iNavCalc Command-Line Interface Users' Guide

Throughout this document, italicised text pertains to information specific to UNREGISTERED users for whom features and functionality are limited. Note: it is FREE to register.

Overview

iNavCalc is a comprehensive flight navigation planning tool which provides by **email** a complete PLOG (**P**ilot's **LOG**) for any desired route anywhere in the world. It is primarily intended for VFR flying (though may be applied to any type of flight), and provides navigation calculations for any user-specified aircraft/performance parameters such as IAS, altitude, fuel flow-rate, etc., which can be specified to be constant or to vary along the route. Specifically, the following are included in the generated PLOG , taking into account the initial climb-out, cruise, and en-route climb/descent portions of the flight: (i) navigation computations (heading, track, TAS, ground-speed) including wind correction per leg (using an automated global weather wind & temperature forecast and actual MET data which -- we call this feature "AutoMETic") and automated magnetic variation offsets (ii) time and distance calculations along the route, including the ability to "reset the stopwatch" at a specific waypoint other than the point of origin, and to specify time according to "my watch" (i.e., with specified UTC offset); (iii) fuel consumed/remaining along the route -- taking into account extra burn for initial climb-out, en-route climb, and reduced burn for en-route descent -- for user-specified fuel consumption parameters (separately specified for climb, cruise, descent, economy); (iv) endurance and remaining-range at each waypoint (for continuous diversion planning) for user-specified fuel capacity and consumption; (iv) altimetry computations (estimation of pressure altitude, transition level, minimum usable flight level) along the route using current reported atmospheric temperatures and pressures at MET stations automatically identified within the proximity of each waypoint; (v) estimation of density altitude and freezing altitude along the route using current reported atmospheric temperatures and pressures at MET stations; (vi) computation of solar incidence angles relative to the aircraft heading, for each point along the route, including day/night notification, plus length-of-day calculations. A "reverse route" feature enables the PLOG to be computed for the return journey at no extra effort.

Routes, waypoints, and performance settings can be stored for modification and re-use later by a given user, and can be shared with other users and/or with the public. As a further convenience, the key parameters used in the calculations and presented in the generated reports can be specified in user-preferred dimensional units (e.g., wind-speed in miles-per-hour

or knots, fuel capacity/usage in US gallons, UK gallons, or litres, pressure-settings in mb or in-Hg, temperature in celsius or fahrenheit, etc).

Command-line interface

All aspects of the PLOG generation can be controlled via parameters specified on the *command-line interface* as described in this document. The *command-line* in question can refer equivalently to any of the following:

- the *subject-line* of an email if you are using the **iNavCalc email app**
- the *Plan command-line* if you are using the **iNavCalc mobile app** or **web app**
- the Twitter *message-string* if you are using the **iNavCalc Twitter app**

For the remainder of this document, the **iNavCalc email app** will be most often referred to, but it should be understood that all instructions pertain to all of the *command-line* interfaces in the above list.

How do I specify routes and waypoints ?

Routes and waypoints can be entered/accessed in three ways: (i) selected from a (user-maintained) database of waypoints and routes stored online at FlyLogical, built-up over time with usage; (ii) via user-specified list of waypoint coordinates and/or radials (bearing/range) relative to other waypoints -- either true or magnetic bearings can be specified (magnetic variation automatically computed for location and epoch), and can be defined as either ‘great circles’ or ‘rhumb-lines’ (corresponding to constant true heading i.e., straight-lines drawn on a typical Mercator projection aeronautical chart) ; (iii) via open-standard GPX files provided as email attachments -- these files can be created by hand or generated automatically by other software products e.g., Memory-Map.

Note: when working with stored waypoints when constructing a route, the resulting route is decoupled from the stored waypoints once created. In other words, the stored waypoints are used to determine the coordinates of the route waypoints, but are not themselves bound-in to the route -- only the location information (originally derived from the stored waypoints) is held in the route (so, for example, if those stored waypoints happen to be deleted from the database later, the stored route would be unaffected).

How do I generate a PLOG ?

To generate a PLOG, simply send an email to **plogs@flylogical.com**. Everything -- including the specified route -- is controlled via a set of optional parameters specified on the subject-line of the email (the optional parameters and their functionality is described later in the **Parameters** section). The resulting PLOG will be sent by return email a few seconds later (or, in the worst-case, a few minutes later, depending on internet traffic and server load). Specifically, the returning email will contain the following file attachments:

1. **PLOG** as a viewable & printable PDF file
2. **MET** report as a viewable & printable PDF file containing METARs and TAFs along the route
3. **NOTAM** report as a viewable & printable PDF file containing NOTAMS (airfield and FIR) along the route
4. **FREQS** report as a viewable & printable PDF file containing NAVCOM frequencies along the route. *Note: the FREQS data relies on updates from the user-community, and so may not be up-to-date.*
5. **Route GPX** file containing the route encoded in the open-standard GPX XML format. This can be used as the basis for transforming into any other format (e.g., for importing into a GPS device), and can be opened directly in other applications which support the GPX format.
6. **Route FPL** containing the route encoded in the popular FPL XML format used by Garmin. This can be used for importing into a Garmin GPS device.

How can I share my routes with others ?

If you are a registered user, you can save your routes in the FlyLogical database so you can conveniently retrieve them and generate new PLOGs for these routes at any future time. Once saved, a given route can nominally only be viewed by you unless you choose to share it. You can share a given route with any other (registered) user -- or multiple (registered)users -- defined by you. You can also choose to make a given route “public”, whereby everyone can view it. The saving and sharing functionality is controlled via the appropriate parameters as described in the **Parameters** section below. (You can also save and share individual waypoints, but this is controlled via the **Waypoint Manager** web app, and not via the **iNavCalc** email interface directly.)

Route saving and sharing functionality is not available to UNREGISTERED users.

Parameters controlling the PLOG generation

General rules for setting parameters

When specifying parameters on the email subject-line (or, equivalently, in the parameter window of the web app), the following general rules apply:

- Each parameter is specified in the pairwise format **par_name=value** where **par_name** is the name of the parameter, and **value** is the desired value
- For a multi-valued parameter, each value is separated by a comma, for example, **par_name=value1,value2,value3** etc
- Parameter names and values are case-insensitive (makes for ease-of-typing, especially on a mobile device)
- Multiple parameters are separated by the semicolon (**;**) character. For example, three successive multi-valued parameters would be specified as follows **par1_name=par1_value1, par1_value2; par2_name=par2_value1, par2_value2; par3_name=par3_value1, par3_value2** etc
- When specifying multiple parameters, the order of the parameters is irrelevant
- Blank spaces between successive parameter values, and between successive parameters are ignored (so you may insert spaces for clarity, without affecting the meaning)
- Certain parameters are treated as **"sticky"** such that their values are "remembered" from the last time they were specified. This is for convenience so that it is unnecessary to have to keep re-specifying commonly-used parameters for which the values don't tend to change. An example of a sticky parameter is the **culture** parameter which determines the choice of dimensional units for specifying technical quantities and displaying results ('knots' versus 'miles-per-hour' for airspeed, 'litres' versus 'gallons' for fuel, etc). The **culture** parameter is typically specified only once (or changed but only rarely) by a given user (respecting their nominal cultural preferences). Sticky parameter stored values are user-specific. *The sticky parameter functionality is not available to UNREGISTERED users.*

List of available parameters and their meaning

PARAMETER	DESCRIPTION/FUNCTION AND EXAMPLES OF USAGE
route	Multi-valued parameter for defining the route for which the PLOG is to be calculated. Successive values determine the successive waypoints in the route. Each waypoint can be specified in a variety of ways, depending on how the given value is formatted: ● **Name of waypoint**, via which the waypoint is retrieved from the FlyLogical database using a best-match search of all waypoints accessible to given registered user (as identified by sender email address). Accessibility to a given waypoint by a given user is granted if either (i) the user is the owner/creator of the waypoint; (ii) the waypoint has been shared with the given user by the

	owner; or (iii) the waypoint has been made "public" (go to the Waypoint Manager to change the share attributes for a given waypoint). If the database contains multiple waypoints sharing the same name, the one closest to the other waypoints in the route is selected, or failing that, the one closest to the last waypoint accessed, regardless of route. To narrow the search under such conditions use the type descriptor to decorate the name (or description). Note: it is expected that users will contribute waypoints to the database, building it up over time to become a global resource. As such, only a selection of waypoints have been initially pre-loaded by FlyLogical (and made "public"). Specifically, the main airports and navaids in the UK. If any specified wayponts cannot be found in the database, the PLOG generation fails. An email message containing a list of any unresolvable waypoint(s) is sent instead of the PLOG. In such cases, you should access the Waypoint Manager, and create entries for the desired waypoints, then re-submit the PLOG application. ● **Radial bearing and distance from named waypoint**. The reference waypoint is retrieved from the database as above, then the new position is computed relative to this waypoint. The precise format of the entry must be the name of the reference waypoint followed by the bearing in degrees (3 digits), followed by the '**/**' separator, then the range in nautical miles. For example, the value **TRN265/22** would represent the waypoint located 22 nautical miles along the 265 degree radial from the Turnberry (TRN) VOR beacon. The bearing can be either true or magnetic, and the distance along the radial can be computed either as a rhumb-line or a great circle. These specifics are controlled via the **bearingtype** parameter (see next parameter entry in this table). ● **Radial bearing and distance from previous waypoint in route**. Same as above, except the reference waypoint is taken to be the preceding waypoint in the list., indicated by the '**>**' symbol preceding the radial info. For example, the value **>265/22** would represent the waypoint located 22 nautical miles along the 265 degree radial from the preceding waypoint in the route.. As before, the specifics are controlled via the **bearingtype** parameter (see next parameter entry in this table). ● **Waypoint location coordinates** specified as latitude, longitude, name, separated by spaces, contained within curly braces {lat lon name}. The latitude and longitude can be specified in the following common formats (i) degrees decimal-minutes (such as used in many GPS devices and publications); (ii) degrees minutes decimal-seconds (as used by Google Earth); (iii) degrees minutes seconds (as used in various devices and publications such as UK AIP); and (iv) decimal degrees (as used in software packages for performing navigation calculations). Specification of

	the latitude and longitude are mandatory, but the name attribute can be omitted. Note: the specification of all location coordinates in **iNavCalc** assume the WGS-84 reference geoid. If any of the waypoints in the **route** value list cannot be resolved via any of the optional formats detailed above, the PLOG generation fails. An email message containing a list of any unresolvable waypoint(s) is sent instead of the PLOG. In such cases, you should correct the format of the erroneous waypoints, then re-submit the PLOG application. Examples: (i) Simple example route with one leg joining two named waypoints from the FlyLogical database: a flight from EGNS (Ronaldsway, Isle of Man) to EGPK (Prestwick, Scotland): **route=EGNS, EGPK** (ii) Same as (i), but with an intermediate waypoint specified as 51 nm along the 348 (degrees magnetic) radial from the IOM VOR beacon. Note: this location actually corresponds to the published position fix known as BLACA, but this location has has not (yet) been entered as a waypoint in the FlyLogical database (so therefore it needs to be specified manually, for example, as a radial from the IOM beacon, which is in the database): **route=EGNS, IOM348/51, EGPK; bearingtype=magnetic** The resulting PLOG calculates the coordinates of the intermediate waypoint as N54º53.07' W005º09.62' (computed on 2 May 2011 using the built-in magnetic field model). This differs very slightly from the published values (from the UK AIP) which place BLACA at N54º53.0' W005º09.5'. The difference is due to the differences in assumed magnetic variation. The published values use magnetic variation values which are fixed at the date of publication, whereas **iNavCalc** calculates the magnetic variation as of the current calendar date. (iii) Same as (ii), specifying the intermediate waypoint explicitly in terms of it's coordinate location in the commonly-used degrees-decimal-minutes format (note: we have also specified the name BLACA for the waypoint since this additional attribute is supported when entering coordinates): **route=EGNS, {N5453.07 W00509.62 BLACA}, EGPK** (iv) Same as (iii), but using the degrees-minutes-seconds format for the coordinates of the intermediate waypoint (note: under this format the N, S, E, W symbols appear after the coordinate

	numerical digits, whereas in the previous format, they appear before the numerical digits): **route=EGNS, {545304N 0050937W BLACA}, EGPK** (v) Same as (iv), but using the more precise degrees-minutes-decimal-seconds format for the coordinates of the intermediate waypoint, as displayed by Google Earth (note: again, under this format the N, S, E, W symbols appear after the coordinate numerical digits): **route=EGNS, {545304.20N 0050937.20W BLACA}, EGPK** (vi) Same as (v), but using the decimal-degrees format for the coordinates of the intermediate waypoint (note, by convention: N and E are positive, S and W are negative): **route=EGNS, {54.8844879 -5.1602881 BLACA}, EGPK** (vii) Same as (ii), adding another intermediate waypoint immediately after departure from EGNS, demonstrating the use of the "radial from previous waypoint" option. Specifically, after departure, fly initially on a a (magnetic) track of 350 degrees (relative to previous waypoint, in this case, the origin waypoint EGNS) for 20 nm, before resuming on to IOM348/51 etc: **route=EGNS, >350/20, IOM348/51, EGPK; bearingtype=magnetic** (viii) Further example demonstrating the use of the "radial from previous waypoint" option. Specifically, after departure from EGNS, fly an equilateral triangle, built-up from successive 20 nm segments defined relative to preceding waypoints, assuming true-track rhumb-lines. **route=EGNS, >075/20, >195/20, >315/20; bearingtype=true** By comparing the coordinates of the end point with the starting point (from the generated PLOG), it can be seen that they are almost coincident (N54º05.00' W004º37.43' versus N54º05.00' W004º36.70'). This corresponds to a discrepancy of less than a nautical mile (over the 60 nm total distance), and is due to the fact that the "flat earth" geometry is not truly representative of the curved geometry. (ix) An example demonstrating how to use waypoint type descriptors when searching for waypoint names (or descriptions) which are ambiguous. **route=goodwood airfield, egtk, goodwood vor, egtc**

	This will correctly resolve **goodwood airfield** into **EGHR,** and **goodwood vor** into **GWC** (specifying **goodwood** without a type descriptor would always resolve to **EGHR** which is fine if the airfield is the intended waypoint but not if the VOR beacon is the intended waypoint). Note: the **iNavCalc Route Manager and iNavCalc Waypoint Manager Web Apps** enable you to create, edit and manage your routes and waypoints using the same options as available via the **route** parameter, but using web-forms rather than parameter list interface (*not available to UNREGISTERED users*). **Shorthand for quick specification of route** For the quick specification of a route, you can omit the **route** keyword (and all other keywords) and simply list the waypoints, separated by spaces (*not* commas), as follows: **EGNS EGPK** *or* **EGNS {54.8844879 -5.1602881 BLACA} EGPK** *or* **EGNS >075/20 >195/20 >315/20** *etc*
bearingtype [sticky]	Used in connection with the **route** parameter (see above) to specify whether bearings should be treated as true or magnetic, and whether distances along radials should be computed as rhumb-lines or great circles. Possible values are: **magnetic** **true** **magnetic great circle** **true great circle** The default value (if **bearingtype** parameter never specified) is **magnetic.** If **great circle** is omitted from the value, rhumb-lines are assumed, in which case, the end coordinates correspond to the distance along a straight-line drawn on a Mercator projection chart. When building a route using published radials from navigation beacons,

	use the default values (**bearingtype=magnetic** , (rhumb-line (implied)). Alternatively, when building a route from straight lines drawn on a Mercator-projection chart (with true grid-lines), use **bearingtype=true**, (rhumb-line (implied)). It is unlikely you would ever need to use the **great circle** qualifier (since radials are almost never specified as great-circle starting angles), but this option has been included for completeness. Note: for short distances (on the order of 100 nm, typical of GA route leg lengths), the differences between the two approaches is generally negligible from a flight-planning point-of view. In all cases, the PLOG is computed assuming great-circles are flown between successive waypoints, irrespective in the manner in which the waypoint coordinates have been calculated along the radial (i.e, great circle or rhumb-line). The track and heading angles on the PLOG therefore represent the initial angles at the start of each great-circle arc between the waypoints, and the leg lengths on the PLOG correspond to the great-circle distances between the coordinates (not the rhumb-line distances). Examples: (i) Fly out along the 060 (true) radial from the IOM (VOR beacon) for 100nm, then turnaround an fly back along the same track in the opposite direction. Since **bearingtype=true**, and since **great circle** not specified, the default rhumb-line computation will apply. Look at the generated PLOG: you will see that the start and end point coordinates are identical, as would be expected when computing the coordinates from rhumb-lines (i.e., straight-lines) on a Mercator projection. However, you will observe that the true tracks to be flown are 059 (and 241 degrees returning) rather than the specified 060 (and 240). That's because the PLOG assumes all legs are flown as great-circle arcs, whereby the track/headings correspond to the initial angles of the arc. **route=IOM, IOM060/100, >240/100; bearingtype=true** (i) Same but specify **great circle** rather than the default rhumb-line. Look at the generated PLOG: you will see that the start and end point coordinates are no longer identical, since the true tracks 060 (and 240) now correspond to starting angles on great-circle arcs, rather than as constant headings (rhumb-lines) which are straight lines on a Mercator projection. You will observe that the true tracks to be flown are 060 (and 240 degrees returning) are exactly as specified. That's because the PLOG assumes all legs are flown as great-circle arcs, whereby the track/headings correspond to the initial angles of the arc, as specified by the **great circle** qualification in **bearingtype.** **route=IOM, IOM060/100, >240/100; bearingtype=true great circle**

	(iii) Same as (i) except using **magnetic** rather than **true** bearings. For the bearing calculations, the magnetic variation is computed at each waypoint, subtracted from the respective specified bearing, then applied using the rhumb-line assumption of constant true track on the Mercator projection. As such, the start and end point coordinates are no longer identical (as they were under **true)** because the magnetic variation (computed automatically) at the start of the first leg is different from at the start of the return leg (by 0.8 degrees). Hence, the outbound and inbound legs are slight misaligned (by 0.8 degrees) **route=IOM, IOM060/100, >240/100; bearingtype=magnetic** (iv) For completeness, same as (ii) except using **magnetic** rather than **true** bearings. In this case, the start and end point coordinates are different due to the combined effects of the great-circle arc starting angles (versus rhumb-lines) discussed in example (ii); and the magnetic variation differences discussed in example (iii). **route=IOM, IOM060/100, >240/100; bearingtype=magnetic great circle** (v) For illustrative purposes, to demonstrate the difference between rhumb-line distance and great-circle distance, consider a 2000 nm radial-range computed using the default rhumb-line assumption. In the computed PLOG, you will see that the corresponding leg length (great-circle distance) is only 1917 nm (i.e., 83 nm shorter than the rhumb-line distance). Moreover, the initial true track (great circle arc starting angle) is 034 instead of the specified 060. **route=IOM, IOM060/2000; bearingtype=true** Note: the **iNavCalc Route Manager Web App** enables you to create, edit and manage your routes using the same options as available via the **route** parameter, but using a web-form rather than parameter list interface (*not available to UNREGISTERED users*).
reverse	Reverse the route when computing the PLOG. Possible values: **true**, in which case, the PLOG is computed for the reverse of the route specified. Also, when used in combination with the **save** parameter, it is the reversed route that is saved. Example: **route=EGNS, EGPK; reverse=true**

	The PLOG will be computed for the reverse of the route specified, i.e., EGPK to EGNS.
save	Save the route into the FlyLogical database (*this functionality not available for UNREGISTERED users*). Possible values: **true** if the **routeid** parameter (see next parameter entry in this table) has not also been specified, the route is saved as a new route into the FlyLogical database under the ownership of the registered user (as determined by the email address of the sender). If the save is successful, in addition to the usual PLOG email, you will receive a confirmation email containing the **routeid** of the newly-created route. If a valid **routeid** has been specified, then the **name** and/or **description** optional fields are updated for the existing route in the database (see **name** and **description** parameter descriptions later in this table); **copy** only available if the **routeid** parameter has been specified (see next parameter entry in this table). In which case, the route given by the **routeid** is copied into a duplicate route in the FlyLogical database under the ownership of the registered user (as determined by the email address of the sender). If the copy is successful, in addition to the usual PLOG email, you will receive a confirmation email containing the **routeid** (see next parameter entry in this table) of the newly-created route. Examples: (i) Simple example of creating and saving a new route **route=EGNS, EGPK; save=true** Two emails will be returned: the first containing the PLOG as usual, the second containing the **routeid** for the newly-created route. (ii) Same as (i), but this time specifying the optional **name** and **description** parameters (see **name** and **description** parameter descriptions later in this table) **route=EGNS, EGPK; save=true; name=My Saved Route; description=About My Saved Route** (iii) Changing the **name** of a route which already exists in the database. **routeid=9ca17d64-1f48-44ad-ba11-3de4016d9ee7; save=true; name=My New Name for Route; description=About My Saved Route**

	(iii) Changing the **name** and **description** of a route which already exists in the database. **routeid=9ca17d64-1f48-44ad-ba11-3de4016d9ee7; save=true; name=My New Name for Route; description=My New Description** Note: the **iNavCalc Route Manager Web App** enables you to create, edit and manage your routes using the same options as available via the **route** parameter, but using a web-form rather than parameter list interface (*not available to UNREGISTERED users*).
routeid [sticky]	A unique identifier (36-character string) automatically assigned to each route saved in the FlyLogical database. By specifiying the **routeid** for a given stored route, the PLOG can be computed on a pre-existing route without having to specify the waypoints via the **route** parameter. The **routeid** is sticky, so that you do not need to keep re-specifying it when working with the same route. Also, when working with the Route manager web app interface (rather than the email interface), there is no need to specify the **routeid** in the parameter list since it is automatically inserted for the given route under consideration. Only those routes accessible to a given user can be accessed. Accessibility to a given route by a given user is granted if either (i) the user is the owner/creator of the route; (ii) the route has been shared with the given user by the owner (see the **share** parameter later); or (iii) the route has been made "public" (see the **public** parameter later) Examples: **routeid=368df0fc-8d43-410b-bfe4-cdb6d039a3f4** Generates the PLOG for a pre-existing route stored in the FlyLogical database (in this example, the demo "EGNH to EGSF via POL" route which is shared pubicly).This is sticky, so that you may then issue **time=12:00** which will re-compute the PLOG for a departure time of 12:00, without having to re-specify the **routeid**. Likewise, the homeward journey can be computed using: **time=16:30; reverse=true** again, without having to re-specify **routeid** In addition to generating PLOGs by email, stored routes owned by a

	given registered user can be accessed via the Route manager web app. You can open the Route manager web app directly on a given route via the following URL (replacing **routeid=368df0fc-8d43-410b-bfe4-cdb6d039a3f4** as appropriate for desired **routeid**): http://flylogical.com/FlyLogicalWeb/Fly/eba21f6b-6e68-4f25-9c64-6a789440f5b7/tt_Hggt52/Routes.aspx?**routeid=368df0fc-8d43-410b-bfe4-cdb6d039a3f4** For convenience, the appropriate URL for a given **routeid** is included in each PLOG email response.
name	Optional parameter used in conjunction with the **save** parameter (see earlier). Text field for specifying the name of the route when saving in the database. It is recommended that a name be specified, otherwise the route will appear as a blank entry in the list of routes, making it awkward to identify via the Route manager web app.
description	Optional parameter used in conjunction with the **save** parameter (see earlier). Text field for specifying a description of the route when saving in the database. Can generally be omitted.
delete	Used in conjunction with the **routeid** parameter for deleting a route from the FlyLogical database (*this functionality not available for UNREGISTERED users*). You can only delete a route for which you are the owner. Possible values: **true**, in which case, the route given by the **routeid** is deleted. Example: **routeid=9ca17d64-1f48-44ad-ba11-3de4016d9ee7; delete=true**
share	Used in conjunction with the **routeid** parameter for sharing a route with other (registered) users (*this functionality not available for UNREGISTERED users*). Multi-valued parameter, each value being the email address of a registered user with whom you wish to share the route. You must be the owner of a route to share it with others. Example: **routeid=9ca17d64-1f48-44ad-ba11-3de4016d9ee7; share=joe.blogs@mysite.com, brenda@soundsgood.org** Shares the route (with specified **routeid**, which you must be owner of) with registered users whose email addresses are **joe.blogs@mysite.com** and **brenda@soundsgood.org.** Note: these email addresses must already be registered with FlyLogical. Each user on the list will be sent an email notifying them that the route has been shared with them.

	Note: the **iNavCalc Route Manager Web App** enables you to manage your route shares using a web-form rather than parameter list interface (*not available to UNREGISTERED users*).
unshare	Used in conjunction with the **routeid** parameter for un-sharing a route with other (registered) users (or for you to opt-out from somone else's share) (*this functionality not available for UNREGISTERED users*). Possible values: Multi-valued parameter, each value being the email address of a registered user from whom you wish to remove access to the route. You must be the owner of a route to remove share from others; **all** remove access to the route from all users. You must be the owner of a route to remove share from others; **true** opt-out from sharing a route that has been shared with you by another user. Examples: (i) Remove the shares created earlier (under the **share** example) **routeid=9ca17d64-1f48-44ad-ba11-3de4016d9ee7; unshare=joe.blogs@mysite.com, brenda@soundsgood.org** Removes the route (with specified **routeid**, which you must be owner of) from being shared with (registered) users whose email addresses are **joe.blogs@mysite.com** and **brenda@soundsgood.org.** (ii) Remove all shares for given **routeid** (which you must be owner of) **routeid=9ca17d64-1f48-44ad-ba11-3de4016d9ee7; unshare=all** (iii) Opt-out from sharing a route which has been shared with you by another owner **routeid=9ca17d64-1f48-44ad-ba11-3de4016d9ee7; unshare=true** Note: the **iNavCalc Route Manager Web App** enables you to manage your route shares using a web-form rather than parameter list interface (*not available to UNREGISTERED users*).
public	Used in conjunction with the **routeid** parameter. Similar to **share**, except

	public enables sharing the given route with all other (registered) users at once, rather than a select list (*this functionality not available for UNREGISTERED users*). Possible parameter values: **true**, in which case the route is made accessible by all users. You must be the owner of a route to make it public. Example: **routeid=9ca17d64-1f48-44ad-ba11-3de4016d9ee7; public=true** Shares the route (with specified **routeid**, which you must be owner of) with all other registered users (i.e., public). Note: the **iNavCalc Route Manager Web App** enables you to manage your route shares using a web-form rather than parameter list interface (*not available to UNREGISTERED users*).
unpublic	Undoes the effect of **public.** Used in conjunction with the **routeid** parameter, removes the public accessibility for specified route. Possible parameter values: **true**, in which case the public access is removed for the route. You must be the owner of a route to remove it's public access status. Example: **routeid=9ca17d64-1f48-44ad-ba11-3de4016d9ee7; unpublic=true** Removes public access for the route (with specified **routeid**, which you must be owner of). Note: the **iNavCalc Route Manager Web App** enables you to manage your route shares using a web-form rather than parameter list interface (*not available to UNREGISTERED users*).
noplog	Possible values: **true**, in which case, the generation of a PLOG (and associated email) is suppressed. This parameter is typically used when you are working with the FlyLogical database (e.g., saving, copying, re-naming, sharing, deleting route(s), etc) where there is no immediate need for a PLOG to be generated. Example: **routeid=9ca17d64-1f48-44ad-ba11-3de4016d9ee7; delete=true; noplog=true** In this case, the primary intent of the command is to delete an unwanted route. As such there is no need to generate a PLOG.

culture [sticky]	For selecting the preferred dimensional units to be used in the PLOG calculations. For example, 'knots' vs 'miles-per-hour' for airspeed, 'litres' versus 'gallons' for fuel. Typical preferences used by pilots from different geographical regions (e.g., UK, US) are grouped into 'culture sets' which are selected via the **culture** parameter. Possible values are: **UK** (typical settings used in UK/Europe, fuel in litres) **UK GAL** (same as above, but fuel in UK gallons) **US** (typical settings used in US) The default culture (if **culture** parameter unspecified) is **UK**. To see what units are actually included within each culture, see the table titled **Dimensional units per culture** below. If you would like a different variation, let us know (via Twitter.com/FlyLogical). Examples: (i) Compute PLOG using typical UK/European settings (note: "mb" used for pressure which is numerically identical to "hPa" as more commonly specified in Europe). Fuel specified in litres. **route=EGNS, EGPK; culture=UK** (ii) Compute PLOG using typical UK settings but with fuel specified in UK gallons (rather than the default litres). Useful for older aircraft types such as the Scottish Aviation Bulldog (of which we are devotees at FlyLogical) where the fuel gauges display UK gallons (not litres). **route=EGNS, EGPK; culture=UK GAL** (iii) Compute PLOG using typical US settings. Temperature specified in 'F' rather than 'C', windspeed in 'mph' rather than 'kts', pressure in 'inchesHg' rather than 'mb', fuel specified in US gallons, etc. **route=EGNS, EGPK; culture=US**
alt	Multi-valued parameter for specifying the cruise altitude, one per way-point, units as per **culture**. Applies last specified value to all following waypoints if number of values specified is less than the number of waypoints. It is not necessary to specify zero altitude for the first waypoint since this will be automatically accounted for in the fuel calculations. So, the value for the first waypoint should be set to the altitude of the first leg, i.e., after the initial climb-out. The fuel consumption calculations account for the initial climb-out during which the fuel flow rate is typically higher. Likewise, if the altitude values increase from waypoint-to-waypoint, it is assumed that the fuel flow rate for the climb is used for the climb portion within the leg, and then the cruise settings thereafter for the remaining portion of the leg.To ensure

	that the fuel computations are consistent, the altitude profile for the route should respect the climb rate for the given aircraft. The specified altitude increase between successive waypoints should not exceed those achievable for given aircraft climb performance over given leg distance. Similar considerations apply for descent segments. Likewise, the specified altitude for the final waypoint should not be set to zero, but rather to the cruise altitude for the last leg. To err on the side of caution, the final descent-to-land is not computed at a lower fuel flow rate. By default, winds are computed automatically for each waypoint using forecast winds at given **alt** values along the route (including accounting for possible wind variations between ground level and cruise altitude at start and end of route). The average wind along a given leg is used to compute the wind correction to be steered for given leg (as reporting in the PLOG 'Hdg' column). This "AutoMETic" behaviour can be deactivated by setting **met=manual**, in which case the wind values per leg should be entered manually (see the **met** and **wind** parameters). Default value of 2500 ft (in equivalent units as per **culture**) is assumed if **alt** unspecified. When used in combination with the **save** parameter, the **alt** values are saved with the route so that when the route is re-used via it's **routeid** parameter, there is no need to re-specify the **alt** profile, unless you wish to change it. Examples: (i) Same example route from earlier. Specify a single altitude (3000 ft) to be used for entire route. The fuel calculations will include the initial climb-out to 3000 ft, plus cruise thereafter. The wind drift will be computed using the forecast winds at 3000 ft. **route=EGNS, >350/20, IOM348/51, EGPK; alt=3000** (ii) Same route as (i), but specify a varying altitude profile along the route.The wind drift will be computed using the forecast winds at each specified altitude, averaged across the end-points for each leg. **route=EGNS, >350/20, IOM348/51, EGPK; alt=3000, 5000, 4500, 2000** (iii) Same route as (ii), but specify a varying altitude profile with fewer values specified than number of waypoints in route. In this case, it is assumed that there will be an en-route climb from 3000 ft to 5000 ft (and consequent increased fuel consumption) during the second leg, but that the rest of the trip will be flown at 5000 ft (since the number of specified **alt** values is less than the number

	of waypoints). **route=EGNS, >350/20, IOM348/51, EGPK; alt=3000,3000,5000**
ias [sticky]	Multi-valued parameter for specifying the cruise IAS (indicated airspeed), one value per way-point, units as per **culture**. Applies last specified value to all following waypoints if number of values specified is less than the number of waypoints. If the IAS varies between waypoints, the average across a leg is used in the nav calculations. Parameter is sticky (the first value in a multi-value list is the one remembered) since IAS usually applies to a given aircraft so you typically only need to change it if you are switching to a different aircraft for a given PLOG calculation. Default value of 100 kts (in equivalent units as per **culture**) if **ias** has never been specified by given user, otherwise uses sticky value from last usage by given user. When used in combination with the **save** parameter, the **ias** values are saved with the route so that when the route is re-used via it's **routeid** parameter, there is no need to re-specify the **ias** profile, unless you wish to change it. Examples: (i) Same example route from earlier. Specify a single **ias** (110 kts) to be used for entire route. **route=EGNS, >350/20, IOM348/51, EGPK; ias=110** (ii) Same but without specification of the **ias** parameter. The sticky value of 110 kts will be assumed. **route=EGNS, >350/20, IOM348/51, EGPK** (ii) Same but with a multiple **ias** values. The first in the list (110 kts) will be retained as the sticky value for next use. The average value across the first two waypoints, 105 kts, will be used for the first leg, and the value of 100 kts will be used for all legs thereafter (since the number of specified **ias** values is less than the number of waypoints). **route=EGNS, >350/20, IOM348/51, EGPK; ias=110,100**
iasclimb [sticky]	For specifying the climb IAS (indicated airspeed) in order to facilitate correct treatment of wind correction and en-route time computations during climb (both for the initial climb-out, and any altitude increase along the route).

	Parameter is sticky since climb IAS usually applies to a given aircraft so you typically only need to change it if you are switching to a different aircraft for a given PLOG calculation. Default value of 80 kts (in equivalent units as per **culture**) if **iasclimb** has never been specified by given user, otherwise uses sticky value from last usage by given user. For examples of usage, see examples for the **fuelfloweconomy** parameter later in this table.
iaseconomy [sticky]	For specifying the IAS (indicated airspeed) for best-economy cruise in order to facilitate calculation of remaining endurance for diversion. Parameter is sticky since best-economy IAS usually applies to a given aircraft so you typically only need to change it if you are switching to a different aircraft for a given PLOG calculation. Default value of 85 kts (in equivalent units as per **culture**) if **iaseconomy** has never been specified by given user, otherwise uses sticky value from last usage by given user. For examples of usage, see examples for the **fuelfloweconomy** parameter later in this table.
iasdescent [sticky]	For specifying the IAS (indicated airspeed) during descent in order to facilitate correct treatment of wind correction and en-route time computations during descent (both for the any en-route altitude decrease, as well as the final descent-to-land). Parameter is sticky since decsent IAS usually applies to a given aircraft so you typically only need to change it if you are switching to a different aircraft for a given PLOG calculation. Default value of 100 kts (in equivalent units as per **culture**) if **iasdescent** has never been specified by given user, otherwise uses sticky value from last usage by given user. For examples of usage, see examples for the **fuelfloweconomy** parameter later in this table.
climbrate [sticky]	For specifying the vertical climb rate in order to facilitate correct treatment of time computations during climb (both for the initial climb-out, and any altitude increase along the route). Parameter is sticky since climb-rate usually applies to a given aircraft so you typically only need to change it if you are switching to a different aircraft for a given PLOG calculation. Default value of 1000 ft-per-minute (in equivalent units as per **culture**) if **climbrate** has never been specified by given user, otherwise uses sticky value from last usage by given user. For examples of usage, see examples for the **fuelfloweconomy** parameter later in this table.
descentrate [sticky]	For specifying the vertical descent rate in order to facilitate correct treatment of time computations during descent (both for the any en-route

	altitude decrease, as well as the final descent-to-land). Parameter is sticky since descent-rate usually applies to a given aircraft so you typically only need to change it if you are switching to a different aircraft for a given PLOG calculation. Default value of 500 ft-per-minute (in equivalent units as per **culture**) if **descentrate** has never been specified by given user, otherwise uses sticky value from last usage by given user. For examples of usage, see examples for the **fuelfloweconomy** parameter later in this table.
fuelflow [sticky]	Multi-valued parameter for specifying the cruise fuel flow-rate, one value per way-point, units as per **culture**. Applies last specified value to all following waypoints if number of values specified is less than the number of waypoints. If the fuel flow-rate varies between waypoints, the average across a leg is used in the fuel consumption calculations. Note: the fuel used in climbing and descending is automatically calculated (see **fuelflowclimb** and **fuelflowdescent** parameters) so there is no need to account for climb and descent in the cruise fuel flow-rate specifications. Simply specify the values corresponding to the cruise portion of a given leg. Parameter is sticky (the first value in a multi-value list is the one remembered) since fuel flow-rate usually applies to a given aircraft so you typically only need to change it if you are switching to a different aircraft for a given PLOG calculation. Default value of 36 litres per hour (in equivalent units as per **culture**) if **fuelflow** has never been specified by given user, otherwise uses sticky value from last usage by given user. Examples: (i) Same example route from earlier. Specify a single **fuelflow** (40 litres per hour, units as per UK **culture**) to be used for entire route. **route=EGNS, >350/20, IOM348/51, EGPK; ias=110; fuelflow=40; culture=UK** **It is very important to ensure that you have set the correct culture when specifying fuelflow to make sure the units are consistent. Hence, in this example we enforce culture=UK since we want to ensure that the fuelflow is interpreted in litres-per-hour as intended by the numerical value.** (ii) Same but without specification of the **fuelflow** parameter (or **culture** parameter). The sticky value of 40 litres-per-hour will be assumed (in the context of the sticky **culture** value of **UK** for the correct units)

	route=EGNS, >350/20, IOM348/51, EGPK (ii) Same but with a multiple **fuelflow** values. The first in the list (40 litres per hour) will be retained as the sticky value for next use. The average value across the first two waypoints, 38 litres per hour, will be used for the first leg, and the value of 36 litres per hour will be used for all legs thereafter (since the number of specified **fuelflow** values is less than the number of waypoints). **route=EGNS, >350/20, IOM348/51, EGPK; ias=110, 100; fuelflow=40, 36**
startfuel [sticky]	Specifies fuel at start of flight, units as per **culture**. Normally you would set as the usable fuel capacity of given aircraft, assuming departing on full tanks as the norm, and leave as sticky value accordingly. However, if you are planning a trip where you are not departing on full tanks, you would re-specify with the planned take-off fuel etc. Default value 32 UK gallons (in equivalent units as per **culture**) if **startfuel** has never been specified by given user, otherwise uses sticky value from last usage by given user. Examples: (i) Same example route from earlier. Specify **startfuel** as 90 litres (units as per UK **culture**). **route=EGNS, >350/20, IOM348/51, EGPK; startfuel=90; culture=UK** **It is very important to ensure that you have set the correct culture when specifying startfuel to make sure the units are consistent. Hence, in this example we enforce culture=UK since we want to ensure that the startfuel is interpreted in litres as intended by the numerical value.** (ii) Same but without specification of the **startfuel** parameter (or **culture** parameter). The sticky value of 90 litres will be assumed (in the context of the sticky **culture** value of **UK** for the correct units) **route=EGNS, >350/20, IOM348/51, EGPK**
fuelflowclimb [sticky]	Specifies fuel flow rate during climb, units as per **culture** for use in fuel calculations. Parameter is sticky since fuel flow rate in climb usually applies to a given aircraft so you typically only need to change it if you are switching to a different aircraft for a given PLOG calculation.

	Default value 60 litres per hour (in equivalent units as per **culture**) if **fuelflowclimb** has never been specified by given user, otherwise uses sticky value from last usage by given user. For examples of usage, see examples for the **fuelfloweconomy** parameter later in this table.
fuelflowdescent [sticky]	Specifies fuel flow rate during descent, units as per **culture** for use in fuel calculations. Parameter is sticky since fuel flow rate during descent usually applies to a given aircraft so you typically only need to change it if you are switching to a different aircraft for a given PLOG calculation. Default value 30 litres per hour (in equivalent units as per **culture**) if **fuelflowdescent** has never been specified by given user, otherwise uses sticky value from last usage by given user. For examples of usage, see examples for the **fuelfloweconomy** parameter later in this table.
fuelfloweconomy [sticky]	Specifies fuel flow rate during at best-economy cruise, units as per **culture,** for use in endurance calculations for diversion planning. Parameter is sticky since fuel flow rate during best-economy cruise usually applies to a given aircraft so you typically only need to change it if you are switching to a different aircraft for a given PLOG calculation. Default value 30 litres per hour (in equivalent units as per **culture**) if **fuelfloweconomy** has never been specified by given user, otherwise uses sticky value from last usage by given user. Examples (for this and related/similar parameters): (i) Same example route from earlier. Specify values for various performance parameters including **fuelfloweconomy** (units as per UK **culture**). **route=EGNS, >350/20, IOM348/51, EGPK; startfuel=90; culture=UK; iasclimb=75; iaseconomy=90; iasdescent=95; fuelflowclimb=65; fuelfloweconomy=35; fuelflowdescent=35; climbrate=1100; descentrate=600;** **It is very important to ensure that you have set the correct culture when specifying these parameters to make sure the units are consistent. Hence, in this example we enforce culture=UK since we want to ensure that the parameters are interpreted in the units as intended by the numerical values.** (ii) Same but without specification of the various performance parameters (or **culture** parameter). The sticky values from previous example will be assumed (in the context of the sticky **culture** value of **UK** for the correct units)

	route=EGNS, >350/20, IOM348/51, EGPK
date	Date of flight (spelled out as a string), defaults to today's date if unspecified. The date value primarily affects the solar incidence angle (and day/night) calculations (since solar incidence varies with calendar date for given location) and the magnetic variation calculations (since the Earth's magnetic field changes over time). Examples (i) Do not specify the **date** parameter. Today's date will automatically be assumed. **route=EGNS, >350/20, IOM348/51, EGPK;** (ii) Specify the **date** (in this example 1 April 2011) **route=EGNS, >350/20, IOM348/51, EGPK;date = 1 April 2011**
time	Local time of flight (spelled out as a string), defaults to current time if unspecified. The **time** value determines the departure time of the flight (in local time, see also the **zone** parameter) and consequently the ETA (estimated time of arrival) at each waypoint, as reported on the PLOG. It also directly affects the solar incidence angle (and day/night) calculations which are computed for the ETA at each waypoint. Examples (i) Do not specify the **time** parameter. Current time will automatically be assumed. **route=EGNS, >350/20, IOM348/51, EGPK;** (ii) Specify the **time** (in this example 14:30 local) **route=EGNS, >350/20, IOM348/51, EGPK;time = 14:30**
zone [sticky]	Defines the time zone for which "my watch" is set relative to UTC and thereby also determines the time of departure in UTC for a given **time** parameter (which is in local time). The **zone** parameter is specified as a numerical value corresponding to the offset in numbers of hours (can be fractional) between local time and UTC. It is positive when local time is ahead of UTC, and negative when local time is behind UTC. The PLOG displays time of departure from origin and and time of arrival at destination in both UTC and local ("my watch"). The detailed route timeline (per waypoint) is displayed in UTC. The parameter is sticky since it is typically only changed by a given user whenever the clocks go backwards or forwards, or when the user

	happens to switch to a different time zone (and re-sets "my watch" accordingly). It can also be set to 0 if the user prefers to always set "my watch" to UTC. The default value is 0 if **zone** has never been specified by given user, otherwise uses sticky value from last usage by given user. Examples (i) Specify the **zone** parameter to BST (British Summer Time) which is one hour ahead of UTC, and the **time** parameter to 14:30 local. This implies that the PLOG will have a departure time of 13:30 UTC (14:30 on "my watch") **route=EGNS, >350/20, IOM348/51, EGPK;zone=1;time=14:30** (ii) Same as (i), but don't specify the **zone** parameter again. The PLOG will have a departure time of 13:30 UTC (14:30 on "my watch") assuming the previous sticky value of +1 for the **zone** value. **route=EGNS, >350/20, IOM348/51, EGPK;time=14:30** (ii) Specify the **zone** parameter to correspond to EDT (Eastern Daylight Time) which is four hours behind UTC, and the **time** parameter to 14:30 local. This implies that the PLOG will have a departure time of 10:30 UTC (14:30 on "my watch") **zone=-4;time=14:30**
stopwatch [sticky]	Defines the waypoint along the route for which the stopwatch is reset to zero. Specified as an integer value corresponding to the index of the waypoint at which the stopwatch is reset. The PLOG displays accumulated time along the route in terms of the stopwatch (alongside UTC), enabling, for example, a navex (navigation exercise) to be timed starting from a specified waypoint different from the point of departure. The parameter is sticky. The default value is 1 meaning that the stopwatch starts at the first waypoint (i.e., the point of departure) if **stopwatch** has never been specified by given user, otherwise uses sticky value from last usage by given user. Example Reset the stopwatch to zero at the second waypoint in the route. The PLOG will reflect this in it's timeline display. **route=EGNS, >350/20, IOM348/51, EGPK;stopwatch=2**
transalt	Multi-valued parameter for specifying the transition altitude, one per way-point, units as per **culture**. Applies last specified value to all following waypoints if number of values specified is less than the number of

	waypoints. The transition altitude is used to calculate the minimum usable flight level (in the altimetry section of the PLOG) at the location of each waypoint for given atmospheric conditions. By default, atmospheric conditions are computed automatically for each waypoint using actual surface temperates and pressures along the route This "AutoMETic" behaviour can be deactivated by setting **met=manual**, in which case the atmospheric conditions should be entered manually (see the **met**, **oat**, **qnh**, **elev**, **lapserate** parameters). Default value of 3000 ft (in equivalent units as per **culture**) is assumed if **transalt** unspecified. When used in combination with the **save** parameter, the **transalt** values are saved with the route so that when the route is re-used via it's **routeid** parameter, there is no need to re-specify the **transalt** profile, unless you wish to change it. Examples: (i) Same example route from earlier. Specify a single transition altitude (6000 ft) to be used for entire route. **route=EGNS, >350/20, IOM348/51, EGPK; transalt=6000** (ii) Same route as (i), but specify a varying transition altitude profile along the route (e.g., switching from 3000 ft to 6000 ft for Scotland) **route=EGNS, >350/20, IOM348/51, EGPK; alt=3000, 3000, 6000, 6000**
metrange [sticky]	Specifies range (in nautical miles) around each waypoint for which weather stations should be included in the separate MET report accompanying the PLOG. Note: this MET report is produced entirely separately from the PLOG, and can can be broadened in geographical coverage (via the **metrange** parameter) to include stations further afield than the route itself (e.g., to provide a broader view of the weather in the surrounding region). By contrast, within the PLOG itself, the MET stations nearest to each waypoint are used, irrespective of the **metrange** value. The parameter is sticky. The default value is 25 nm if **metrange** has never been specified by given user, otherwise uses sticky value from last usage by given user. Example

	Generate a weather briefing (accompanying the PLOG) covering a region which includes all weather stations within 50 nautical miles of each waypoint on specified route. **route=EGNS, >350/20, IOM348/51, EGPK; metrange=50**
met	Determines whether the weather calculations used throughout the PLOG should be derived automatically from global weather reports & forecasts (we call this "AutoMETic"), or whether the weather data should be specified manually. Possible values: **auto** weather data derived automatically. For each waypoint, a search is performed to find the nearest MET station, for which the current weather data is determined for use in the calculations. Note: the **currently available** weather data is used i.e., at the point in time when the PLOG requested is submitted, irrespective of the **date** and **time** parameter settings which are ignored in the weather computations. In other words, all weather computations correspond to the global weather reports currently available when the request is made**,** so when using AutoMETic, the PLOG should be computed at least within the day of the flight for the weather computations to be generally relevant, but better still, within an hour or so of the flight since the METAR's are typically updated hourly (or half-hourly). **manual** weather data to be specified manually (see **wind**, **oat**, **qnh**, **elev**, **lapserate** parameters)
wind	Multi-valued parameter specifying the wind to be used in the PLOG calculations if **met** set to **manual** such that "AutoMETic" has been deactivated. Applies last specified value to all following waypoints if number of values specified is less than the number of waypoints. Each value is specified as **direction/speed** pair (separated by ***/***) where **direction** is the wind direction (degrees true) (at the altitude of flight) and **speed** is the corresponding wind-speed in units as per **culture** Default value of zero wind is assumed if **wind** is unspecified. When used in combination with the **save** parameter, the **wind** values are saved with the route so that when the route is re-used via it's **routeid** parameter, and if "AutoMETic" is disabled (**met** set to **manual)**, you can compute the PLOG using the previously-saved **wind** profile. Examples (i) Same example route from earlier. Specify a single **wind** (280 degrees, 10 kts, units as per UK **culture**) to be used for entire route.

	route=EGNS, >350/20, IOM348/51, EGPK; alt=2000;wind=280/10; culture=UK **It is very important to ensure that you have set the correct culture when specifying wind to make sure the units are consistent. Hence, in this example we enforce culture=UK since we want to ensure that the wind-speed is interpreted in kts as intended by the numerical value.** (ii) Same but with a wind profile varying along the route (e.g., typically as a a function of location and altitude as determined manually from forecasts etc) (in the context of the previously-assigned sticky **culture** value of **UK** for the correct wind-speed units) **route=EGNS, >350/20, IOM348/51, EGPK; alt=3000, 5000, 4500, 2000; wind=280/35, 300/40, 295/30, 280/10** (iii) Same but with fewer specified wind values than number of waypoints, in which case the last specified value (300/40) will be assumed for all remaining waypoints **route=EGNS, >350/20, IOM348/51, EGPK; alt=3000, 5000, 4500, 2000; wind=280/35, 300/40**
oat	Multi-valued parameter specifying the temperature at the surface for use in the atmospheric calculations (within the PLOG) if **met** set to **manual** such that "AutoMETic" has been deactivated. Applies last specified value to all following waypoints if number of values specified is less than the number of waypoints. Each value is specified as outside-air-temperature in units as per **culture,** pertaining to the ground surface location (with elevation given by **elev**) coincident with the waypoint coordinates. Default value of 15 C (in equivalent units as per **culture**) is assumed if **oat** is unspecified. When used in combination with the **save** parameter, the **oat** values are saved with the route so that when the route is re-used via it's **routeid** parameter, and if "AutoMETic" is disabled (**met** set to **manual)**, you can compute the PLOG using the previously-saved **oat** profile. Examples (i) Same example route from earlier. Specify a single **oat** (10 C, units as per UK **culture**) to be used for entire route. Also demonstrated are analogous settings for **qnh** (1009 mb), **elev** (35 feet), **lapserate** (-1.97 degrees-C-per-thousand-feet). **route=EGNS, >350/20, IOM348/51, EGPK; oat=10; qnh=1009; elev=35; lapserate=-1.97; culture=UK**

	It is very important to ensure that you have set the correct culture when specifying the atmospheric properties to make sure the units are consistent. Hence, in this example we enforce culture=UK since we want to ensure that the respective units for the quantities are interpreted as intended by the numerical values. (ii) Same but with atmospheric conditions varying along the route (e.g., typically as a a function of location, determined manually from forecasts etc) (in the context of the previously-assigned sticky **culture** value of **UK** for the correct wind-speed units) **route=EGNS, >350/20, IOM348/51, EGPK; oat=10,9,8,8; qnh=1009,1009,1008,1007; elev=35,0,0,100; lapserate=-1.97,-1.97,-1.98,-1.98** (iii) Same but with fewer data points per parameter than number of waypoints.The last specified value (per parameter) will be assumed for all remaining waypoints. **route=EGNS, >350/20, IOM348/51, EGPK; oat=10,9; qnh=1009,1009,1008; elev=35,0,0,100; lapserate=-1.98**
qnh	Multi-valued parameter specifying the sea level atmospheric pressure (QNH) for use in the atmospheric calculations (within the PLOG) if **met** set to **manual** such that "AutoMETic" has been deactivated. Applies last specified value to all following waypoints if number of values specified is less than the number of waypoints. Each value is specified as sea-level-pressure in units as per **culture,** pertaining to mean-sea-level coincident with the waypoint coordinates. Default value of 1013.25 millibars (in equivalent units as per **culture**) is assumed if **qnh** is unspecified. When used in combination with the **save** parameter, the **qnh** values are saved with the route so that when the route is re-used via it's **routeid** parameter, and if "AutoMETic" is disabled (**met** set to **manual)**, you can compute the PLOG using the previously-saved **qnh** profile. (For examples, see **oat** entry examples which include **qnh** parameter examples.)
elev	Multi-valued parameter specifying the surface elevation at which **oat** is specified for use in the atmospheric calculations (within the PLOG) if **met** set to **manual** such that "AutoMETic" has been deactivated. Applies last specified value to all following waypoints if number of values specified is less than the number of waypoints. Each value is specified as surface elevation AMSL (above mean sea level) in units as per **culture** pertaining to the ground surface location (at which **oat** is specified)**.**

	Default value of 0 is assumed if **elev** is unspecified. When used in combination with the **save** parameter, the **elev** values are saved with the route so that when the route is re-used via it's **routeid** parameter, and if "AutoMETic" is disabled (**met** set to **manual)**, you can compute the PLOG using the previously-saved **elev** profile. (For examples, see **oat** entry examples which include **elev** parameter examples.)
lapserate	Multi-valued parameter specifying the temperature-lapse-rate for use in the atmospheric calculations (within the PLOG) if **met** set to **manual** such that "AutoMETic" has been deactivated. Applies last specified value to all following waypoints if number of values specified is less than the number of waypoints. Each value is specified as temperature-lapse-rate in units as per **culture,** assumed to be constant with altitude, pertaining to the location coincident with the waypoint coordinates. Default value of -1.981 degrees-C-per-thousand-feet (in equivalent units as per **culture**) is assumed if **lapserate** is unspecified. When used in combination with the **save** parameter, the **lapserate** values are saved with the route so that when the route is re-used via it's **routeid** parameter, and if "AutoMETic" is disabled (**met** set to **manual)**, you can compute the PLOG using the previously-saved **lapserate** profile. (For examples, see **oat** entry examples which include **lapserate** parameter examples.)

Unit dimensions per culture

	UK	**UK GAL**	**US**
airspeed	knots	knots	knots
wind direction	degrees true	degrees true	degrees true
wind speed	knots	knots	miles per hour
climb rate	feet per minute	feet per minute	feet per minute
fuel capacity	litres	UK gallons	US gallons
fuel flow rate	litres per hour	UK gallons per hour	US gallons per hour
atmospheric temperature	celsius (C)	celsius (C)	fahrenheit (F)

atmospheric pressure	millibars	millibars	inches of mercury
atmospheric temperature lapse rate	degrees celsius per thousand feet	degrees celsius per thousand feet	degrees celsius per thousand feet
distance / range	nautical miles	nautical miles	nautical miles
altitude	feet	feet	feet
transition altitude	feet	feet	feet
flight levels	thousands of feet divided by 100	thousands of feet divided by 100	thousands of feet divided by 100
elevation (airfield, MET station etc.) AMSL	feet	feet	feet

Waypoint Searching

When searching for waypoints in the FlyLogical database, it is normally sufficient to specify either the Name (e.g., EGNS) or the Description (e.g., Ronaldsway) **BUT NOT BOTH**. However, on occasion, if Name or Description are shared by multiple waypoints, then it is necessary to zoom-in on the type of waypoint in order to make the distinction. This is achieved by simply adding (a space followed by) the (full or initial sub-string of the) type descriptor after the Name (or Description). For example, if you don't know a priori the ICAO code for the airfield called Goodwood and you search simply on the description **Goodwood** this will return two entries: **EGHR** and **GWC**, where **EGHR** is the airfield called Goodwood and **GWC** is the VOR beacon sharing the same name. In order to zoom-in on the airfield, specify **Goodwood Airfield** which will uniquely resolve to **EGHR**. Likewise, searching for **Goodwood VOR** will uniquely resolve to GWC. The following table contains all the available type descriptors which you can use to decorate your waypoint searches (not case-sensitive).

Available waypoint type descriptors for use in searching

airfield (civil)
airfield (mil)
airfield (civil/mil)
private strip
microlight/gliders

helipad (civil)
helipad (mil)
helipad (civil/mil)
private helipad
vor
vor/dme
dme
ndb
tacan
vortac
other navaid
vrp
reporting point
user waypoint

Web Apps

The **iNavCalc** web-apps provide you with a web-browser based interface to manage your routes, PLOGs, and waypoints. This can be more convenient to use than the email subject-line interface when you are performing administrative tasks such as creating & editing routes and waypoints. The web-apps are available via the following links:

> **iNavCalc Route Manager (routes management, PLOG requests)**
> **iNavCalc Waypoint Manager (waypoints management)**
> **iMETbrief (separate "companion" app, METAR and TAFs only)**

The usage of the web-apps should be self-explanatory. But here are a few tips and hints:

- The **iNavCalc** apps feature grids for displaying lists of routes, waypoints, etc. These grids share the following common features:

- **Sorting**: simply click on the given column header in a given grid to sort by that entity. Multi-column sorting is supported. Re-click to reverse the sort-order, and re-clock again to switch-off sorting on the given column
- **Filtering** (searching): certain columns (such as Route name, Waypoint Name, etc) have an associated filter (or search field). It has an icon which looks like an oil filter/funnel. Use this to limit the list (and de-clutter your screen) by specifying the desired content in the filter edit-box, e.g., to zoom-in on single entity, specify it's name in the filter edit-box, and set the filter to "contains" (each filter has a range of options/actions available which are evident when clicking on the filter icon -- "contains" is the default). Multi-column filtering is supported where appropriate.

- **To share or not to share ?** iNavCalc provides a sharing mechanism whereby you can choose to share your routes and waypoints either with other specified (registered) users, or with everyone (public). Here are some guidelines on sharing "best-practice":

  - If you are planning to share a waypoint, make sure that you use sensible naming conventions so that the database search algorithms can make best use of the meta information. For example, if you have defined an airfield that you are planning on sharing, ensure that you use the ICAO designator (4 character string, e.g., KLAX for Los Angeles International, EGNH for Blackpool UK) -- if it exists -- for the Name of the waypoint (you can put the common-names such as Los Angeles, Blackpool, IATA airport names, etc in the Description). This provides for more efficient/consistent searching since the ICAO designator is globally unique. Note: the search algorithm uses the Name and the Description field in a "UNION" search, but returns the Name for inclusion in the route/PLOG. You **should not combine** Name and Description tags in a single search: you should seek one or the either in a given search. Combining them will result in a failed search. For example, if you specify the Description contents in the route search parameters: **route=Palo Alto, Half Moon Bay** , the search algorithm will successfully resolve to a route between **KPAO** (ICAO designator for **Palo Alto**) and **KHAF** (ICAO designator for **Half Moon Bay**) because we took care to enter **KPAO** and **KHAF** in the respective Name fields and **Palo Alto** and **Half Moon Bay** in the respective Description fields when creating these example public waypoints. This is the pattern we strongly recommend that you follow. Note: you can of course construct the same route using the Names rather than Descriptions in the search tags i.e., via **route=KPAO, KHAF** . However, if you try to specify both Name and Descriptions together at the same time, the search will fail. So, for example, the following would fail: **route=KPAO Palo Alto, KHAF Half Moon Bay** .

  - Make sure that your waypoint location coordinates are absolutely correct before making public (for obvious reasons).

    - Make sure that you correctly specify the Country for the given waypoint when you are creating it. This helps the database search when looking for “best-fit”. This is particularly important when working with navaids whereby the same name is used for navaids in different countries.

    - Likewise, make sure that you specify the correct type for your waypoint (airfield, navaid, etc), since incorrect **type descriptor** entries will be confusing.

    - Only make a waypoint public if it will be truly useful to the public. Examples of such would be airfields and navaids. Examples which should not be made public would be your own “pet” waypoints “in the middle of the countryside” that you happen to use on your personal nav sorties.

    - Generally avoid making your routes public since these will generally be of little relevance to all other users, and will clutter their search grids. Instead, only make a route public if you expect that it will be truly interesting to all.

    - If you wish to make waypoints and/or routes available to a limited group of known individuals(e.g., friends, members of a flying club, participants in a fly-out, etc), then share with these individuals explicitly (i.e., rather than making public).

- **Use responsibly**. The infrastructure resources (servers, network, etc) are finite and provided to you for free. Avoid unnecessary saving of waypoints and routes in the database: only save what is truly useful (remember: you can always create a route whenever you need it). Also, avoid unnecessary or frivolous MET and PLOG requests: only make a request when you really want/need it. We retain the right to suspend access to any user who we deem to be abusing the privileges offered in our free service. Moreover, if the system continues to get “snowed under” because of frivolous/irresponsible use by many users, we will simply switch off the public access. In other words, “play nice” (even if only for reasons of enlightened self-interest).

Troubleshooting

Failing AutoMETic

On occasion you may see the following warning in your PLOG response email:

**** AutoMETic has FAILED, reason unknown. Default weather parameters (zero wind, ISA temperatures and pressures) have been used in the attached PLOG. In case the failure was due to a network error, try submitting your PLOG request again. Otherwise, you will have to set MET=MANUAL and specify all weather parameters manually ****

This is invariably due to network connectivity errors, unknown server problems, etc, and is generally beyond the control of FlyLogical since we obtain weather data from third-parties via

the internet. Basically, the problem should self-resolve (i.e., once the external network/server issues get fixed their end). So, the only advice we can offer is to try re-submitting your PLOG request(s) later (or, specify weather parameters manually).

History and background

iNavCalc has been developed as a personal “pet project”, born out of the need for a quick, convenient but comprehensive method of generating PLOGs on demand for VFR private flying, from any device. It now has thousands of users, worldwide.

Disclaimer

FlyLogical applications (**iNavCalc** and **iMETbrief**) retrieve information from various free and unregulated sources on the internet and is **NOT** associated with, or approved by, any official body. Information provided is **NOT** guaranteed to be accurate and is used **strictly at your own risk**. If you pass on information generated by FlyLogical applications, you **MUST** inform the recipient of it's origination and the essence of this warning. Briefings issued from FlyLogical applications should be read in conjunction with the original and official sources before flight to ensure that the content is complete and not unduly changed or corrupt.