

This document is an attempt to goad development of the polar function component of OckamSoft 4. The OckamSoft 4 polar server provides standard polar data (polar speed, target angle) to multiple clients. In the case of OS4, these include Racecourse, BIFs and the Simulator.

VPPers should have the capability to deploy as much polar technology as his customer needs/wants, without impacting client applications (i.e. keep Ockam out of the polar development loop). We want the VPPER to sell custom POLAR.DLL as well as a polar database if the customer needs it.

### **Philosophy**

Standard polars are a static function of true wind angle and speed, so they can't adapt to the conditions mentioned above. Sailors know when they can't achieve their targets, that they're sailing in a particular race with different crew weight or sail inventory, and that the performance on port tack is better, etc. In OckamSoft 2, this created a demand for hacks, for example the so-called "Windweight" factor (which adjusts the Vt input to the polar), AutoCal databases that monkeyed with Polar calibration based on sea state, and so on.

Making polars smarter requires code written by the VPPER, for which he must be compensated. Since his customer is the sailor (not Ockam), it follows that a custom PLRSVR.DLL should be sold by him and not by Ockam. By making the server public, Ockam gains by not having to write hacks, and having a program that produces better answers.

A market also has to develop for "smart" polars before VPPers can sell them, so OS4 comes with a replacable "stupid" polar server that uses the static database paradigm. Initially, VPPers could supply standard (OS2 compatible) polar worksheet which the customer can convert to a binary TPO file that the default server can use. A polar worksheet and binary (of Niña) and a conversion program (CVTPO.EXE) are included in the standard OS4 package (both demo and real).

### **The server API**

As of first release, the polar server includes the following calls;

- SelectPolar(hwnd) selects the polar database. In the standard server, this produces a file open dialog for \*.tpo.
- GetPolarInfo(\*Title,len,\*Descr,len,\*nCfg) retrieves the title (eg "polar.tpo"), description (eg "Niña © Velocity inc. 1995") current configuration name (eg "Base") and current configuration number.
- PolarSpeed(Vt,Bt,BtSign) returns the polar speed at the given wind speed and angle. BtSign is used to specify tack as follows; Assuming the database covers standard range (0 to 180°), the Bt used to access the database is |Bt|\*sign(BtSign)\*sign(Bt). This argument allows full 360° coverage for each tack (more important in TargAngle where conventional Bt can range over more than 180°).
- TargAngle(Vt,BtRhumb,BtSign) returns the angle of maximum Vmc relative to the rhumbline. For example, PolarSpeed(Vt,TargAngle(Vt,0,0),0) is the upwind maximum Vmg, while PolarSpeed(Vt,180+TargAngle(Vt,180,0),0) is the downwind maximum.
- SetGradient(k) is the gradient knob. For the default server, Vt is multiplied by (1+k) ie, "Windweight". What should the real units of k be for this function?

- SetShear(k) is the shear knob. For the default server, k is added to  $|Bt| \cdot \text{sign}(Bt \cdot \text{Sign}) \cdot \text{sign}(Bt)$ . The actual units of k need to be determined. Degrees/mastheight?
- SetSeaState(k) is the seastate knob. Not used in the default server. What should this k really look like??
- SetConfig(hwnd,CfgID,len) sets the configuration. If CfgID is a defined configuration name (eg "Base"), sets polar configuration to the saved state. If invalid, brings up a configuration dialog to specify a new configuration. In the default server, the only operational configuration parameter is K1, an overall polar cal number.
- EnumConfig(Index,\*CfgID,len) enumerates the configuration names.

Since sails are important to the polar function, the sail database has been included in the polar server. At present, OS4 has defined a sail information function designed for producing a sail selection screen a'la OckamSoft 2 (although this screen has not yet been written).

- EnumSails(Index,\*SailInfo,\*InCurCfg) returns a structure that contains [sail name, manufacturer, type, sail year, weight and wind range], and whether the given sail is in the current configuration. In the default driver, this information is contained in the plrsvr.ini file as text.