Contents

1 Overview

This document details the GoFree web interface provided by Navico Multifunctional Displays (MFD). The GoFree web interface is designed to allow developers to create web pages and applications that use data from the Navico system. Additionally the web interface can be used to change settings on the Navico system and control some functions.

The GoFree web interface is provided by an HTTP server and a WebSocket server.

The WebSocket server is used to bi-directionally send data between the Client (Web Browser, IPad App, PC Application, etc) and a selected Navico device. The WebSocket data is encoded in simple JSON strings, which are very easy to work with on the client side. The basic format of any data is:

```
{“ObjectType” : {OBJECT DATA} }
```

2 Service Discovery

2.1 UDP Multicast

Mobile devices wishing to connect can listen to multicast address 239.2.1.1, port 2052 to discover the IP address and port of MFDs that support the data bridging function.

Any MFD will send the following JSON string to this multicast address:

{ “Name” : “*Name*”, “IP”: “*N.N.N.N*”, “Model” : “*Model*”, “Services” : [{ “Service” : “*Service*” , “Version”: *N* , “Port”: *N* }] }

Where:

<*Name*> The friendly name of the MFD. If this is changed on the MFD the multicast will change to match.

<*IP*> IP Address of the MFD. This will be obtained from a DHCP server if one is available, otherwise it will be a Zeroconfig address. Note that if a DHCP server is added to the network after power on the IP address announced will switch from Zeroconfig to DHCP.

<*Model*> Model name of the MFD (e.g. "NSS-12").

<*Service*> Describes the type of data services that the MFD supports. For MFDs that support websockets, this will be “navico-nav-ws”.

<*Version*> Version of the data protocol.

<*Port*> Port number used to transmit the data (e.g. 443).

MFDs will send this message at a rate of 1Hz. If this message is not received for a period of 2 second or longer, clients should consider the MFD to be unavailable, and attempt to connect to a different MFD for data.

A code example is included in the appendices.

2.2 Bonjour

The server will be announced via the Bonjour protocol as a TCP service called “navico-nav-ws” on the port used to transmit the data (e.g. 443). This is automatically supported in OS X and iOS, and can be added to Windows by installing Bonjour for Windows:

http://support.apple.com/kb/DL999

3 Web Socket

A WebSocket is a web technology which provides a bi-directional, full-duplex communication channel over a TCP connection. The WebSocket API is being standardised by the WSC and the WebSocket protocol has been standardised by the IETF as RFC 6455.

WebSockets have been designed to be implemented in web browsers and web servers. However they can be used by a client or server application. They provide a standardised way for the server to send content to the browser without it being requested by the client. This is therefore useful for sending live real-time data to a client as required in the Navico system without the client being required to poll for the data.

As it is bi-directional it also provides a convenient way for the client to send commands to the server.

Communications are done over the regular TCP port number 443. The WebSocket protocol is currently supported in several browsers including Safari, Firefox, Chrome and Internet Explorer 10. More information on browser support can be found at http://en.wikipedia.org/wiki/WebSocket#Browser_support .

In an HTML file the standard client side scripting language is JavaScript.

Connecting to a WebSocket server is as simple as:

```
websocket = new WebSocket( WsUri );
```

The GoFree SDK includes a JavaScript library that contains basic functions for WebSocket control.

A complete function to obtain full WebSocket functionality can be found in Appendix A.

4 JSON

NOTES:

- JSON Basics: “{}” surrounds an object, “[]” surrounds an array.
- S2C = Server to Client
- C2S = Client to Server

5 Digital Data

Digital Data is any data available in the Navico system that can be displayed on the Info panel on a Navico MFD. This includes data like boat speed, engine RPM and heading. It doesn’t include high bandwidth data such as radar or sonar images.

The WebSocket client can get information about what digital data is available, information about an individual digital data value and get an individual digital data value. It can also register to get digital data values in real time.

Each digital data value has a unique ID that is used in the JSON messages. These unique IDs are detailed in Section 12. They can also be obtained from the server using some of the commanded detailed below.

This section details the JSON commands and responses used for digital data.

5.1 Requesting a list of data IDs - C2S

The client can request a list of data IDs. Digital data values are grouped so a client can request all the data IDs in a group (e.g. request all engine data). The groups are detailed in Section 12.

```
{"DataListReq":{"groupId":1}}
```

group = The group of the data IDs being requested, in this case 1.

5.2 Send a list of data IDs - S2C

The server will send a list of Data IDs in response to a data list request (see 5.1)

```
{"DataList":{"groupId":0,"list":[40,41,42,43,44,45]}}
```

Data IDs, Format examples and unit information can be found in Section 12.

5.3 Request full information about a data value - C2S

The client can request information about a data value, using its data ID.

```
{"DataInfoReq":[40]}
```

40 in the example above is the ID of the data requested. The client can request information about many data values at the same time.

5.4 Send full information about a data value - S2C

The server will send information about a data value in response to a data info request (see 5.3).

```
{"DataInfo":[{ "id":40,"sname":"SOG","lname":"Speed Over
Ground","unit":"Knts","min":0,"max":99.9,"numInstances":2,
"instanceInfo":[{"inst":0,"location":0,"str":"Port"},{"inst":1,"location":3,"str":"
Starboard"}]
}]}
```

id = ID of the data value
sname = Short name of the data value in the currently selected language
lname = Long name of the data value in the currently selected language
unit = Units. These will be the currently selected system units.
min = Minimum possible value.
max = Maximum possible value.
numInstances = The number of instances of the data
instanceInfo = Information about specific instances
instanceInfo.inst = the instance Id
instanceInfo.location = the location ID of the instance
instanceInfo.str = the human readable instance name

All of the values other than ID are optional, and more may be added.

5.5 Requesting a value – C2S

The client can request a data value.

```
{"DataReq":[{"id":1,"repeat":false,"inst":0}]}
```

id = The ID of the data
repeat = Should the data repeat, or be sent only once. This is an optional parameter that defaults to false. If set to true then the server will periodically send the data to the client whilst the WebSocket connection remains open. The update rate is set by the server.
Inst = The data instance. This is an optional parameter that defaults to 0. It is used for values such as Engine RPM which can have more than one instance.

5.6 Unsubscribing – C2S

The client can unsubscribe from a repeating data value.

```
{"UnsubscribeData":[{"id":1,"inst":0}]}
```

Id = The ID of the data
Inst = The data instance. This is an optional parameter that defaults to 0.

5.7 Sending array of values – S2C

The server will send values either when they are requested by a data request (see 5.5) or periodically if the data was requested with repeat set to true.

```
{"Data":[{"id":1,"val":10.4,"valStr":"10.4","sysVal":5.62,"inst":0,"valid":true},{"
id":2,"valid":false}]}
```

This example shows two pieces of data. IDs 1 and 2, Values 10.4 and 90 respectively:

id = The ID of the data
val = The value of the data in currently selected user units.
valStr = The value of the data in currently selected user units as a correctly formatted string
sysVal = The value in system units. Standard system units can be found in section 12.3
inst = The data instance. This is an optional parameter that defaults to 0.
valid = is the data valid?
Note that there is no limit on the number of data values between [and].

6 Settings

Selected settings in the Navico system can be accessed and changed through the WebSocket.

The WebSocket client can get information about what settings are available, information about an individual setting, adjust a setting and get notified if a setting changes.

Each setting has a unique ID that is used in the JSON messages. These unique IDs are detailed in Section 13.3. They can also be obtained from the server using some of the commanded detailed below.

This section details the JSON commands and responses used for settings.

6.1 Request a settings group - C2S

The client can request a list of setting IDs. Settings are grouped into categories so a client can request all the setting IDs in a group (e.g. request all alarm settings). The groups are detailed in Section 13.2.

```
{"SettingListReq":[{"groupId":1}]}
```

group = The group of the setting IDs being requested, in this case 1.

6.2 Send a list of setting IDs - S2C

The server will send a list of setting IDs in response to a setting list request (see 6.1)

```
{"SettingList":{"groupId":1,"list":[40,41,42,43,44,45]}}
```

Setting IDs can be found in Section 13.1

6.3 Request setting information - C2S

The client can request information about a setting, using it's setting ID.

```
{"SettingInfoReq":[40]}
```

40 in the example above is the ID of the setting requested. The client can request information about many settings at the same time.

6.4 Send setting information - S2C

The server will send information about a setting in response to a setting info request (see 6.3).

```
{"SettingInfo":[{"id":40, "name":"Boat Type", "type":1, "values":[{"id":0,
"title":"Sailing"}, {"id":1,"title":"Fishing"}, {"id":2, "title":"Planing"}]}]}
```

This example shows a enumerated setting with the possible values.
id = The ID of the setting.
name = Text name of the setting in the currently selected language.
type = The type of the setting. 1 indicates an enumerated type (see section 13.1)
Values = Array of possible values.

```
{"SettingInfo":[{"id":1,"name":"Backlight Level","type":2,"low":0,"high":10}]}
```

This example shows a numeric setting with the minimum and maximum possible values.
id = The ID of the setting.
name = Text name of the setting in the currently selected language.
type = The type of the setting. 2 indicates a numeric type (see section 13.1).
low = The lowest possible value.
high = The highest possible value.

```
{"SettingInfo":[{"id":2,"name":"Night Mode","type":3}]}
```

This example shows a boolean setting.
id = The ID of the setting.
name = Text name of the setting in the currently selected language.
type = The type of the setting. 3 indicates a boolean type (see section 13.1).

6.5 Request a list of settings - C2S

The client can request a setting value.

```
{"SettingReq":{"ids":[1,2,3],"register":false}}
```

id = IDs of settings to request. This can contain any number of IDs.
register = Register for updates. If register is true, the client will receive updates for those IDs. This is optional and will default to false.

6.6 Unsubscribe from a list of settings

The client can unsubscribe from settings to which it has registered.

```
{"UnsubscribeSetting":[1,2,3]}
```

6.7 Send a setting- S2C and C2S

The server will send settings either when they are requested by a setting request (see 5.5) or if they change if they were requested with register set to true.

The client will send this command to change a setting value.

```
{"Setting":[{"id":1, "value":true}]}
```

id = The ID of the setting
value = The value of the setting. This will be whatever is relevant for a specific setting.

Note that there is no limit on the number of setting values between [and].

7 Events

Selected events in the Navico system can be received and sent through the WebSocket.

Each event has a unique ID that is used in the JSON messages. These unique IDs are detailed in Section 14.1.

This section details the JSON commands and responses used for events.

7.1 Register to receive an event - C2S

The client can register to receive an event from the server.

```
{"EventReg":[1,2]}
```

The elements of the array are event IDs (see section 14.1).

7.2 Unregister from an event – C2S

The client can unregister from an event.

```
{"EventUnreg":[1,2]}
```

The elements of the array are event IDs (see section 14.1).

7.3 MOB Event – C2S and S2C

The server or the client can set an event.

```
{"EventSet":[{"id":5,"active":true}]}
```

Or

```
{"EventSet":[{"id":5,"name":"MOB","active":true}]}
```

This event will look the same being sent client to server (e.g. raising the alarm) or server to client (e.g. telling all nodes that an event has been raised).

When MOB is cancelled the same message will be sent, but with “active”:false. MOB cannot currently be cancelled from the GoFree Web Interface.

7.4 Create a Waypoint – C2S and S2C

```
{"EventSet":[{"id":5, "latitude":50.9892, "longitude":-1.4975,
“name”:“Waypoint1”}]}
```

You can set a waypoint with the following parameter combinations:
latitude, longitude
latitude, longitude, wpName
No parameters – this will drop a waypoint with an automatically generated name at the current boat location.

When a waypoint is created in the system it will also contain a “wpId” parameter. This should be used to track changes and deletions. NOTE: Currently waypoint changes are only supported server to client. Deletions are not yet supported in any way.

Activate/Silence/Acknowledge/Deactivate an alarm – C2S and S2C

```
{"EventSet":[{"id":10, "alarmId":5, "alarmText":"Low Speed","severity":1}]}
```

ID is the event ID. There will be a separate ID for each alarm action. Severity and AlarmText are optional and are unnecessary for a client when silencing an alarm.
Alarms cannot be activated from GoFree.

7.5 Reset a Trip Log – C2S

```
{"EventSet":[{"id":7, "inst":0}]}
```

ID is the event ID.
Inst is the trip log instance (currently 0 or 1, for “Trip 1” and “Trip 2” respectively)

8 Vessels

Vessel information can be received through the WebSocket. A vessel can currently only be an AIS target. The client can register to receive updates when a vessel of each type is changed.

This section details the JSON commands and responses used for vessels.

8.1 Register for vessels – C2S

The client can register to receive vessel information from the server.

```
{"TrafficReq":{"subscribe":true,"id":12345,"sourceType":0 } }
```

sourceType is described in section 15.1 and is compulsory
subscribe - if true then all updates of sourceType will be sent to the client. This is optional and defaults to false. It is not possible to subscribe to a single vessel ID.
id is vessel ID (mmsi for AIS). It is optional. If not included all data of “sourceType” will be sent.

8.2 Register for vessels – C2S

The client can register to receive all traffic updates of a specified type.

```
{"TrafficReg":[0,2]}
```

The elements of the array are vessel type IDs (see section 15.1).

8.3 Unregister for vessels – C2S

The client can unregister to stop receiving vessel information from the server.

```
{"TrafficUnreg":[0,2]}
```

The elements of the array are vessel type IDs (see section 15.1).

8.4 Vessel Data – S2C

The server will send vessel data each time a vessel is updated to any client that has registered to receive updates for that type of vessel.

```
{"Traffic":[{"type":1,"id":40,"name":"vessel name","lost":false,"distance":10.2,
"lat":50.9892,"lon":-
1.4975,"trueBearing":18.3,"cpa":1.54,"tcpa":245.2,"sog":3.8,"cog":195.5,"relative
sog":5.0,"relative cog":210.1,"status":2}]}
```

type = Type of vessel e.g. AIS (see section 15.1).
id = Vessel ID which is the MMSI of the vessel for AIS and DSC vessel types.
name = Name of the vessel
lost = Indicates if the vessel has been lost
distance = Distance of the vessel from own vessel in current distance units
lat = Latitude of the vessel in degrees
lon = Longitude of the vessel in degrees
trueBearing = True bearing of the vessel in degrees
cpa = Closest point of approach to own vessel in current distance units
tcpa = Time to the closes point of approach in seconds
sog = Speed over ground of the vessel in current speed units
cog = Course over ground of the vessel in degrees
relativeSog = Relative speed over ground of the vessel in current speed units. This is the speed relative to own vessel.
relativeCog = Relative course over ground of the vessel in degrees. This is the course relative to own vessel.
status = Status of the vessel (see section 15.2 for values).

Other available data include eta, draught, destination, status, isAton, atonType, dimensions { bow, stern, port, stbd}, accuratePosition, aisClass, callsign, imo, sourceType, trueHeading and offPosition.

Only type and id are guaranteed to be present.

8.5 Vessel Removed – S2C

A client subscribed to a traffic source will be notified if a vessel is removed.

```
{"TrafficRemoved":[{"sourceType":0, "id":12345}]}
```

sourceType is described in section 15.1 and is compulsory
id is vessel ID (mmsi for AIS). It is optional. If not included all data of “sourceType” will be sent.

9 Other WebSocket Functionality

9.1 Grouping many instructions

Many instructions can be bundled as one message using the format below.

```
{"Many":[{INSTRUCTION1},{INSTRUCTION2}]}
```

e.g.

```
{"Many":[{"EventReg":[1,2]},{"TrafficReg":[0,2]}]}
```

10 Appendix A – Service Discovery

Sample C# code for service discovery. This uses the open source plugin Json.NET (http://www.codeplex.com/json/) for simplicity.

```
using System;
using System.Linq;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;
using Newtonsoft.Json;
using System.Diagnostics;

namespace NavicoDiscovery
{
    class Program
    {
        public class MFDService
        {
            public MFDService(string service, uint version, uint port)
            {
                Service = service;
                Version = version;
                Port = port;
            }

            public string Service;
            public uint Version;
            public uint Port;
        }

        public class MFD
        {
            public string Name;
            public string IP;
            public string Model;
            public MFDService[] Services;
        }

        static readonly object _locker = new object();

        private static void ReceiveOldMessage()
        {
            // Navico NSS 2.0: Multicast 239.2.1.1, port 2050
            UdpClient client = new UdpClient();

            IPEndPoint localEp = new IPEndPoint(IPAddress.Any, 2050);
            client.Client.Bind(localEp);

            IPAddress multicastaddress = IPAddress.Parse("239.2.1.1");
            client.JoinMulticastGroup(multicastaddress);

            string output;
            while (true)
            {
```

```
                Byte[] data = client.Receive(ref localEp);
                if (data.Length > 0)
                {
                    string strData = Encoding.ASCII.GetString(data);
                    char[] lineEnd = { '\r', '\n' };
                    strData = strData.TrimEnd(lineEnd);
                    string[] strParams = strData.Split(',');
                    output = string.Format("Service: {0}\tVersion: {1}\tIP: {2}\tPort: {3}",
strParams[0], strParams[1], strParams[2], strParams[3]);
                    lock (_locker)
                    {
                        Console.WriteLine(output);
                        Debug.Print(output + string.Format("\r\n"));
                    }
                }
            }
        }

        private static void ReceiveNewMessage()
        {
            // Navico NSS 2.5: Multicast 239.2.1.1, port 2052
            UdpClient client = new UdpClient();

            IPEndPoint localEp = new IPEndPoint(IPAddress.Any, 2052);
            client.Client.Bind(localEp);

            IPAddress multicastaddress = IPAddress.Parse("239.2.1.1");
            client.JoinMulticastGroup(multicastaddress);

            string output;
            while (true)
            {
                Byte[] data = client.Receive(ref localEp);
                if (data.Length > 0)
                {
                    string strData = Encoding.ASCII.GetString(data);
                    MFD deserializedMFD = JsonConvert.DeserializeObject<MFD>(strData);

                    output = string.Format("MFD: {0}\tModel: {1}\tIP: {2}", deserializedMFD.Name,
deserializedMFD.Model, deserializedMFD.IP);
                    lock (_locker)
                    {
                        Console.WriteLine(output);
                        Debug.Print(output);
                        for (uint service = 0; service < deserializedMFD.Services.Count(); service++)
                        {
                            output = string.Format("\tService {0}: {1}\tVersion: {2}\tPort: {3}",
service, deserializedMFD.Services[service].Service, deserializedMFD.Services[service].Version,
deserializedMFD.Services[service].Port);
                            Console.WriteLine(output);
                            Debug.Print(output);
                        }
                    }
                }
            }
        }
```

```
        static void Main(string[] args)
        {
            Thread oldThread;
            oldThread = new Thread(new ThreadStart(ReceiveOldMessage));
            oldThread.IsBackground = true;
            oldThread.Start();
            Thread newThread;
            newThread = new Thread(new ThreadStart(ReceiveNewMessage));
            newThread.IsBackground = true;
            newThread.Start();

            while (true) { }
        }
    }
}
```

11 Appendix B - Websocket

The code below is an example of how to connect to a WebSocket from a client using JavaScript. The functions that are called within event handlers will need to be implemented elsewhere.

```
function initWebsocket(WsUri)
{
    if ("WebSocket" in window)
    {
        try
        {
            if (typeof MozWebSocket == 'function')
            {
                WebSocket = MozWebSocket;
            }
            websocket = new WebSocket(WsUri);

            websocket.onopen = function (evt)
            {
                onWebSocketConnect();
            };
            websocket.onclose = function (evt)
            {
                onDisconnect();
            };
            websocket.onmessage = function (evt)
            {
                handleMessage(evt.data);
            };
            websocket.onerror = function (evt)
            {
                debug('ERROR: ' + evt.data, 1);
            }
            catch (exception)
            {
                debug('ERROR: ' + exception, 1);
            }
        }
```

```
        else
        {
            alert("I don't think this browser supports WebSockets");
        }
    }
}
```

There is currently a bug in Safari for IOS that causes what look like buffer overruns. To work around this issue it is necessary to handle messages as below:

```
websocket.onmessage = function (evt) {
    setTimeout(function() { handleMessage( evt.data );}, 0);
};
```

12 Appendix C

12.1 Data Groups

Data Group	ID
GPS	1
Navigation	2
Vessel	3
Sonar	4
Weather	5
Trip	6
Time	7
Engine	8
Transmission	9
Fuel Tank	10
Fresh Water Tank	11
Gray Water Tank	12
Live Well Tank	13
Oil Tank	14
Black Water Tank	15
Engine Room	16
Cabin	17
Bait Well	18
Refrigerator	19
Heating System	20
Freezer	21

Data Group	ID
Battery	22
Rudder	23
Trim Tab	24
AC Input	25
Digital Switching	26
Other	27
GPS Status	28
Route Data	29
Speed Depth	30
Log Timer	31
Environment	32
Wind	33
Pilot	34
Sailing	35
AC Output	36
Charger	37
Inverter	38

12.2 Data Types

Note : Not all multifunction displays or systems will support all the data types listed below.

Data Type	ID	Group
Altitude	1	GPS
Position Error	3	GPS
HDOP	4	GPS Status

Data Type	ID	Group
VDOP	5	GPS Status
TDOP	6	GPS Status
PDOP	7	GPS Status
Geoidal Seperation	8	GPS Status
COG	9	GPS
Position Quality	10	GPS Status
Position Integrity	11	GPS Status
Satellites In View	12	GPS Status
WAAS Status	13	GPS Status
Bearing (To Waypoint)	14	Navigation
Bearing Origin to Waypoint	15	Navigation
Course To Steer	17	Navigation
Cross Track Error	18	Navigation
VMG to Waypoint	19	Navigation
Destination	20	Navigation
Distance to Turn	21	Navigation
Distance to Destination	22	Navigation
Time To Turn	23	Route Data
Time To Destination	24	Route Data
ETA At Turn	25	Route Data
ETA At Destination	26	Route Data
Total Distance	27	Route Data

Data Type	ID	Group
Steer Arrow	28	Navigation
Odometer	29	Trip
Trip Distance	30	Trip
Trip Time	31	Trip
Local Date	32	Time
Local Time	33	Time
UTC Date	34	Time
UTC Time	35	Time
Local Time Offset	36	Time
Heading	37	Vessel
Voltage	38	Other
Current Set	39	Other
Current Drift	40	Other
SOG	41	GPS
Water Speed	42	Sonar
Pitot Speed	43	Vessel
Average Trip Speed	44	Trip
Maximum Trip Speed	45	Trip
Apparent Wind Speed	46	Weather
True Wind Speed	47	Weather
Water Temperature	48	Sonar
Outside Temperature	49	Weather

Data Type	ID	Group
Inside Temperature	50	Vessel
Engine Room Temperature	51	Engine Room
Main Cabin Temperature	52	Cabin
Live Well Temperature	53	Live Well Tank
Bait Well Temperature	54	Bait Well
Refrigeration Temperature	55	Refrigerator
Heating System Temperature	56	Heating System
Dew Point Temperature	57	Weather
Apparent Wind Chill Temperature	58	Weather
Theoretic Wind Chill Temperature	59	Weather
Heat Index Temperature	60	Weather
Freezer Temperature	61	Freezer
Engine Temperature	62	Engine
Engine Air Temperature	63	Engine
Engine Oil Temperature	64	Engine
Battery Temperature	65	Battery
Atmospheric Pressure	66	Weather
Engine Boost Pressure	67	Engine
Engine Oil Pressure	68	Engine
Engine Water Pressure	69	Engine
Engine Fuel Pressure	70	Engine
Engine Manifold Pressure	71	Engine

Data Type	ID	Group
Stream Pressure	72	Other
Compressed Air Pressure	73	Other
Hydraulic Pressure	74	Other
Depth	77	Sonar
Water Distance	78	Sonar
Engine RPM	79	Engine
Engine Trim	80	Engine
Engine Alternator Potential	81	Engine
Engine Fuel Rate	82	Engine
Engine Percent Load	83	Engine
Engine Percent Torque	84	Engine
Suzuki Alarm Level Low	85	Engine
Suzuki Alarm Level High	86	Engine
Fuel Tank Level	87	Fuel Tank
Fresh Water Fluid Level	88	Fresh Water Tank
Gray Water Fluid Level	89	Gray Water Tank
Live Well Fluid Level	90	Live Well Tank
Oil Fluid Level	91	Oil Tank
Black Water Fluid Level	92	Black Water Tank
Fuel Remaining	93	Fuel Tank
Fresh Water Fluid Volume	94	Fresh Water Tank
Gray Water Fluid Volume	95	Gray Water Tank

Data Type	ID	Group
Live Well Fluid Volume	96	Live Well Tank
Oil Fluid Volume	97	Oil Tank
Black Water Fluid Volume	98	Black Water Tank
Generic Fluid Volume	99	Unconfigured
Generic Tank Capacity	105	Unconfigured
Fuel Tank Capacity	106	Fuel Tank
Fresh Water Tank Capacity	107	Fresh Water Tank
Gray Water Tank Capacity	108	Gray Water Tank
Live Well Tank Capacity	109	Live Well Tank
Oil Tank Capacity	110	Oil Tank
Black Water Tank Capacity	111	Black Water Tank
Tank Fuel Used	112	Fuel Tank
Engine Fuel Used	113	Engine
Trip Fuel Used	114	Engine
Seasonal Fuel Used	115	Engine
K Value	116	Engine
Battery Potential	117	Battery
Battery Current	118	Battery
Trim Tab	119	Trim Tab
Rate Of Turn	121	Vessel
Yaw	122	Vessel
Pitch	123	Vessel

Data Type	ID	Group
Roll	124	Vessel
Magnetic Variation	125	Other
Deviation	126	Other
Water Fuel Economy	127	Engine
GPS Fuel Economy	128	Engine
Water Fuel Range	130	Engine
GPS Fuel Range	131	Engine
Engine Hours Used	132	Engine
Engine Type	133	Engine
Vessel Fuel Rate	134	Vessel
Vessel Water Fuel Economy	135	Vessel
Vessel GPS Fuel Economy	136	Vessel
Vessel Fuel Remaining	137	Vessel
Vessel Water Fuel Range	138	Vessel
Vessel GPS Fuel Range	139	Vessel
Apparent Wind Angle	140	Weather
True Wind Angle	141	Weather
True Wind Direction	142	Weather
Inside Humidity	143	Vessel
Outside Humidity	144	Weather
Set Humidity	145	Vessel
Rudder Angle	146	Rudder

Data Type	ID	Group
Transmission Gear	147	Transmission
Transmission Oil Pressure	148	Transmission
Transmission Oil Temperature	149	Transmission
Commanded Rudder Angle	150	Rudder
Rudder Limit	151	Rudder
Off Heading Limit	152	Vessel
Radius of Turn Order	153	Vessel
Rate of Turn Order	154	Vessel
Off Track Limit	155	Vessel
Logging Time Remaining	156	Other
Position Fix Type	157	GPS Status
Engine Discrete Status	158	Engine
Transmission Discrete Status	159	Transmission
GPS Best of Four Signal to Noise Ratio	160	GPS Status
Generic Fluid Level	161	Unconfigured
Generic Pressure	162	Unconfigured
Generic Temperature	163	Unconfigured
Internal Voltage	164	Other
Structure Depth	166	Sonar
Loran Position	167	Vessel
Vessel Status	168	Vessel
Battery DC Type	169	Battery

Data Type	ID	Group
Battery State of Charge	170	Battery
Battery State of Health	171	Battery
Battery Time Remaining	172	Battery
Battery Ripple Voltage	173	Battery
AC Input 1 Quality	174	AC Input
AC Input 2 Quality	175	AC Input
AC Input 3 Quality	176	AC Input
AC Input 1 Voltage	177	AC Input
AC Input 2 Voltage	178	AC Input
AC Input 3 Voltage	179	AC Input
AC Input 1 Current	180	AC Input
AC Input 2 Current	181	AC Input
AC Input 3 Current	182	AC Input
AC Input 1 Frequency	183	AC Input
AC Input 2 Frequency	184	AC Input
AC Input 3 Frequency	185	AC Input
AC Input 1 Breaker Size	186	AC Input
AC Input 2 Breaker Size	187	AC Input
AC Input 3 Breaker Size	188	AC Input
AC Input 1 Real Power	189	AC Input
AC Input 2 Real Power	190	AC Input
AC Input 3 Real Power	191	AC Input

Data Type	ID	Group
AC Input 1 Reactive Power	192	AC Input
AC Input 2 Reactive Power	193	AC Input
AC Input 3 Reactive Power	194	AC Input
AC Input 1 Power Factor	195	AC Input
AC Input 2 Power Factor	196	AC Input
AC Input 3 Power Factor	197	AC Input
Switch State	198	Digital Switching
Switch Current	199	Digital Switching
Switch Fault	200	Digital Switching
Switch Dim Level	201	Digital Switching
Previous Commanded Heading	202	Pilot
Commanded Wind Angle	203	Pilot
Commanded Bearing Offset	204	Pilot
Commanded Bearing	205	Pilot
Commanded Depth Contour	206	Pilot
Commanded Course Change	207	Pilot
Pilot Drift	208	Pilot
Pilot Distance To Turn	209	Pilot
Pilot Time To Turn	210	Pilot
Pilot Reference Position	211	Pilot
DC Status	212	Battery
AC Input 1 Status	213	AC Input

Data Type	ID	Group
Switch Voltage	214	Digital Switching
Battery Capacity Remaining	215	Battery
H3000 Linear 1	217	Sailing
H3000 Linear 2	218	Sailing
H3000 Linear 3	219	Sailing
Boom Position	220	Sailing
Sailing Course	221	Sailing
Daggerboard Position	222	Sailing
H3000 Linear 4	223	Sailing
Heading on Next Tack	224	Sailing
Keel Angle	225	Sailing
Leeway	226	Sailing
Mast Angle	227	Sailing
Target True Wind Angle	228	Sailing
Keel Trim Tab	229	Sailing
Race Timer	230	Sailing
Canard Angle	231	Sailing
Next Leg Apparent Wind Angle	232	Sailing
Next Leg Apparent Wind Speed	233	Sailing
Target Boat Speed	234	Sailing
VMG to Wind	235	Sailing
Time to Layline	236	Sailing

Data Type	ID	Group
Distance to Layline	237	Sailing
Aft Depth	238	Sonar
Fore Stay	239	Sailing
Polar Speed	240	Sailing
Polar Performance	241	Sailing
Tacking Performance	242	Sailing
Wind Angle To Mast	243	Sailing
CAN Bus Voltage	244	Other
Internal Temperature	245	Other
Engage Current	246	Other
URef Voltage	247	Other
Supply Voltage	248	Other
Destination Position	249	Navigation
Engine Sync State	252	Engine
Engine Predictive General Maintenance	253	Engine
Engine Throttle	254	Engine
Engine Steering Angle	255	Engine
Engine Break In Required	256	Engine
Engine Break In Remaining	258	Engine
Engine Trim Status	259	Engine
Autopilot Present	260	Pilot
AC Output 1 Quality	261	AC Output

Data Type	ID	Group
AC Output 2 Quality	262	AC Output
AC Output 3 Quality	263	AC Output
AC Output 1 Voltage	264	AC Output
AC Output 2 Voltage	265	AC Output
AC Output 3 Voltage	266	AC Output
AC Output 1 Current	267	AC Output
AC Output 2 Current	268	AC Output
AC Output 3 Current	269	AC Output
AC Output 1 Frequency	270	AC Output
AC Output 2 Frequency	271	AC Output
AC Output 3 Frequency	272	AC Output
AC Output 1 Breaker Size	273	AC Output
AC Output 2 Breaker Size	274	AC Output
AC Output 3 Breaker Size	275	AC Output
AC Output 1 Real Power	276	AC Output
AC Output 2 Real Power	277	AC Output
AC Output 3 Real Power	278	AC Output
AC Output 1 Reactive Power	279	AC Output
AC Output 2 Reactive Power	280	AC Output
AC Output 3 Reactive Power	281	AC Output
AC Output 1 Power Factor	282	AC Output
AC Output 2 Power Factor	283	AC Output

Data Type	ID	Group
AC Output 3 Power Factor	284	AC Output
AC Input 2 Status	285	AC Input
AC Input 3 Status	286	AC Input
AC Output 1 Status	287	AC Output
AC Output 2 Status	288	AC Output
AC Output 3 Status	289	AC Output
Switch Manual Override	290	Digital Switching
Switch Reverse Polarity	291	Digital Switching
Switch AC Source Available	292	Digital Switching
Switch AC Contactor System On State	293	Digital Switching
Charger Battery Instance	294	Charger
Charger Operating State	295	Charger
Charger Mode	296	Charger
Charger Enabled	297	Charger
Charger Equalization Pending	298	Charger
Charger Equalization Time Remaining	299	Charger
Inverter AC Instance	300	Inverter
Inverter DC Instance	301	Inverter
Inverter Operating State	302	Inverter
Inverter Enabled	303	Inverter
Sailing Time To Waypoint	304	Sailing
Sailing Distance To Waypoint	305	Sailing

Data Type	ID	Group
Sailing ETA	306	Sailing
Latitude	309	GPS
Longitude	310	GPS

12.3 System Units

Used for	Unit
Wind speed and boat speed	Knots
Distance	Nautical Miles
Depth	Feet
Temperature	Celsius
Altitude	Feet
Map units	Metres
Pressure	Inches of Mercury
Voltage	Voltage
Volume	Gallons
Volume rate	Gallons per Hour
Amps	Amps
Economy	Nautical miles / Gallon
Angle, Heading, Position	Radians
Angular Speed	Radians per second
Engine Speed	Revolutions per minute
Timer	Seconds
Percent	Percent

Temperature Rate	**Celsius per minute**
Acceleration	**Metres per second square**
Distance per revolution	**Inches per revolution**
No units	**No units**

13 Appendix D

13.1 Setting Types

Setting Type	ID
Enumeration	1
Boolean	2
Number	3

13.2 Setting Groups

Setting Group	ID
Display	1
Units	2
Alarms	3
Trip Log	4

13.3 Setting IDs

Setting Name	Group	Setting ID	Type	Possible Values
Backlight Level	Display	1	Number	0 -> 10
Night Mode	Display	2	Boolean	true, false
Trip Log 1 Enabled	Trip Log	3	Boolean	true, false
Trip Log 2 Enabled	Trip Log	4	Boolean	true, false

14 Appendix E

14.1 Events

Event Name	Group	ID	Data
Activate Alarm	Alarms	1	
Deactivate Alarm	Alarms	2	
Acknowledge Alarm	Alarms	3	
Silence Alarm	Alarms	4	
MOB	MOB	5	
Create Waypoint	Waypoint	6	
Reset Trip Log	Trip Log	7	
Zoom In	Display	8	
Zoom Out	Display	9	

15 Appendix F

15.1 Vessel Types

Event Group	ID
AIS	1

15.2 Vessel Status

Event Group	ID
Dangerous	1
Safe	2
Acquiring	3
Acquiring Failed	4
Out of Range	5

Event Group	ID
Lost Out of Range	6