
  },
  "required": ["dscAddress"],
  "additionalProperties": false
}
```

Because our new `phoneNumber` field is not a required field, and because version `1.0.0` had `additionalProperties` set to `false`, we know that all historical data will work with this new schema, ie. any json which validates against 1.0.0 will also be valid against 1.0.1.

Therefore we are looking at an `ADDITION`, and so we bump the schema version to `1.0.1`.

Revision

Let’s now make our JSON Schema support additionalProperties - this constitutes another `ADDITION`, so we are now on `1.0.2`:

```
{
  "type": "object",
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "https://signalk.org/demospecification/1.0.2/schemas/groups/communicat
  "description": "Schema describing the communication child-object of a Vessel
  "title": "communication",
  "properties": {
    "dscAddress": {
      "type": "string",
      "description": "MMSI Callsign for VHF communication"
    },
    "phoneNumber": {
      "type": "string",
      "description": "Phone number of skipper",
      "example": "+64xxxxxx"
    }
  },
  "required": ["dscAddress"],
  "additionalProperties": true
}
```

After a while, we add a new field, `callsignHf` :

```
{
  "type": "object",
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "https://signalk.org/demospecification/1.1.0/schemas/groups/communicat
  "description": "Schema describing the communication child-object of a Vessel
  "title": "communication",
  "properties": {
    "dscAddress": {
      "type": "string",
      "description": "MMSI Callsign for VHF communication"
    },
    "phoneNumber": {
      "type": "string",
      "description": "Phone number of skipper",
      "example": "+64xxxxxx"
    },
    "callsignHf": {
      "type": "string",
      "description": "Callsign for HF communication",
      "example": "ZL3RTH"
    }
  },
  "required": ["dscAddress"],
  "additionalProperties": true
}
```

Will this new schema validate all historical data? Unfortunately we can't be certain, because there could be historical JSONs where the analyst added their own `callsignHf` field which was not a string.

So we are effectively making a `REVISION` to the data schema - so we bump the version to `1.1.0` (resetting `ADDITION` to `0`).

Model

Oh dear - we have just realized that not every-one has DSC! It should have been a VHF callsign. Here is our new JSON Schema:

```
{
  "type": "object",
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "https://signalk.org/demospecification/2.0.0/schemas/groups/communicat
  "description": "Schema describing the communication child-object of a Vessel
  "title": "communication",
  "properties": {
    "callsignVhf": {
      "type": "string",
      "description": "Callsign for VHF communication",
      "example": "ZL1234"
    },
    "phoneNumber": {
      "type": "string",
      "description": "Phone number of skipper",
      "example": "+64xxxxxx"
    },
    "callsignHf": {
      "type": "string",
      "description": "Callsign for HF communication",
      "example": "ZL3RTH"
    }
  },
  "required": ["callsignVhf"],
  "additionalProperties": false
}
```

We have changed our `MODEL` - because we can have no reasonable expectation that any of the historical data can interact with this schema. That means our new version is `2.0.0`

Note that we also decided to use this “reboot” of the `MODEL` to change `additionalProperties` back to `false`, because (as we have learnt) it will help us to avoid unnecessary REVISIONs in the future.

Note: https://signalk.org/demospecification/... is not real, it is just used here for illustration. The real schemas are located at: https://signalk.org/specification/...

Supplementary rules

In Signal K we have a few variations from SchemaVer:

- We use dots (.) to separate the version parts, not hyphens (-s) as in SchemaVer
- We use a suffix to denote in-progress or special releases, as commonly seen in Maven

Access Requests

When a device needs to gain access to a secured Signal K server, it can use **Access Requests** to request and be granted access to the server.

See Request/Response for more information on request/response semantics.

A device could be a display or for example an engine sensor or a temperature sensor.

Definitions

- `clientId` is a string that uniquely identifies a device. It must be a v4 UUID The client should use the same value for all its requests.
- `requestId` is a string generated by the server in response to an access request and used by the client to correlate the request with the results. The client should not rely on the format or contents of this string.

Device Requests

The device will send a REST request to the server:

```
$ curl -k \
    --header "Content-Type: application/json" \
    --request POST \
    --data '{"clientId":"1234-45653-343453","description":"My Awesome Humidity
    https://localhost:3443/signalk/v1/access/requests
```

If the server does not support access requests it must return HTTP status code 501 Not Implemented.

For a successfully received access request the server must return an HTTP status code 202 and a JSON response with an href to check the status and get the response.

```
{
  "state": "PENDING",
  "href": "/signalk/v1/access/requests/358b5f32-76bf-4b33-8b23-10a330827185"
}
```

The server should then provide a process for an administrator to review and approve or deny the request.

In the meantime, a device should poll the server using the `requestId` in the response above to see if it has been granted access and get the token.

Response to a Pending Request

```
$ curl -k https://localhost:3443/signalk/v1/access/requests/358b5f32-76bf-4b33
{
  "state": "PENDING"
}
```

Response to a Denied Request

```
$ curl -k https://localhost:3443/signalk/v1/access/requests/358b5f32-76bf-4b33
{
  "state": "COMPLETED",
  "statusCode": 200,
  "accessRequest": {
    "permission": "DENIED"
  }
}
```

When a device gets a denied response, it should refrain from sending further access requests until the device is reset, rebooted or the user takes some action.

Response to an Approved Request

Note: The `expirationTime` property is optional.

```
$ curl -k https://localhost:3443/signalk/v1/access/requests/358b5f32-76bf-4b33
{
  "state": "COMPLETED",
  "statusCode": 200,
  "accessRequest": {
    "permission": "APPROVED",
    "token": "eyJhbGciOiJIUzI1NiIs...BAP8bt3tNBT1WiIttm3qM",
    "expirationTime": "2018-09-20T16:51:31.350Z"
  }
}
```

Response Indicating an Error With the Request

```
$ curl -k https://localhost:3443/signalk/v1/access/requests/358b5f32-76bf-4b33
{
  "state": "COMPLETED",
  "statusCode": 400,
  "message": "A device with clientId '1234-45653-343453' has already requested
}
```

After Access Approval

On approval, the device would save the token in a secure way and use it when sending or requesting data.

At some point in the future the provided token could expire, access to the server could be revoked or the server could be replaced. In all cases the server will respond to requests with a 403 status code. The device should then submit a new request for access and follow the process defined above.