NEAREST AIRPORT GROUP

A NearestAirport search returns a list of airports nearest the reference point that is normally set using the current position of the aircraft. It sorts the data by ascending distance. In very large searches, ascending distance isn't strictly maintained, but in smaller searches up to 10 items or so, ascending order seems always to be maintained.

Common with all Nearest searches, not all available airport information can be obtained in a NearestAirport search, only the variables that begin with NearestAirport. If airport frequencies, runways, transitions, etc, are needed, then an ICAO transfer into WaypointAirport must be performed. Refer to the ICAO Transfer section.

- **NearestAirportCurrentLatitude**
- **NearestAirportCurrentLongitude (degrees, radians) [Get, Set]**

Latitude and Longitude of the reference point, usually the aircraft. Input is in degrees (decimal format, not deg, min, sec) or radians.

- **NearestAirportMaximumItems (enum) [Get, Set]**

The limit of numbers of items to be returned in the search.

- **NearestAirportMaximumDistance (enum) [Get, Set]**

Maximum search radius. Especially in large searches, MaximumDistance is not strictly adhered to. For some reason apparently having to do with the way the search algorithm works, some searches return items 20 to 25% more distant than MaximumDistance.

- **NearestAirportItemsNumber (enum) [Get]**

The number of airports actually returned in the NearestAirport search.

- **NearestAirportCurrentLine (enum) [Get, Set]**

The Index pointer. Refer to the GPS Database Search section for further description.

- **NearestAirportCurrentICAO (string) [Get]**

The ICAO of each airport retrieved in the NearestAirport search.

❑ NearestAirportCurrentIdent (string) [Get]

The 3 to 4 character Ident of each airport retrieved in the NearestAirport search.

❑ NearestAirportCurrentAirportKind (enum) [Get]

A number representing Airport Class.

WaypointAirportKind (Class)

#	Class (Kind)	#	Class (Kind)
0	UNKNOWN_KIND_AIRPORT = 0	3	WATER_SURFACE_AIRPORT = 3
1	HARD_SURFACE_AIRPORT = 1	4	HELIPAD_AIRPORT = 4
2	SOFT_SURFACE_AIRPORT = 2	5	PRIVATE_AIRPORT = 5

http://msdn.microsoft.com/en-us/library/cc526954.aspx#AirportClass

❑ NearestAirportCurrentLongestAirportDirection (degrees) [Get]

Direction (magnetic) of the longest runway.

❑ NearestAirportCurrentDistance (nmiles) [Get]

The distance of each airport retrieved in the NearestAirport search from the reference point (NearestAirportCurrent Latitude and CurrentLongitude), which is normally set from the aircraft's current position.

❑ NearestAirportCurrentTrueBearing (degrees) [Get]

The bearing (true) from the reference point to each VOR retrieved in the NearestVor search.

❑ NearestAirportCurrentBestApproachEnum (enum) [Get]

A number representing the most precise approach available at the airport.

WaypointAirportBestApproach

#	Approach Type	#	Approach Type	#	Approach Type
0	UNKNOWN = 0	5	LORAN = 5	10	LDA = 10
1	VFR = 1	6	RNAV = 6	11	LOC = 11
2	HEL = 2	7	VOR = 7	12	MLS = 12
3	TACAN = 3	8	GPS = 8	13	ILS = 13
4	NDB = 4	9	SDF = 9		

http://msdn.microsoft.com/en-us/library/cc526954.aspx#AirportApproachType

❑ NearestAirportCurrentBestApproach (string) [Get]

The name of the most precise approach available at the airport. Refer to table above.

❑ NearestAirportCurrentComFrequencyName (string) [Get]

NearestAirportCurrentComFrequencyName is the abbreviation for the airport traffic control name if one is present at the airport. These include:

- ❑ twr = Tower
- ❑ CTF = Common Traffic Advisory Frequency (CTAF)
- ❑ uni = Unicomm
- ❑ mul = Multicomm

Ground, Clearance, Approach, Departure, ATIS, ASOS, AWOS, and FSS are not named by CurrentComFrequencyName. Those frequencies are available only from the WaypointAirport Group.

❑ NearestAirportCurrentComFrequencyValue (MHz) [Get]

The frequency value of CurrentComFrequencyName. If more than one tower is available at an airport, only the first frequency will be returned by CurrentComFrequencyValue. If no traffic control frequencies are available for the airport, CurrentComFrequencyValue returns 0.00.

```
NEAREST AIRPORT SEARCH

  49.3668 :Current Lat    15 :Max Items   15 :Items Num
 -97.1143 :Current Lon   100 :Max Distance

Current   ICAO      lll ---------------------- Current -------------------------
Line      123456789012 Ident Kind Rwy*  Dist  Brg Best Appr  Com     Freq Length
0         A       CKK7   CKK7    1  176  18.0   64    0        uni  122.80   3000
1         A       CJB3   CJB3    1  149  20.2   57    0        mul  122.70   3000
2         A       CJL6   CJL6    1  180  22.9  226    0        mul  123.20   3248
3         A       KA4     KA4    2  126  24.5  102    0             0.00   2875
4         A       KPMB   KPMB    1  160  25.9  191    7  VOR   CTF  122.80   3797
5         A       NA67   NA67    2  177  26.8  211    0             0.00   2000
6         A       CJL5   CJL5    2   96  29.7   11    0        mul  123.20   3000
7         A       8NA6   8NA6    2   87  30.5  214    0             0.00   3800
8         A       CKJ7   CKJ7    2  137  30.5  314    0        mul  123.40   3000
9         A       5ND3   5ND3    2  156  31.7  195    0             0.00   1600
10        A       CYWG   CYWG    1  188  32.9  352   13  ILS   twr  118.30  10988
11        A       CKZ7   CKZ7    1   90  33.7  250    0        uni  122.80   2900
12        A       MN57   MN57    2   96  35.3  168    0             0.00   2500
13        A       KHC0   KHC0    1  140  37.5  170    8  GPS   CTF  122.80   4002
14        A       CKA8   CKA8    2   17  37.5  333    0             0.00   3000
```

❑ NearestAirportCurrentLongestRunwayLength (feet) [Get]

Length of the longest runway at the airport.

NEAREST INTERSECTION GROUP

A NearestIntersection search returns a list of Intersections nearest the reference point that is normally set from the current aircraft position. It sorts the data by ascending distance. In very large searches, ascending distance isn't strictly maintained, but in smaller searches up to 10 items or so, ascending order seems always to be maintained.

Common with all Nearest searches, not all available Intersection information can be obtained in a NearestIntersection search, only the variables that begin with NearestIntersection. If Region, Nearest VOR Ident, Nearest VOR Type, Nearest VOR True Radial, Nearest VOR Magnetic Radial, or Nearest VOR Distance from an intersection following a NearestIntersection search is needed, then an ICAO transfer into WaypointIntersection must be performed. Refer to the ICAO Transfer section.

❑ NearestIntersectionCurrentLatitude (degrees, radians) [Get, Set]

Latitude of the reference point, usually the aircraft. Input is in degrees (decimal format, not deg, min, sec) or radians.

❑ NearestIntersectionCurrentLongitude (degrees, radians) [Get, Set]

Longitude of the reference point, usually the aircraft. Input is in degrees (decimal format, not deg, min, sec) or radians.

❑ NearestIntersectionMaximumItems (enum) [Get, Set]

The limit of numbers of items to be returned in the search. In practice, this should be kept realistically small so the NeasrestIntersection search returns data quickly. The gps_500 gauge, for example, sets this value to 9.

❑ NearestIntersectionMaximumDistance (enum) [Get, Set]

Maximum search radius. Especially in large searches, MaximumDistance is not strictly adhered to. For some reason apparently having to do with the way the search algorithm works, some searches return items 20 to 25% more distant than MaximumDistance.

❑ NearestIntersectionCurrentFilter (enum) [Get, Set]

NearestIntersectionCurrentFilter is a number between 0 and 255 that is the decimal equivalent of the 8 bit binary number that indicates which of the 8 Intersection types are to be included in the NearestIntersection search.

The Intersection types for FSX, and presumably also for FS9 are:

Bit	Name and Type #	Bit	Name and Type #
0	UNKNOWN = 0	4	NDB = 4
1	NAMED = 1	5	OFFROUTE = 5
2	UNNAMED = 2	6	IAF = 6
3	VOR = 3	7	FAF = 7

http://msdn.microsoft.com/en-us/library/cc526954.aspx#NearestIntersectionData

The default fs9gps NearestIntersectionCurrentFilter value is 230. The binary equivalent is*:

```
7 . . 4  3 . . 0 – Bit number (Bit 0 thru Bit 7 = 8 Bits)
1 1 1 0  0 1 1 0
```

which includes Intersection types 1, 2, 5, 6, and 7. These are **NAMED**, **UNNAMED**, **OFFROUTE**, **IAF**, and **FAF** intersections. That is, all Intersection types except **VOR** and **NDB**.

❑ NearestIntersectionAddIntersectionType (enum) [Set]

NearestIntersectionAddIntersectionType is an enum (not a binary) that adds an Intersection type to the NearestIntersection search.

❑ NearestIntersectionRemoveIntersectionType (enum) [Set]

NearestIntersectionRemoveIntersectionType is an enum (not a binary) that removes an Intersection type from the NearestIntersection search.

❑ NearestIntersectionSetDefaultFilter (enum) [Set]

NearestIntersectionSetDefaultFilter returns the NearestIntersection search to the default value CurrentFilter = 230. The proper syntax is:

```
(>C:fs9gps:NearestIntersectionSetDefaultFilter)
```

Note that an argument is not required, however you can include anything you want as log as the '>' is included.

EXAMPLE: NearestIntersection Add, Remove, and SetDefault

To begin with, an example search result using the default filter setting:

```
NEAREST INTERSECTION SEARCH

  37.6625 :Curr Lat  10 :Max Items  10 :Items Num
 -95.4875 :Curr Lon  50 :Max Dist  230 :Filter

---------- NearestIntersectionCurrent ----------
          ICAO     111
Line      123456789012    Ident  Type   Dist   Brg
0         WK3KCNUFLOUR    FLOUR     1    0.0   187
1         WK3KCNUMACEZ    MACEZ     1    0.3   331
2         WK3    BABGE    BABGE     1    3.4   162
3         WK3KCNUMYYER    MYYER     1    5.0   187
4         WK3K88 ABUYA    ABUYA     1    8.1    26
5         WK3KPPFAGNEW    AGNEW     1    9.5   186
6         WK3KPPFFF17      FF17     2   14.5   184
7         WK3KPPFDENUM    DENUM     1   16.0   182
8         WK3K88 SAVOF    SAVOF     1   17.7    19
9         WK3KPPFBATTR    BATTR     1   17.9   194
```

A few points of interest in the 12 character ICAO identifier include:

- The first character of the 12 character ICAO identifier, ICAO Type, is "W" for Intersection types 1 and 2 = NAMED and UNNAMED (Waypoint) Intersections, "V" for Intersection type 3 = VOR Intersection, and "N" for Intersection type 4 = NDB Intersection.
- For Type 1 NAMED Intersections, the Intersection Idents are 5 letters long (character positions 8 through 12) and are often geographically recognizable words.
- Some Intersection ICAOs contain an Airport Ident listed in character positions 4 through 7. These are Terminal Waypoints and are displayed with blue intersection symbols △ on the FS9 map. Terminal Waypoints are part of fs9gps terminal procedures or part of approaches or departures towards and away from a runway. The ident of the airport that "owns" Terminal Waypoint is included in the ICAO. The ICAOs without an Airport Ident are enroute intersections generally used to create Victor Airway and Jet Airway routes. These are the magenta colored intersections △ on the FS9 map.

The fs9gps database appears to be populated with only 4 types of Intersections: Type 1 = NAMED, Type 2 = UNNAMED, Type 3 = VOR, and Type 4 = NDB. If correct, then including types 5, 6, and 7 is irrelevant.

(*) Binary numbers are read from right to left, the right-most digit is always Bit 0, the digit to its left is Bit 1, and so forth.

EXAMPLE: NearestIntersectionAddIntersectionType

To add Intersection Type 3 (VOR Intersection) to the search, the following xml is used:

```
<Update>
3 (>C:fs9gps:NearestIntersectionAddIntersectionType)
</Update>
```

which yields these search results:

```
NEAREST INTERSECTION SEARCH

  37.6625 :Curr Lat  10 :Max Items  10 :Items Num
 -95.4875 :Curr Lon  50 :Max Dist 238 :Filter

---------- NearestIntersectionCurrent ----------
          ICAO     111
Line      123456789012     Ident  Type   Dist   Brg
0         WK3KCNUFLOUR     FLOUR     1    0.0   187
1         WK3KCNUMACEZ     MACEZ     1    0.3   331
2         WK3     BABGE    BABGE     1    3.4   162
3         WK3KCNUMYYER     MYYER     1    5.0   187
4         VK3     CNU        CNU     3    5.5   247
5         WK3K88 ABUYA     ABUYA     1    8.1    26
6         WK3KPPFAGNEW     AGNEW     1    9.5   186
7         WK3KPPFFF17       FF17     2   14.5   184
8         WK3KPPFDENUM     DENUM     1   16.0   182
9         WK3K88 SAVOF     SAVOF     1   17.7    19
```

The VOR Intersection 'CNU' (Line 4) is in the search results and the filter value is 238.

EXAMPLE: NearestIntersectionRemoveIntersectionType

To subsequently remove Intersection Type 1 (Named Intersection) from the search, the xml would be:

```
<Update>
3 (>C:fs9gps:NearestIntersectionAddIntersectionType)
1 (>C:fs9gps:NearestIntersectionRemoveIntersectionType)
</Update>
```

which yields these search results:

```
NEAREST INTERSECTION SEARCH

  37.6625 :Curr Lat   10 :Max Items   10 :Items Num
 -95.4875 :Curr Lon   50 :Max Dist  236 :Filter

----------- NearestIntersectionCurrent ----------
          ICAO     111
Line      123456789012     Ident  Type   Dist   Brg
0         VK3     CNU        CNU     3    5.5   247
1         WK3KPPFFF17       FF17     2   14.5   184
2         WK3KPPFMA118     MA118     2   19.7   183
3         WK3KPPFFF35       FF35     2   25.5   182
4         WK3KJLNJLN31     JLN31     2   27.6   106
5         VK3     OSW        OSW     3   33.2   156
6         WK3KIDPMA023     MA023     2   33.7   205
7         WK3KUKLFF36       FF36     2   36.3   341
8         WK3KPTSFF166     FF166     2   36.7   104
9         WK3KCFVFF35       FF35     2   39.8   186
```

Intersection Type 1 has been removed and the filter value is 236.

EXAMPLE: NearestIntersectionSetDefaultFilter

If you wish to now reset the filter to the default setting, the xml would be:

```
<Update>
3 (>C:fs9gps:NearestIntersectionAddIntersectionType)
1 (>C:fs9gps:NearestIntersectionRemoveIntersectionType)
(>C:fs9gps:NearestIntersectionSetDefaultFilter)
</Update>
```

and the search results return to the default filter value 230 setting:

```
NEAREST INTERSECTION SEARCH

  37.6625 :Curr Lat   10 :Max Items   10 :Items Num
 -95.4875 :Curr Lon   50 :Max Dist  230 :Filter

----------- NearestIntersectionCurrent ----------
          ICAO     111
Line      123456789012     Ident  Type   Dist   Brg
0         WK3KCNUFLOUR     FLOUR     1    0.0   187
1         WK3KCNUMACEZ     MACEZ     1    0.3   331
2         WK3     BABGE    BABGE     1    3.4   162
3         WK3KCNUMYYER     MYYER     1    5.0   187
4         WK3K88 ABUYA     ABUYA     1    8.1    26
5         WK3KPPFAGNEW     AGNEW     1    9.5   186
6         WK3KPPFFF17       FF17     2   14.5   184
7         WK3KPPFDENUM     DENUM     1   16.0   182
8         WK3K88 SAVOF     SAVOF     1   17.7    19
9         WK3KPPFBATTR     BATTR     1   17.9   194
```

❑ NearestIntersectionItemsNumber (enum) [Get]

The number of Intersections actually returned in the NearestIntersection search.

❑ NearestIntersectionCurrentLine (enum) [Get, Set]

The Index pointer. Refer to the GPS Database Search section for further description.

❑ NearestIntersectionCurrentICAO (string) [Get]

The ICAO of each Intersection retrieved in the NearestIntersection search.

❑ NearestIntersectionCurrentIdent (string) [Get]

The 1 to 5 character Ident of each Intersection retrieved in the NearestIntersection search.

❑ NearestIntersectionCurrentType (enum) [Get]

The type of each Intersection retrieved in the NearestIntersection search. See NearestIntersectionCurrentFilter for additional discussion.

❑ NearestIntersectionCurrentDistance (nmiles or meters) [Get]

The distance of each Intersection retrieved in the NearestIntersection search from the reference point (NearestIntersectionCurrent Latitude and CurrentLongitude), which is normally set from the aircraft's current position.

❑ NearestIntersectionCurrentTrueBearing (degrees or radians) [Get]

The bearing (true) from the reference point to each Intersection retrieved in the NearestIntersection search.

NEAREST VOR GROUP

A NearestVor search returns a list of VORs nearest the reference point that is normally set using the current position of the aircraft. It sorts the data by ascending distance. In very large searches, ascending distance isn't strictly maintained, but in smaller searches up to 10 items or so, ascending order seems always to be maintained.

Common with all FS9 Nearest searches, not all available VOR information can be obtained in a NearestVor search, only the variables that begin with NearestVor. If Class, Name, Region, Elevation, Weather Broadcast, or Magnetic Variation of a VOR from a NearestVor search is needed, then an ICAO transfer into WaypointVor must be performed. Refer to the ICAO Transfer section. This is one area in which the gps module in FSX is much easier to work with.

- **NearestVorCurrentLatitude**
- **NearestVorCurrentLongitude (degrees, radians) [Get, Set]**

Latitude and Longitude of the reference point, usually the aircraft. Input is in degrees (decimal format, not deg, min, sec) or radians.

- **NearestVorMaximumItems (enum) [Get, Set]**

The limit of numbers of items to be returned in the search.

- **NearestVorMaximumDistance (enum) [Get, Set]**

Maximum search radius. Especially in large searches, MaximumDistance is not strictly adhered to. For some reason apparently having to do with the way the search algorithm works, some searches return items 20 to 25% more distant than MaximumDistance.

- **NearestVorCurrentFilter (enum) [Get, Set]**

For the fs9gps database, NearestVorCurrentFilter is a number between 0 and 63 that is the decimal equivalent of the 6 bit binary number that indicates which of the 6 FS9 VOR types are included in the NearestVor search.

I cannot locate documentation that lists the 6 VOR types, however, Microsoft's ESP SDK lists 8 types of VORs (FSX):

Bit	Name and Type #	Bit	Name and Type #
0	UNKNOWN = 0	4	TACAN = 4
1	VOR = 1	5	VORTAC = 5
2	VOR_DME = 2	6	ILS = 6
3	DME = 3	7	VOT = 7

http://msdn.microsoft.com/en-us/library/cc526954.aspx#VorType

The assumption is that the VOR types for FS9 are:

Bit	Name and Type #	Bit	Name and Type #
0	UNKNOWN = 0	3	DME = 3
1	VOR = 1	4	TACAN = 4
2	VOR_DME = 2	5	VORTAC = 5

The default fs9gps NearestVorCurrentFilter value is 62. The binary equivalent is*:

```
5 . . . . 0 – Bit number (Bit 0 thru Bit 5 = 6 Bits)
1 1 1 1 1 0
```

which includes all VOR types except UNKNOWN.

To include only VOR types 1, 2, and 3 in the NearestVor search, the binary would be*:

```
5 . . . . 0 – Bit number (Bit 0 thru Bit 5 = 6 Bits)
0 0 1 1 1 0
```

the decimal equivalent of which is 14. The xml instruction to set the filter would be:

```
14 (>C:fs9gps:NearestVorCurrentFilter)
```

However, the fs9gps database appears to be populated with only 3 types of VORs: Type 1 = VOR, Type 2 = VOR_DME, and Type 3 = DME, so the identities of types 4 and 5 are irrelevant.

* Binary numbers are read from right to left, the right-most digit is always Bit 0, the digit to its left is Bit 1, and so forth.

❑ NearestVorAddVorType (enum) [Set]

NearestVorAddVorType is an enum (not a binary) adds a VOR type to the NearestVor search. If the <Update> section contains the following statement:

```
<Update>
10 (>C:fs9gps:NearestVorCurrentFilter)
</Update>
```

then only VOR types 1 and 3 (decimal 10 = binary **001010**) will be included in the NearestVor search, and a typical search result would look like:

```
NEAREST VOR SEARCH

  42.9638 :Current Lat   20 :Max Items  11 :Items Num
 -88.9091 :Current Lon  300 :Max Dist   10 :Filter

----------------- NearestVorCurrent -----------------
      ICAO     111
Line  123456789012  Ident Type     Freq    Dist   Brg
0     VK5     BJB      BJB    1   109.80    43.9    51
1     VK5     VOK      VOK    3   110.40    83.1   315
2     VK5     OLK      OLK    1   110.40   183.5   123
3     VK5     CGG      CGG    1   109.80   208.2    59
4     VK5     TOL      TOL    3   112.50   241.2   108
5     VK5     AOH      AOH    1   108.40   259.0   120
6     VK3     EST      EST    1   110.40   256.7   278
7     VK5     MAH      MAH    1   114.90   261.4   115
8     VK3     ULM      ULM    3   111.30   255.9   290
9     VK5     SKE      SKE    3   112.20   268.6   189
10    VK5     BUD      BUD    1   109.80   296.9   116
```

But, if the <Update> section is further edited to include a NearestVorAddVorType instruction, the NearestVor search changes:

```
<Update>
10 (>C:fs9gps:NearestVorCurrentFilter)
2 (>C:fs9gps:NearestVorAddVorType)
</Update>
```

```
NEAREST VOR SEARCH

  42.9638 :Current Lat   20 :Max Items  20 :Items Num
 -88.9091 :Current Lon  300 :Max Dist   14 :Filter

----------------- NearestVorCurrent -----------------
      ICAO     111
Line  123456789012  Ident Type     Freq     Dist   Brg
0     VK5     MSN      MSN    2   108.60     21.8   300
1     VK5     JVL      JVL    2   114.30     25.8   200
2     VK5     BAE      BAE    2   116.40     28.9    71
3     VK5     BUU      BUU    2   114.50     31.4   121
4     VK5     LJT      LJT    2   112.50     39.2    77
5     VK5     BJB      BJB    1   109.80     43.9    51
6     VK5     RFD      RFD    2   110.80     46.1   196
7     VK5     ENW      ENW    2   109.20     48.3   117
8     VK5     HRK      HRK    2   117.70     49.6   104
9     VK5     DLL      DLL    2   117.00     51.3   314
10    VK5     LNR      LNR    2   112.80     57.2   291
11    VK5     OBK      OBK    2   113.00     61.4   136
12    VK5     OSH      OSH    2   111.80     63.5    14
13    VK5     PLL      PLL    2   111.20     65.8   205
14    VK5     FAH      FAH    2   110.00     66.9    43
15    VK5     DPA      DPA    2   108.40     69.0   159
16    VK5     ORD      ORD    2   113.90     73.5   142
17    VK5     VOK      VOK    3   110.40     83.1   315
18    VK3     DBQ      DBQ    2   115.80     86.3   248
19    VK5     MTW      MTW    2   111.00     88.0    37
```

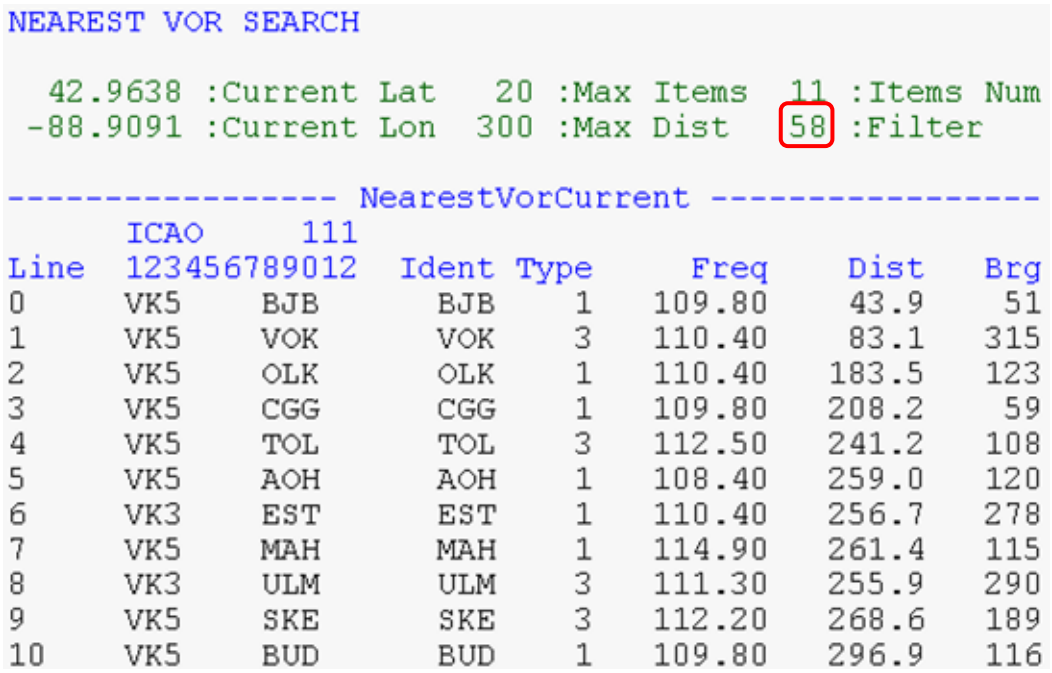
Now, VOR Type 2 has been added, the Filter value becomes 14 (binary **001110**, VOR types 1, 2, and 3), and additional VORs have been found within the 300 mile search radius.

❑ NearestVorRemoveVorType (enum) [Set]

NearestVorRemoveVorType functions in the opposite manner from AddVorType. If, for example, no NearestVorCurrentFilter instruction is given, the default CurrentFilter = 62 is assumed and all VOR types except UNKNOWN are included in the NearestVor search. The following will remove VOR Type =2 from the NearestVor search:

```
<Update>
2 (>C:fs9gps:NearestVorRemoveVorType)
</Update>
```

and the search result becomes:



```
NEAREST VOR SEARCH

  42.9638 :Current Lat   20 :Max Items  11 :Items Num
 -88.9091 :Current Lon  300 :Max Dist   58 :Filter

----------------- NearestVorCurrent -----------------
      ICAO     111
Line  123456789012  Ident Type     Freq    Dist   Brg
0     VK5    BJB      BJB    1   109.80    43.9    51
1     VK5    VOK      VOK    3   110.40    83.1   315
2     VK5    OLK      OLK    1   110.40   183.5   123
3     VK5    CGG      CGG    1   109.80   208.2    59
4     VK5    TOL      TOL    3   112.50   241.2   108
5     VK5    AOH      AOH    1   108.40   259.0   120
6     VK3    EST      EST    1   110.40   256.7   278
7     VK5    MAH      MAH    1   114.90   261.4   115
8     VK3    ULM      ULM    3   111.30   255.9   290
9     VK5    SKE      SKE    3   112.20   268.6   189
10    VK5    BUD      BUD    1   109.80   296.9   116
```

Only VOR types 1, 3, 4 and 5 are included in the search, and the Filter, which had been the default value 62, is now 58, binary **`111010`** (the fs9gps database appears to be populated with no Type 4 or 5 VORs, so none can be found in a search).

❑ NearestVorSetDefaultFilter (enum) [Set]

NearestVorSetDefaultFilter returns the NearestVor search to the default value CurrentFilter = 62. The proper syntax is:

```
(>C:fs9gps:NearestVorSetDefaultFilter)
```

Note that a value apparently is not required, however you can include anything you want. Whatever is already in the stack will be fine; a negative number, zero, positive number, decimal…

To recap:

```
<Update>
14 (>C:fs9gps:NearestVorCurrentVorType)
</Update>
```

sets the CurrentFilter to 14 (binary **`001110`**) and VOR types 1, 2, and 3 are included in the NearestVor search.

```
<Update>
14 (>C:fs9gps:NearestVorCurrentVorType)
2 (>C:fs9gps:NearestVorRemoveVorType)
</Update>
```

removes VOR Type 2, CurrentFilter becomes 10 (binary **001010**), and VOR types 1 and 3 are included in the NearestVor search.

```
<Update>
14 (>C:fs9gps:NearestVorCurrentVorType)
2 (>C:fs9gps:NearestVorRemoveVorType)
(>C:fs9gps:NearestVorSetDefaultFilter)
</Update>
```

resets CurrentFilter to 62 (binary **111110**) and all VOR types except UNKNOWN are included in the NearestVor search. The following SetDefaultFilter statements would all have done the same thing:

```
1 (>C:fs9gps:NearestVorSetDefaultFilter)
0 (>C:fs9gps:NearestVorSetDefaultFilter)
-5 (>C:fs9gps:NearestVorSetDefaultFilter)
12.378 (>C:fs9gps:NearestVorSetDefaultFilter)
```

but,

```
(C:fs9gps:NearestVorSetDefaultFilter)
```

will not work.

❑ NearestVorItemsNumber (enum) [Get]

The number of VORs actually returned in the NearestVor search.

❑ NearestVorCurrentLine (enum) [Get, Set]

The Index pointer. Refer to the GPS Database Search section for further description.

❑ NearestVorCurrentICAO (string) [Get]

The ICAO of each VOR retrieved in the NearestVor search.

❑ NearestVorCurrentIdent (string) [Get]

The 1 to 3 character Ident of each VOR retrieved in the NearestVor search.

❑ NearestVorCurrentType (enum) [Get]

The type of each VOR retrieved in the NearestVor search. It appears that only Type 1 = VOR, Type 2 = VOR DME, and Type 3 = DME exist in the fs9gps database. See **NearestVorCurrentFilter** for additional discussion.

❑ NearestVorCurrentFrequency (MHz) [Get]

The frequency of each VOR retrieved in the NearestVor search.

❑ NearestVorCurrentDistance (nmiles or meters) [Get]

The distance of each VOR retrieved in the NearestVor search from the reference point (NearestVorCurrent Latitude and CurrentLongitude), which is normally set from the aircraft's current position.

❑ NearestVorCurrentTrueBearing (degrees) [Get]

The bearing (true) from the reference point to each VOR retrieved in the NearestVor search.

NEAREST NDB GROUP

A NearestNdb search returns a list of NDBs nearest the reference point that is normally set from the current aircraft position. It sorts the data by ascending distance. In very large searches, ascending distance isn't strictly maintained, but in smaller searches up to 10 items or so, ascending order seems always to be maintained.

Common with all Nearest searches, not all available NDB information can be obtained in a NearestNdb search, only the variables that begin with NearestNdb. If Name, Region, Elevation, Weather Broadcast, or Magnetic Variation of a specific NDB found in a NearestNdb search is needed, then an ICAO transfer into WaypointNdb must be performed. Refer to the ICAO Transfer section.

❑ NearestNdbCurrentLatitude (degrees, radians) [Get, Set]

Latitude of the reference point, usually the aircraft. Input is in degrees (decimal format, not deg, min, sec) or radians.

❑ NearestNdbCurrentLongitude (degrees, radians) [Get, Set]

Longitude of the reference point, usually the aircraft. Input is in degrees (decimal format, not deg, min, sec) or radians.

❑ NearestNdbMaximumItems (enum) [Get, Set]

The limit of numbers of items to be returned in the search. In practice, this should be kept realistically small so the NeasrestNdb search returns data quickly. The gps_500 gauge, for example, sets this value to 9.

❑ NearestNdbMaximumDistance (enum) [Get, Set]

Maximum search radius. Especially in large searches, MaximumDistance is not strictly adhered to. For some reason apparently having to do with the way the search algorithm works, some searches return items 20 to 25% more distant than MaximumDistance.

❑ NearestNdbItemsNumber (enum) [Get]

The number of NDBs actually returned in the NearestNdb search.

❑ NearestNdbCurrentLine (enum) [Get, Set]

The Index pointer. Refer to the GPS Database Search section for further description.

❑ NearestNdbCurrentICAO (string) [Get]

The ICAO of each NDB retrieved in the NearestNdb search.

❑ NearestNdbCurrentIdent (string) [Get]

The 1 to 5 character Ident of each NDB retrieved in the NearestNdb search.

❑ NearestNdbCurrentType (enum) [Get]

The type of each NDB retrieved in the NearestNdb search.

The following is a list of NDB Type and Class, real life Transmission Power, real life Effective Range (which may not match how it is modeled in FS9):

0: Unknown. There appear to be no Type 0 NDBs in the fs9gps database.

1: Compass Locator. Below 25 watts, 15 - 25 nmiles. Type 1 NDBs are absent within the U.S.A. in the fs9gps database, but are common in other parts of the world, especially Europe (eg, U.K.).

2: MH. Below 50 watts, 25 - 50 nmiles. Directional Beacon Approach Facility found at or near airports where it is the primary approach aid. This is the most common type of NDB in the fs9gps database.

3: H. 50 to 1,999 watts, 50 - 75 nmiles. Enroute Airway Beacon, common in Canada and Caribbean

4: HH. 2,000+ watts, 75 - 125 nmiles. High powered Beacon found along coasts in the U.S.A.

An example of a NearestNdb search:

```
NEAREST NDB SEARCH

  52.7392 :Curr Lat    50 :Max Items    12 :Items Num
  -0.0070 :Curr Lon    50 :Max Distance

---------------- NearestVorCurrent ----------------
         ICAO     111
Line     123456789012  Ident  Type  Freq   Dist  Brg
0        NEG     FNL      FNL      4   401    0.8  281
1        NEG     CWL      CWL      2   423   24.5  315
2        NEG     BOU      BOU      2   392   31.7  183
3        NEG     CAM      CAM      2   333   32.5  168
4        NEG     LE        LE      2   384   38.2  258
5        NEG     NN        NN      2   379   39.1  228
6        NEG     NOT      NOT      2   430   40.4  286
7        NEGEGTCCIT       CIT      2   850   41.7  209
8        NEG     CIT      CIT      2   850   41.7  209
9        NEG     EME      EME      2   354   43.5  278
10       NEG     NWI      NWI      1   343   47.4   94
11       NEG     CM        CM      3   314   49.3   76
```

The "CIT" NDB (CurrentLine 7) contains the "EGTC" airport ident in character positions 4 through 7. This NDB serves as a Terminal Waypoint in a fs9gps approach procedure. The airport EGTC is the "owner" of the waypoint, consequently, its ident is included in the NDB ICAO.

❑ NearestVorCurrentFrequency (kHz) [Get]

The frequency of each NDB retrieved in the NearestNdb search, usually expressed in KhZ units.

❑ NearestNdbCurrentDistance (nmiles) [Get]

The distance of each NDB retrieved in the NearestNdb search from the reference point (NearestNdbCurrent Latitude and CurrentLongitude). The reference point is normally set from the aircraft's current position.

❑ NearestNdbCurrentTrueBearing (degrees or radians) [Get]

The bearing (true) from the reference point to each NDB retrieved in the NearestNdb search.

NEAREST AIRSPACE GROUP

All Nearest searches require the latitude and longitude of the reference point, also known as the Current point, which is normally the aircraft location, plus specified limitations to the amount of data you want to be returned in the search: maximum search distance (radius) and maximum number of returned items.

Nearest Airspace searches, however, require more information to define the search than Nearest Airport, Intersection, VOR, and NDB searches because an airspace is a three dimensional shape rather than a one dimensional point. Another feature of Nearest Airspace searches is that some gauges that make use of NearestAirspace, like the stock MSFS gps_500 gauge, issue messages to the pilot when the aircraft is simply *near* an airspace. Consequently, "closeness" to an airspace must also be defined.

Required information for a NearestAirspace search includes CurrentLatitude, CurrentLongitude, CurrentAltitude, TrueGroundTrack, GroundSpeed, NearDistance, NearAltitude, AheadTime, Query, MaximumItems and MaximumDistance. The first 5 are constantly changing in flight and are usually input via an A: variable. The last 6 would usually not be changed during flight, and are input using standard value declarations in your code.

- **NearestAirspaceCurrentLatitude**
- **NearestAirspaceCurrentLatitude (degrees) [Get, Set]**

The location of the reference point, expressed as latitude and longitude. The units of Lat/Lon can be degrees (formatted +/-ddd.dddd where S16 degrees 30 minutes would be formatted as -16.5000) or radians (d.dddd). In most applications, the reference point for Nearest searches is the current aircraft location.

(A:PLANE LATITUDE, Radians) (>@c:NearestAirspaceCurrentLatitude, Radians)
(A:PLANE LONGITUDE, Radians) (>@c:NearestAirspaceCurrentLongitude, Radians)

Some people prefer use of A:PLANE LATITUDE / LONGITUDE rather than A:GPS POSITION LAT / LON because A:PLANE is updated every gauge update cycle whereas A:GPS is updated every one second (referring to time, as in 1/60th of a minute).

- **NearestAirspaceCurrentAltitude (feet) [Get, Set]**

Altitude (ASL) of the reference point, which is normally the aircraft. Common units are feet or meters.

```
(A:GPS POSITION ALT, feet)
(>@c:NearestAirspaceCurrentAltitude, feet)
```

Of course, `(A:PLANE ALTITUDE, feet)` works also.

❑ NearestAirspaceTrueGroundTrack (degrees) [Get, Set]

Ground track of the aircraft relative to true north.

```
(A:GPS GROUND TRUE TRACK, degrees)
(>@c:NearestAirspaceTrueGroundTrack, degrees)
```

❑ NearestAirspaceGroundSpeed (knots) [Get, Set]

Ground speed of the aircraft.

```
(A:GPS GROUND SPEED, knots)
(>@c:NearestAirspaceGroundSpeed, knots)
```

❑ NearestAirspaceNearDistance (nmiles) [Get, Set]

NearestAirspaceNearDistance is the horizontal distance between the aircraft and an airspace boundary at which point NearestAirspaceCurrentStatus changes and airspace encroachment messages can be issued. It is discussed more completely in the NearestAirspaceCurrentStatus section later in this chapter.

The default is 2 nmiles.

❑ NearestAirspaceNearAltitude (feet) [Get, Set]

NearestAirspaceNearAltitude is a vertical distance buffer applied to the current aircraft altitude such that, if the aircraft is within + / - NearAltitude of an airspace floor or ceiling, the NearestAirspaceCurrentStatus changes and airspace encroachment messages can be issued. It is discussed more completely in the NearestAirspaceCurrentStatus section later in this chapter.

The default is 200 feet.

❑ NearestAirspaceAheadTime (minutes) [Get, Set]

NearestAirspaceAheadTime is the time separation between the aircraft and an airspace boundary at which point NearestAirspaceCurrentStatus changes and airspace encroachment messages can be issued. It is computed using NearestAirspaceGroundSpeed and is discussed more completely in the NearestAirspaceCurrentStatus section later in this chapter.

The default is 10 minutes.

❑ NearestAirspaceQuery (6 digit Hexadecimal 'enum') [Get, Set]

NearestAirspaceQuery tells the gps.dll which type(s) of airspaces to include in the NearestAirspace search. This is somewhat analogous to IcaoSearchStartCursor which tells the gps.dll which types of facilities to include in an IcaoSearch.

NearestAirspaceQuery is expressed in Hexadecimal format to more easily define the types of airspaces to include. It is a six digit hex number which represents 24 bits of information (6 hex digits x 4 = 24 bits). Since each bit is a 1 or 0, it functions as an individual 'include' / 'do not include' switch. Each of the bits is mapped to a specific airspace type. If the bit corresponding to Class C airspace (Bit 4) is set to 1, then the NearestAirspace search will include Class C airspaces, otherwise it will not. Any combination of airspace types can be searched by setting the right bits.

On line 83 of the gps_500 xml gauge there is a parameter declaration named kDisplayedAirspaces that is assigned the hex value **0xEFC038**. Converting this hex number to binary yields:

```
2..2 ...1 ...1
3..0 ...6 ...2 ...8 ...4 ...0 – Bit number (Bit 0 thru Bit 23 = 24 Bits)
1110 1111 1100 0000 0011 1000
```

which contains 12 "1s", meaning that there are 12 airspace types included in kDisplayedAirspaces.

The complete list of airspace types (24 in total) is:

Airspace Bit Map Table

Bit	Name and Type #	Bit	Name and Type #	Bit	Name and Type #
0	NONE = 0	8	CLASS_G = 8	16	PROHIBITED = 16
1	CENTER = 1	9	TOWER = 9	17	WARNING = 17
2	CLASS_A = 2	10	CLEARANCE = 10	18	ALERT = 18
3	CLASS_B = 3	11	GROUND = 11	19	DANGER = 19
4	CLASS_C = 4	12	DEPARTURE = 12	20	NATIONAL_PARK = 20
5	CLASS_D = 5	13	APPROACH = 13	21	MODE_C = 21
6	CLASS_E = 6	14	MOA = 14	22	RADAR = 22
7	CLASS_F = 7	15	RESTRICTED = 15	23	TRAINING = 23

http://msdn.microsoft.com/en-us/library/cc526954.aspx#FAC_BV_TYPE

Therefore, reading **1110 1111 1100 0000 0011 1000** from *right to left,* (the right-most digit is always Bit 0, the digit to its left is Bit 1, and so forth) and comparing each Bit to the Airspace Bit Table, above, the NearestAirspace search of kDisplayedAirspaces will include **CLASS_B, CLASS_C, CLASS_D, MOA, RESTRICTED, PROHIBITED, WARNING, ALERT, DANGER, MODE_C, RADAR,** and **TRAINING** airspace types. In fact, note that these are listed in the comment line (line 82) directly above the kDisplayedAirspaces declaration.

The other NearestAirspaceQuery listed in the gps_500 gauge is kAlwaysDisplayedAirspaces = **0x0FC000** (line 85). Its binary equivalent is:

```
2..2 ...1 ...1
3..0 ...6 ...2 ...8 ...4 ...0 – Bit number
0000 1111 1100 0000 0000 0000
```

which means that kAlwaysDisplayedAirspaces includes **MOA**, **RESTRICTED**, **PROHIBITED**, **WARNING**, **ALERT**, and **DANGER** airspace types.

To include any combination of airspaces, create a binary number by indicating 1 or 0 (include or don't include) for each of the 24 airspace types. Start with Airspace Type 0 (Bit 0) and work up the list to Airspace Type 23 (Bit 23), building the number from right to left. Once a 24 digit number is assembled from this selection process, convert it to hexadecimal format and enter that number into NearestAirspaceQuery.

If you wanted to include **TRAINING**, **RADAR**, **DANGER**, **ALERT**, **WARNING**, **MOA**, and **CLASS_C** airspaces in a NearestAirspace search, the binary would be:

```
2..2 ...1 ...1
3..0 ...6 ...2 ...8 ...4 ...0 – Bit number
1100 1110 0100 0000 0001 0000
```

the hexadecimal equivalent of which is **0xCE4010**. The proper xml instruction is:

```
0xCE4010 (>C:fs9gps:NearestAirspaceQuery)
```

It is acceptable to enter the decimal equivalent instead, but that approach may not make as much sense given the binary origin of the number:

```
13516816 (>C:fs9gps:NearestAirspaceQuery)
```

❑ NearestAirspaceMaximumItems (enum) [Get, Set]

Maximum number of airspace sectors that will be included in the search results. After this number of items is reached, the search process terminates.

```
9 (>@c:NearestAirspaceMaximumItems)
```

In the gps_500 gauge, "9" is used for maximum search items, matching the capability of the Garmin GNS 500 / 530 / 530A system after which it is modeled. When a Nearest search concludes, the list of 9 items (Airports, Intersections, VORs, NDBs, or Airspaces) is displayed using a {loop} within a <String> expression, with the Nearest being at the top.

❑ NearestAirspaceMaximumDistance (nmiles, meters) [Get, Set]

The maximum distance (radius) the search will extend from the reference point. If the search reaches this limit, it will terminate and Airspaces beyond MaximumDistance will not be included (by definition).

The NearestAirspace search will terminate when the earlier of NearestAirspaceMaximumItems or NearestAirspaceMaximumDistance is reached.

```
100 (>@c:NearestAirspaceMaximumDistance, nmiles)
```

❑ NearestAirspaceItemsNumber (enum) [Get]

The number of items actually retrieved during the NearestAirspace search, within the limitation specified by MaximumItems.

❑ NearestAirspaceCurrentLine (enum) [Get, Set]

The Index pointer. NearestAirspace search returns a list of airspace sectors. Setting NearestAirspaceCurrentLine allows you to select a specific sector in the list. Index numbers always start at 0.

❑ NearestAirspaceCurrentName (string) [Get]

The name of the airspace. For many airspace types, this is a descriptive, often recognizable name associated with the particular airspace location. In NearestAirspace searches, different sectors of the same airspace will sometimes have the same CurrentName, but are differentiated by different CurrentMinAltitude and CurrentMaxAltitudes and different CurrentNearDistance and CurrentAheadTime.

Center (type 1) airspaces have 8 character names constructed from the Region ID, the Center ID and Airspace names. As an example, the Center airspace name that Rotterdam, the Netherlands is within begins with the Netherlands Region ID, "EH", then the Center ID, "AA", then the Airspace name, 2302 which forms CurrentName EHAA2302. In the United States, the single character Region ID "K" is used, followed by the Air Route Traffic Control Center ID, then the Airspace name. For example, the Center airspace that Chicago O'Hare International airport is within is named KZAU4988. "ZAU" is the ARTCC ID, and 4988 is the Airspace name.

- **NearestAirspaceCurrentType (enum) [Get]**

A number representing airspace type. Refer to the Airspace Bit Table in NearestAirspaceQuery section. In MSFS, it is possible to be inside more than one airspace type or sector at the same time.

- **NearestAirspaceCurrentFrequency (MHz) [Get]**
- **NearestAirspaceCurrentFrequencyName (string = "Center") [Get]**

NearestAirspaceCurrentFrequency is the radio frequency (MHz) of Center airspace types found in a NearestAirspace search. Center airspaces types (type=1) must be included in NearestAirspaceQuery in order for CurrentFrequency to return a value other than 0.0. Apparently, only Center airspaces have an associated CurrentFrequency and CurrentFrequencyName. Airspace types 9, 10, 11, 12, and 13 (Tower, Clearance, Ground, Departure, and Approach) do not appear to have an associated CurrentFrequency and CurrentFrequencyName.

A NearestAirspace search that includes Centers may return several Center airspaces, all with different CurrentFrequency but the same CurrentFrequencyName, "Center". Centers have CurrentMinAltitude=0 (ground surface) and CurrentMaxAltitude=100000 meters or 328084 feet (edge of space).

As with all NearestAirspace search results, the airspace with the highest NearestAirspaceCurrentStatus will be listed first, then airspaces will be sorted by increasing distance from the reference point, which is usually the aircraft.

- **NearestAirspaceCurrentMinAltitude (feet) [Get]**
- **NearestAirspaceCurrentMaxAltitude (feet) [Get]**

The floor and ceiling of the airspace sector, measured above mean sea level. In fs9gps nomenclature, a value of 0 for MinAltitude refers to the ground surface. On aeronautical maps, it would be abbreviated "SFC"; in fs9gps terms, "0".

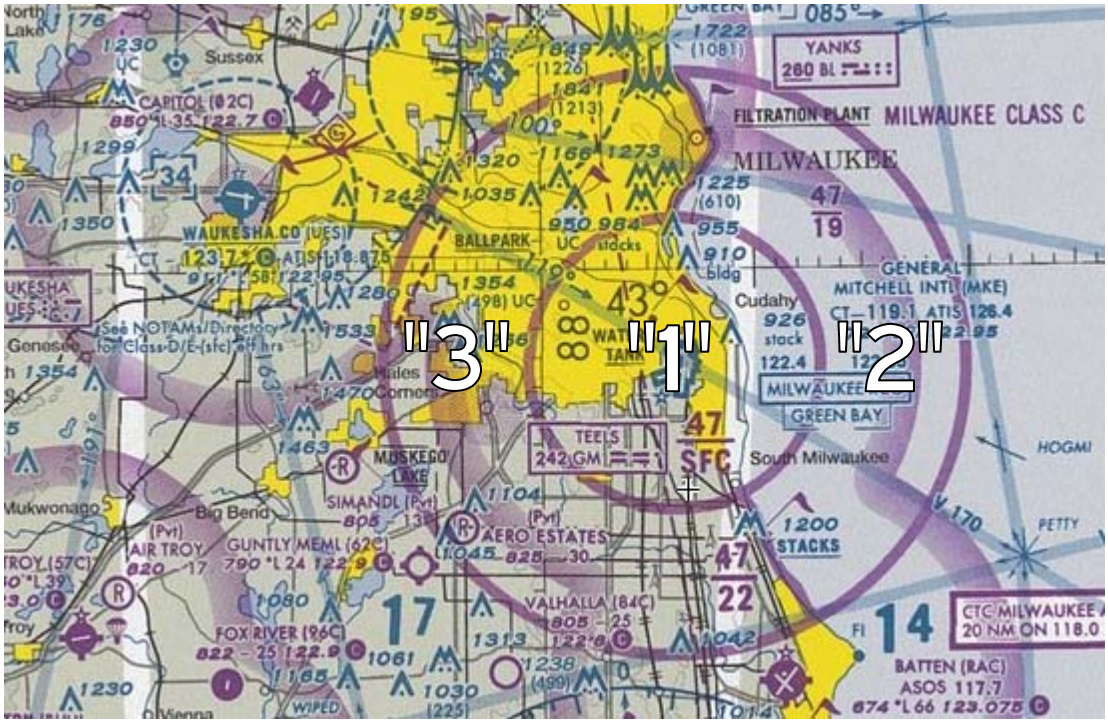
- **NearestAirspaceCurrentStatus (enum: 0, 1, 2, 3, or 4) [Get]**

NearestAirspaceCurrentStatus is a number that reflects an aircraft's positional status relative to nearby airspaces. It is determined from the "closeness" of the aircraft to airspace sectors in either x, y, z space or time. CurrentStatus, in turn, controls MessageItemsNumber and MessageCurrentType that can be used to display warning messages to the pilot. How close the aircraft needs to be to trigger a CurrentStatus change is determined by variables NearDistance, NearAltitude, and NearTime.

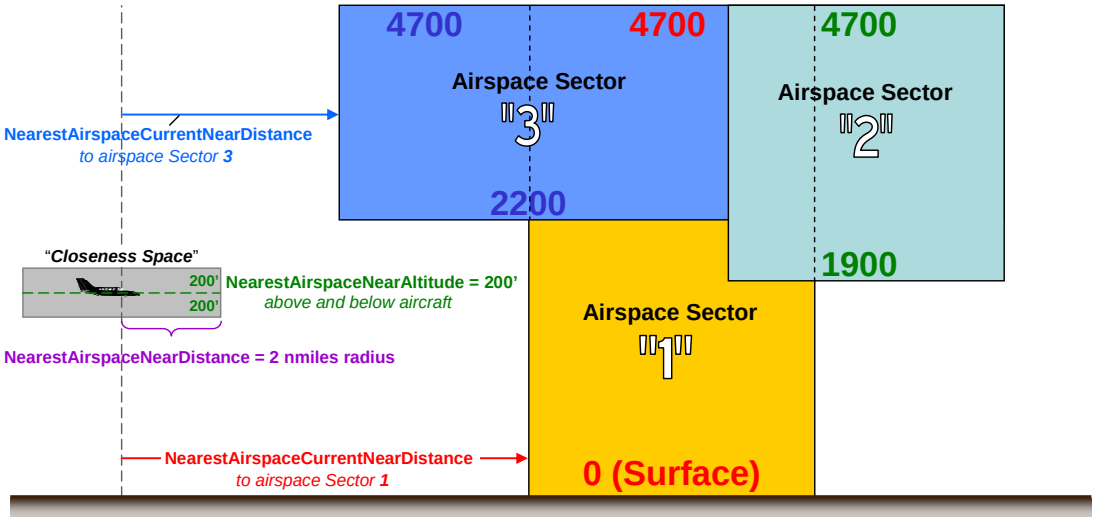
In x, y, z space, one way to think about "closeness" is to visualize as a disk extending outward from the aircraft NearDistance nmiles, whose height is two times NearAltitude feet, with the aircraft in the center. In the two dimensional diagram below, it is represented by the gray rectangle. When this "Closeness Space" (admittedly, not a proper name) touches/enters or exits an airspace sector, CurrentStatus changes. If the aircraft in the figure below is at an altitude of 1800', then as it flies through the Milwaukee airspace, CurrentStatus changes will be triggered when the "Closeness Space" touches Sector 1 and Sector 2. Regarding Sector 3, the aircraft is too low, or NearAltitude is too small, to trigger CurrentStatus values above 0.

In addition to x,y,z position, enroute time "closeness" also controls, or triggers, CurrentStatus changes.

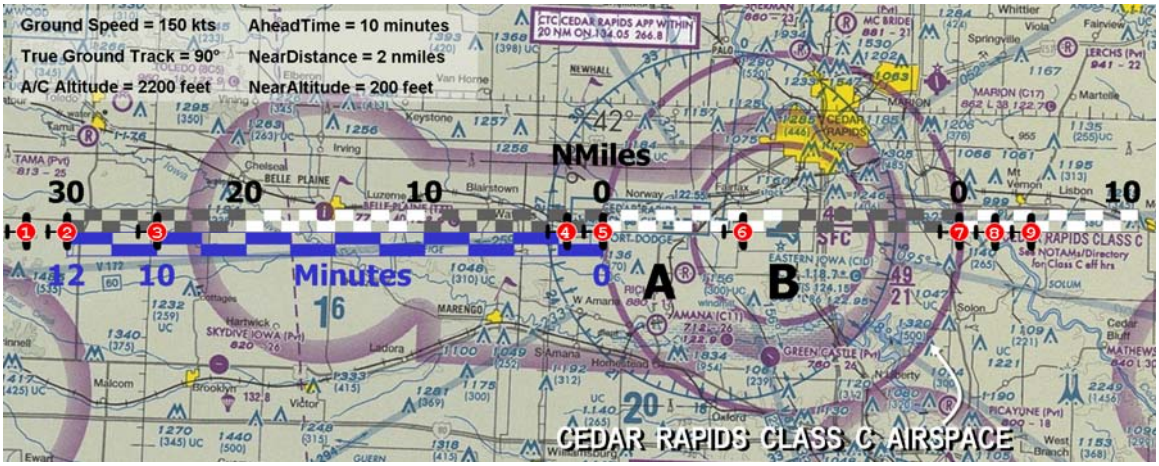
MILWAUKEE CLASS C AIRSPACE



MILWAUKEE CLASS C AIRSPACE



Enroute Time and Distance triggers can be demonstrated by tracking CurrentStatus changes in an example flight. In the figure below, an aircraft flies from Point 1 eastward through the Cedar Rapids Class C Airspace, continuing past Point 9.



CurrentStatus changes occur at various times and positions as the aircraft flies from west to east, as summarized in the table below. For the sake of simplicity, only the search results associated with Airspace Sector A (4900'/2100') are discussed:

Point	Airspace Current Status	Toward / Away from Airspace	Trigger Type	Trigger Condition for Status Change	Message Current Type	gps_500 Message
1	none	N/A	Time	CurrentAheadTime > 1.20 x AheadTime Aircraft too far from Airspace Sector A to display in Search	0 = NONE	none
2	0	Toward	Time	CurrentAheadTime ≤ 1.20 x AheadTime Aircraft appears in NearestAirspace search display	0 = NONE	none
3	2	Toward	Time	CurrentAheadTime ≤ 1.00 x AheadTime	2 = AHEAD	Airspace ahead -- less than 10 minutes
4	3	Toward	Distance	CurrrentNearDistance ≤ 1.00 x NearDistance	3 = NEAR_AHEAD	Airspace near and ahead
5	4	Inside	Distance	Entering Airspace Sector A	4 = INSIDE	Inside Airspace
6	4	Inside	Distance	Inside Airspace Sector A	4 = INSIDE	Inside Airspace
7	1	Away	Distance	Exiting Airspace Sector A CurrrentNearDistance counts upward from 0.00	1 = NEAR	none
8	0	Away	Distance	CurrrentNearDistance ≥ 1.00 x NearDistance	0 = NONE	none
9	none	N/A	Distance	CurrrentNearDistance ≥ 2.00 x NearDistance Aircraft too far from Airspace Sector A, dropped from Search	0 = NONE	none

Point 1 – The aircraft is too distant (in time) from Airspace Sector A to be displayed in the NearestAirspace search.

Point 2 – Airspace Sector A (4900'/2100') first appears in the NearestAirspace search display when CurrentAheadTime = 1.20 x AheadTime = 12 minutes. The 1.20 factor appears to be hard coded into the gps module. The initial CurrentStatus is 0.

```
NEAREST AIRSPACE SEARCH
  41.8837 :Current Lat   2198 :Cur Alt       153 :Gnd Speed     90 :Tru Gnd Trk
 -92.6176 :Current Lon      2 :Near Dist     200 :Near Alt      10 :Ahead Time
                           16 :Query (Dec)   100 :Max Dist      50 :Max Items

------------------------------------ Current ------------------------------------
Line             Name Type Status MaxAlt MinAlt NearDist AheadTime     Freq  FreqName
0         CEDAR RAPIDS    4      0   4900   2100     30.6     12.00    0.000
```

Point 3 - CurrentAheadTime = AheadTime. CurrentStatus = 2. MessageCurrentType = 2. gps_500 message = "Airspace ahead -- less than 10 minutes". CurrentStatus changes to 2 when CurrentAheadTime = AheadTime.

```
NEAREST AIRSPACE SEARCH
  41.8836 :Current Lat   2199 :Cur Alt       153 :Gnd Speed     90 :Tru Gnd Trk
 -92.5046 :Current Lon      2 :Near Dist     200 :Near Alt      10 :Ahead Time
                           16 :Query (Dec)   100 :Max Dist      50 :Max Items

------------------------------------ Current ------------------------------------
Line             Name Type Status MaxAlt MinAlt NearDist AheadTime     Freq  FreqName
0         CEDAR RAPIDS    4      0   4900   2100     25.5     10.00    0.000
1         CEDAR RAPIDS    4      0   4900      0     30.5     11.97    0.000
```

Point 4 – CurrentNearDistance = NearDistance. CurrentStatus = 3. MessageCurrentType = 3. gps_500 message = "Airspace near and ahead".

```
NEAREST AIRSPACE SEARCH
  41.8847 :Current Lat   2199 :Cur Alt       153 :Gnd Speed     90 :Tru Gnd Trk
 -91.9776 :Current Lon      2 :Near Dist     200 :Near Alt      10 :Ahead Time
                           16 :Query (Dec)   100 :Max Dist      50 :Max Items

------------------------------------ Current ------------------------------------
Line             Name Type Status MaxAlt MinAlt NearDist AheadTime     Freq  FreqName
0         CEDAR RAPIDS    4      3   4900   2100      2.0      0.77    0.000
1         CEDAR RAPIDS    4      2   4900      0      7.0      2.74    0.000
```

Point 5 – Entering Airspace Sector A. CurrentNearDistance = 0. CurrentStatus = 4. MessageCurrentType = 4. gps_500 message = "Inside Airspace".

```
NEAREST AIRSPACE SEARCH
  41.8850 :Current Lat   2199 :Cur Alt       153 :Gnd Speed     90 :Tru Gnd Trk
 -91.9302 :Current Lon      2 :Near Dist     200 :Near Alt      10 :Ahead Time
                           16 :Query (Dec)   100 :Max Dist      50 :Max Items

------------------------------------ Current ------------------------------------
Line             Name Type Status MaxAlt MinAlt NearDist AheadTime     Freq  FreqName
0         CEDAR RAPIDS    4      4   4900   2100      0.0      0.00    0.000
1         CEDAR RAPIDS    4      2   4900      0      4.8      1.90    0.000
```

Point 6 – Aircraft is inside both Sector A and B. For both sectors, CurrentNearDistance = 0. CurrentStatus = 4. MessageCurrentType = 4. gps_500 message = "Inside Airspace".

```
NEAREST AIRSPACE SEARCH
  41.8857 :Current Lat   2198 :Cur Alt       153 :Gnd Speed    89 :Tru Gnd Trk
 -91.8204 :Current Lon      2 :Near Dist     200 :Near Alt     10 :Ahead Time
                           16 :Query (Dec)   100 :Max Dist     50 :Max Items

------------------------------------ Current ------------------------------------
Line              Name Type Status MaxAlt MinAlt NearDist AheadTime     Freq  FreqName
0          CEDAR RAPIDS    4      4   4900      0      0.0      0.00   0.000
1          CEDAR RAPIDS    4      4   4900   2100      0.0      0.00   0.000
```

Point 7 – Exiting Airspace Sector A. CurrentNearDistance = 0, and begins counting upwards. CurrentStatus = 1. MessageCurrentType = 1. gps_500 message = none. Note that AheadTime equals 8464.92 minutes. At 153 knots ground speed, this equates to 21,586 nmiles, which is the circumference of the earth at 2200' asl. This is an indication the airspace sector is *behind* the aircraft.

```
NEAREST AIRSPACE SEARCH
  41.8862 :Current Lat   2199 :Cur Alt       153 :Gnd Speed    90 :Tru Gnd Trk
 -91.4829 :Current Lon      2 :Near Dist     200 :Near Alt     10 :Ahead Time
                           16 :Query (Dec)   100 :Max Dist     50 :Max Items

------------------------------------ Current ------------------------------------
Line              Name Type Status MaxAlt MinAlt NearDist AheadTime     Freq  FreqName
0          CEDAR RAPIDS    4      1   4900   2100      0.1   8464.92   0.000
```

Point 8 - CurrentNearDistance = 2.0 and still counting upwards. CurrentStatus = 0. MessageCurrentType = 0. gps_500 message = none.

```
NEAREST AIRSPACE SEARCH
  41.8861 :Current Lat   2199 :Cur Alt       153 :Gnd Speed    90 :Tru Gnd Trk
 -91.4390 :Current Lon      2 :Near Dist     200 :Near Alt     10 :Ahead Time
                           16 :Query (Dec)   100 :Max Dist     50 :Max Items

------------------------------------ Current ------------------------------------
Line              Name Type Status MaxAlt MinAlt NearDist AheadTime     Freq  FreqName
0          CEDAR RAPIDS    4      0   4900   2100      2.1   8463.80   0.000
```

Point 9 - CurrentNearDistance = 2.0 = 2.00 x NearDistance. At 2 times NearDistance, the airspace sector is dropped out of the NearestAirspace search display. The 2.00 factor appears to be hard coded into the gps module.

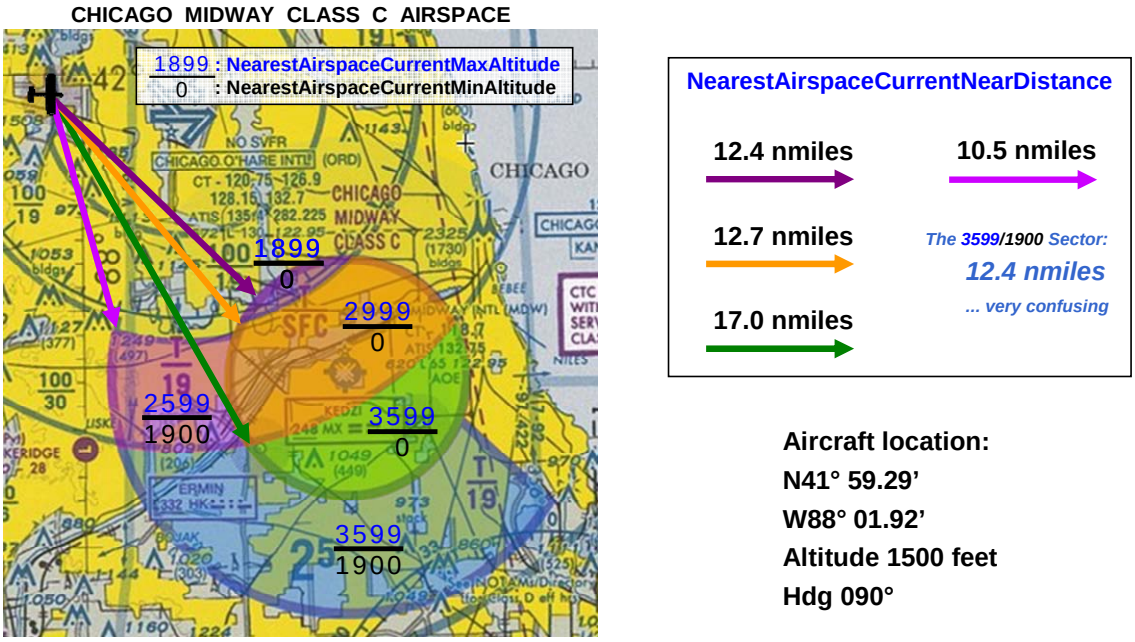
The results of a NearestAirspace search are always sorted first by CurrentStatus, then by increasing distance. All the Status = 4 airspaces are first listed in ascending distance, then the Status = 3 airspaces are listed in ascending order, and so forth.

❑ NearestAirspaceCurrentNearDistance (nmiles) [Get]

NearestAirspaceCurrentNearDistance is the ground distance between the current position of the aircraft and the closest point of each airspace sector to the aircraft (or other reference point).

The figure below demonstrates NearestAirspaceCurrentNearDistance for an aircraft near Chicago Midway International airport after a NearestAirspace search of Class "C" airspaces.

NearestAirspaceCurrentNearDistance



The NearestAirspace search returns a separate CurrentNearDistance for each airspace sector. Note the confusing (to me) result of the far airspace, 3599 / 1900. CurrentNearDistance of this sector is the *same* as the closest airspace that extends to the surface. CurrentNearDistance rules pertaining to 'hidden' or 'farthest' sectors such as this are not clear to me.

NearestAirspaceCurrentAheadTime (minutes) [Get]

NearestAirspaceCurrentAheadTime is the enroute (elapsed) time until the aircraft physically enters the airspace sector. When flying toward an airspace sector, AheadTime counts down, starting at 12 minutes when the sector first appears in NearestAirspace search. CurrentAheadTime = 0 while the aircraft is inside the sector. Leaving the airspace sector is more interesting. Upon exit, CurrentAheadTime resumes a countdown until the aircraft again enters the sector *after flying around the globe*. However, the NearestAirspace distance trigger, CurrentNearDistance, drops the sector from NearestAirspace search when CurrentNearDstance is 2X NearDistance.

MESSAGE GROUP

The Message Group variables control the content of the Airspace messages displayed by the gps_500 gauge. They do not control *when* a message is displayed – variables in the NearestAirspace Group do that. Instead, the Message Group variables determine *which* message will be displayed.

There are four different airspace messages in the gps_500 gauge, and under the right circumstances, up to three can be displayed at the same time. Consequently, the messages are indexed, and a typical display loop is used for the display.

❑ MessageItemsNumber (enum) [Get]

MessageItemsNumber is the number of messages that are active and can be seen by pressing the MSG button on the gps_500 gauge.

❑ MessageCurrentLine (enum) [Get, Set]

MessageCurrentLine is the Index pointer used in the display.

❑ MessageCurrentType (enum) [Get]

MessageCurrentType is the id number of the specific message.

1. Near Airspace – less than 2 nm
2. Airspace ahead – less than 10 minutes
3. Airspace near and ahead
4. Inside Airspace

❑ NewMessagesNumber (enum) [Get]

NewMessagesNumber indicates the number of new messages waiting to be read. It is reset to 0 when the pilot presses the MSG Button to view the messages, but can increase from 1 to 2 or 3 if the messages are ignored.

❑ NewMessagesConfirm (enum) [Set]

NewMessagesConfirm is a write-only variable that is set to 1 when the pilot presses the MSG Button after viewing the message(s).

ICAO SEARCH GROUP

ICAOSearch is a procedure that attempts to find ICAOs that contain the Ident the user enters. The ICAO is important because it is required before access to variables in the Waypoint Groups is possible. ICAOSearch data entry can be accomplished via direct keyboard entry, mouse click, or from code.

While the 12 character ICAO is unique, Idents are sometimes not, and consequently ICAOSearch may find multiple ICAOs that can be matched to facility type and Ident. Because of this, ICAOSearch results are indexed and require an Index Pointer in order to access the desired ICAO in the list returned by ICAOSearch. Often, however, only one ICAO is found that matches the input parameters and the default Index Pointer value 0 (zero) automatically takes care of selection of the proper ICAO.

			NameSearch			ICAOSearch
Facility	**Waypoint Group**	**Name Exists?**	**NAME Searchable?**	**IDENT Exists?**	**ICAO Type**	**IDENT Searchable?**
AIRPORT	AIRPORT	YES	YES	YES	A	YES
RUNWAY WAYPOINT	None	NO	NO	YES	R	NO
VOR	VOR	YES	NO	YES	V	YES
ILS / LOC	AIRPORT	YES	NO	YES	V	YES
NDB	NDB	YES	NO	YES	N or X	YES
fs9gps WAYPOINT	INTERSECTION	NO	NO	YES	W	YES
USER WAYPOINT	None	NO	NO	NO	None	NO

❑ IcaoSearchInitialIcao (string) [Get, Set]

IcaoSearchInitialIcao is the ICAO of the first Ident displayed as certain gps_500 pages containing ICAOSearch code open. It is set by the current FacilityICAO (gps_500 lines 3808 and 3813). In the event that user input updates ICAOSearch and the Ident displayed on the screen changes, but the user ultimately cancels the selections, then the current ICAO will revert back to IcaoSearchInitialIcao.

❑ IcaoSearchStartCursor (1 to 5 character string) [Set]

IcaoSearchStartCursor is used to filter, or restrict, the gps search of ICAOs to those of a certain type:

StartCursor	FACILITY
A	AIRPORT
V	VOR
N or X	NDB
W	WAYPOINT / INTERSECTION
M	Does Not Exist

Only these letters may be used in IcaoSearchStartCursor.

Any combination of the letters can also be entered. 'AVNW' will enable a search of all types of ICAOs. Searching 'W' will yield results for VORs and NDBs in addition to Waypoint/Intersections.

The gps_500 gauge suggests a StartCursor of 'M' , Marker (gps_500 lines 3813, 3814), but no searchable Facility with StartCursor 'M' exists in the fs9gps database.

❑ IcaoSearchStopCursor (enum) [Set]

I am not completely sure what this variable accomplishes. IcaoSearchStopCursor is an enum that appears only in the Enter and Clear macros in the gps_500 (`<Macro Name="ENTButton">` and `<Macro Name="CancelInput">`). The value assigned in the macro is always zero. Consequently, it appears related to the cancelation of user input ("Cursor").

Having said that, I've not yet needed StopCursor in any of the scripts written in preparation of this guidebook. As well, I've assigned different values to StopCursor and also removed it from the gps_500 gauge, all with no effect that I have been able to see.

MSFT put it there for a reason, but so far, it escapes me.

DATA ENTRY METHODS FOR ICAOSearch

There are three methods of data entry of the search filter and Ident:

1. **Keyboard Direct Entry**. The user types input information on the keyboard. IcaoSearchEnterChar will automatically invoke IcaoSearchAdvanceCursor and automatically concatenate keystroke entries, allowing for continuous typing.

```
<On Key="AlphaNumeric">
      <Visible> (L:ICAOSearchEntry, enum) 101 == </Visible>
      (M:Key) chr (>@c:IcaoSearchEnterChar)
</On>
```

2. **Mouse**. Click spots are used to mimic the use of knobs to enter the Ident, as in a Garmin GNS 500 or the FS9 gps_500 gauge. Note that in the example below, IcaoSearchAdvanceCursor and IcaoSearchAdvanceCharacter are used, but IcaoSearchEnterChar is not needed. When entering an Ident with the mouse, as the user advances the cursor, `1 (>C:fs9gps:IcaoSearchAdvanceCursor)`, the character currently selected by AdvanceCharacter is automatically entered into EnterChar. Additionally, as the cursor continues to advance, the character selected using AdvanceCharacter is concatenated with the previous characters, thus building the Ident string. The string is passed to EnterChar each time a character is selected (each time AdvanceCharacter is 'clicked').

   Garmin-type GNS knob, Upper Left click spot:

```
<Area Left="470" Top="307" Width="25" Height="12">
      <Cursor Type="Hand" />
      <Click Kind="LeftSingle" Repeat="No">
      -1 (>C:fs9gps:IcaoSearchAdvanceCursor)
      </Click>
</Area>
```

   Upper Right click spot:

```
<Area Left="500" Top="307" Width="25" Height="12">
      <Cursor Type="Hand" />
      <Click Kind="LeftSingle" Repeat="No">
      1 (>C:fs9gps:IcaoSearchAdvanceCursor)
      </Click>
</Area>
```

   Lower Left click spot:

```
<Area Left="480" Top="336" Width="15" Height="12">
      <Cursor Type="Hand" />
      <Click Kind="LeftSingle" Repeat="Yes">
      -1 (>C:fs9gps:IcaoSearchAdvanceCharacter)
      </Click>
</Area>
```

Lower Right click spot:

```
<Area Left="500" Top="336" Width="15" Height="12">
      <Cursor Type="Hand" />
      <Click Kind="LeftSingle" Repeat="Yes">
      1 (>C:fs9gps:IcaoSearchAdvanceCharacter)
      </Click>
</Area>
```

3. **Code Entry**. The users gauge code may enter data into the IcaoSearchEnterChar variable:

   ```
   'V' (>C:fs9gps:IcaoSearchStartCursor)
   ```

   Code entry of StartCursor, rather than keyboard or mouse, is always used.

   ```
   (A:NAV1 Ident, string) (>C:fs9gps:IcaoSearchEnterChar)
   ```

   or

   ```
   'SEA' (>C:fs9gps:IcaoSearchEnterChar)
   ```

   Up to 5 characters can be entered at once using Code Entry.

❑ IcaoSearchAdvanceCursor (enum) [Set]

IcaoSearchAdvanceCursor is cursor position 'incrementer' necessary when mouse entry of the Ident string is used, as if manipulating the large right knob on a Garmin GNS 500 or the FS9 stock gps-500 gauge.

AdvanceCursor value of 1 or -1, will advance the cursor one position, or back up one position. Not only does -1 cause the cursor to back up one position, so does -2, -3, etc -- they all cause the cursor to back up only one position. A zero value causes the cursor to not move, although zero is not a logical choice for any application. A value of 1 or greater causes the cursor to advance, but only one position at a time regardless of AdvanceCursor value.

❑ IcaoSearchAdvanceCharacter (enum) [Set]

An alphabet advance or backup 'incrementer'. Either 1 or -1 is used for input. The value -1 (or any other negative value) causes the letter at the current cursor position to back up one letter of the alphabet at a time. An AdvanceCharacter value of zero causes no progress or back up in the alphabet, but, of course, would be an illogical choice for applications. A value of 1 (or any positive integer) causes an advance of one letter in the alphabet.

For Airports, the alphabet values are alphanumeric characters:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ 0123456789
```

The 'space' character is between Z and 0 (zero). Advancing one character from '9' yields 'A'. Backing up one character from '0' (zero) yields the space character. Backup one more yields 'Z'.

For other Facilities, the alphabet values are ascii characters:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!"#$%&'()*+,./:;<=>?@[]^_`{|}
~space
```

The last character is the space character. These are shown in the order they are processed by AdvanceCharacter.

Summarizing,

```
1 (>c:fs9gps:IcaoSearchAdvanceCharacter)
```

causes the next letter to be selected, and

```
-1 (>c:fs9gps:IcaoSearchAdvanceCharacter)
```

causes the previous letter to be selected.

❑ IcaoSearchEnterChar (string) [Set]

Similar with AdvanceCharacter and AdvanceCursor, IcaoSearchEnterChar is used to enter the Ident string that the gps module will search for in its database. EnterChar automatically concatenates the Ident character entry to form CurrentIdent. As an example using keyboard direct entry, the user types A, B, C, D, E, F, and then G, and gets the following results:

Keyboard Entry Sequence	EnterChar	CurrentIdent	String Length
1 - 'A'	A	A	1
2 - 'B'	B	AB	2
3 - 'C'	C	ABC	3
4 - 'D	D	ABCD	4
5 - 'E'	E	ABCDE	5
6 - 'F'	F	ABCDF	5
7 - 'G'	G	ABCDG	5

	Cursor Advance	Concatenation
Keyboard Direct Entry	Automatic	Automatic
Mouse Entry	by Mouse	Automatic
Code Entry	Not Necessary	Not Necessary

There is a 5 character limit to Ident entry because Idents have a maximum string length of 5 characters. If more than 5 characters are entered, then the letter in cursor position 4 (the 5th character - the first cursor position is numbered zero) is replaced with the excess character, as above.

If EnterChar is used in association with direct keyboard entry, then the <On Key> parameters **AlphaNumeric** or **Ascii** determine which characters may be entered.

Using **AlphaNumeric**, only characters

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890
```

may be entered. **AlphaNumeric** is a good choice for Idents of an ICAOSearch.

Ascii is similar to **AlphaNumeric** except that characters "**+**", "**-**", "**,**", and "**.**" (plus, minus, comma and period/decimal point) are also accepted. **Ascii** is the choice when entering numbers such as Latitude and Longitude because of the minus sign and decimal point. Note that the gps_500 gauge uses **Ascii** for NameSearch entry (see line 3947).

❑ IcaoSearchBackupChar (enum) [Set]

IcaoSearchBackupChar is used for backspace when entering Ident via keyboard direct entry.

For example, from gps_500 lines 3951 – 3955:

```
<On Key="Backspace">
    <Visible>(C:fs9gps:enteringInput)</Visible>
      10 19 (C:fs9gps:enteringInput) rng 31
      (C:fs9gps:enteringInput) == ||
            if{
                  -1 (>C:fs9gps:IcaoSearchBackupChar) @InitBlinker quit
            }
</On>
```

Another, simpler example:

```
<On Key="Backspace">
    <Visible>(L:AsciiEntryEnable, bool)</Visible>
    (M:Key) (>C:fs9gps:IcaoSearchBackupChar)
</On>
```

Neither (M:Key) nor a number actually need to be entered in order to create a backspace. All of the following create a single backspace at a time as the keyboard backspace key is entered:

```
<On Key="Backspace">
    <Visible>(L:AsciiEntryEnable, bool)</Visible>
    (M:Key) -1 (>C:fs9gps:IcaoSearchBackupChar)
</On>

<On Key="Backspace">
    <Visible>(L:AsciiEntryEnable, bool)</Visible>
    (M:Key) 1 (>C:fs9gps:IcaoSearchBackupChar)
</On>

<On Key="Backspace">
    <Visible>(L:AsciiEntryEnable, bool)</Visible>
    (M:Key) 0 (>C:fs9gps:IcaoSearchBackupChar)
</On>

<On Key="Backspace">
    <Visible>(L:AsciiEntryEnable, bool)</Visible>
    (M:Key) -200 (>C:fs9gps:IcaoSearchBackupChar)
</On>


<On Key="Backspace">
    <Visible>(L:AsciiEntryEnable, bool)</Visible>
    (>C:fs9gps:IcaoSearchBackupChar)
</On>
```

Note the use of <Visible> tags. In the examples above, only when `L:AsciiEntryEnable = 1` will keyboard direct entry for the Ident work. The ability to "turn off" keyboard direct entry is necessary in order to retain normal FS9/X keyboard entry instructions such as "G" = Landing Gear Toggle when Ident entry is not needed.

❑ IcaoSearchCursorPosition (enum) [Get]

IcaoSearchCursorPosition is the current cursor position of the Ident entry. The first character entered is CursorPosition 0. The second is CursorPosition 1, and so forth.

CursorPosition is active for all types of Ident entry – keyboard, mouse, code.

❑ IcaoSearchCurrentIdent (string) [Get]

IcaoSearchCurrentIdent is the facility Ident constructed (concatenated) as the user enters Ident characters. It grows (SLEN increases by one) with each keystroke or mouse AdvanceCursor entry.

❑ IcaoSearchCurrentIcao (string) [Get, Set]

IcaoSearchCurrentIcao is the ICAO formed using StartCursor and CurrentIdent. Because Idents are not unique, there may be multiple ICAOs that can be matched to StartCursor and CurrentIdent. Consequently, CurrentIcao is an indexed variable (a list) that requires an Index Pointer (IcaoSearchMatchedIcao) to access.

With each Ident character entry, the gps module searches the database for ICAOs containing StartCursor and CurrentIdent and if there are any, the number of matches is stored into IcaoSearchMatchedIcaosNumber.

❑ IcaoSearchCurrentIcaoType (string) [Get]

IcaoSearchCurrentIcaoType is the facility type and is the same as StartCursor:

IcaoType	Facility
A	Airport
V	VOR, ILS, LOC
N	NDB
W	Waypoint, Intersection

❑ IcaoSearchCurrentIcaoRegion (string, SLEN=2) [Get]

The two character Region code.

❑ IcaoSearchMatchedIcaosNumber (enum) [Get]

IcaoSearchMatchedIcaosNumber is the number of ICAOs that were matched to StartCursor and CurrentIdent during ICAOSearch.

❑ IcaoSearchMatchedIcao (enum) [Get, Set]

The Index Pointer used to access specific ICAOs returned by ICAOSearch. The default is zero.

RESOLVING MULTIPLE ICAO MATCHES

An example of one method that can be used to resolve multiple ICAOSearch matches is discussed in the **ICAO SEARCH EXAMPLE** chapter (starting on page 34).

NAME SEARCH GROUP

NameSearch is a procedure that allows the user to retrieve the ICAO by entering facility Name. It is limited to Airport Name search only. The ICAO is important because it is required before access to variables in the Waypoint Airport Group is possible. NameSearch data entry can be accomplished via direct keyboard entry, mouse click, or from code.

		NameSearch			ICAOSearch
Facility	Waypoint Group	Name Exists?	NAME Searchable?	IDENT Exists?	ICAO Type	IDENT Searchable?
AIRPORT	AIRPORT	YES	YES	YES	A	YES
RUNWAY WAYPOINT	None	NO	NO	YES	R	NO
VOR	VOR	YES	NO	YES	V	YES
ILS / LOC	AIRPORT	YES	NO	YES	V	YES
NDB	NDB	YES	NO	YES	N or X	YES
fs9gps WAYPOINT	INTERSECTION	NO	NO	YES	W	YES
USER WAYPOINT	None	NO	NO	NO	None	NO

❑ NameSearchInitialIcao (string) [Get, Set]

NameSearchInitialIcao is the ICAO of the first Name displayed as certain gps_500 pages containing NameSearch code open. It is set by the current FacilityICAO (gps_500 lines 3809 and 3814). In the event that user input updates NameSearch and the Name displayed on the screen changes, but the user ultimately cancels the selections, then the current ICAO will revert back to NameSearchInitialIcao.

❑ NameSearchInitialName (string) [Get, Set]

NameSearchInitialName is the Airport Name associated with NameSearchInitialIcao.

❑ NameSearchStartCursor (string) [Set]

NameSearchStartCursor is included in the gps_500 gauge (line 3814) in parallel with NameSearch input. Unlike NameSearchStartCursor, however, it appears to be non-functional and not needed.

❑ NameSearchStopCursor (enum) [Set]

I am not sure what this variable does. NameSearchStopCursor is an enum that is used only in the Enter and Clear macros in the gps_500 (<Macro Name="ENTButton"> and <Macro Name="CancelInput">). In the macros, the value assigned to each is

always zero. The value assigned in the macro is always zero. Consequently, it appears related to the cancelation of user input ("Cursor").

Having said that, I've not needed StopCursor, not yet anyway, in any of the scripts written in preparation of this guidebook. As well, I've assigned different values to StopCursor and also removed it from the gps_500 gauge, all with no effect that I have have been able to see.

MSFT put it there for a reason, but so far, it escapes me.

❑ NameSearchAdvanceCursor (enum) [Set]

NameSearchAdvanceCursor is cursor position 'incrementer' necessary when mouse entry of the Ident string is used, as if manipulating the large right knob on a Garmin GNS 500 or the FS9 stock gps-500 gauge.

AdvanceCursor value of 1 or -1, will advance the cursor one position, or back up one position. Not only does -1 cause the cursor to back up one position, so does -2, -3, etc -- they all cause the cursor to back up only one position. A zero value causes the cursor to not move, although zero is not a logical choice for any application. A value of 1 or greater causes the cursor to advance, but only one position at a time regardless of AdvanceCursor value.

❑ NameSearchAdvanceCharacter (enum) [Set]

An alphabet advance or backup 'incrementer'. Either 1 or -1 is used for input. The value -1 (or any other negative value) causes the letter at the current cursor position to back up one letter of the alphabet at a time. An AdvanceCharacter value of zero causes no progress or back up in the alphabet, but, of course, would be an illogical choice for applications. A value of 1 (or any positive integer) causes an advance of one letter in the alphabet.

For Airports, the alphabet values are alphanumeric characters:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ 0123456789
```

The 'space' character is between Z and 0 (zero). Advancing one character from '9' yields 'A'. Backing up one character from '0' (zero) yields the space character. Backup one more yields 'Z'.

Summarizing,

```
1 (>c:fs9gps:NameSearchAdvanceCharacter)
```

causes the next letter to be selected, and

```
-1 (>c:fs9gps:NameSearchAdvanceCharacter)
```

causes the previous letter to be selected.

❑ NameSearchEnterChar (string) [Set]

NameSearchEnterChar is used to enter the Name string that the gps module will search for in its database.

There are three common methods of data entry/input of the search filter and Name:

❑ **Keyboard Direct Entry**. The user types input information on the keyboard. EnterChar will automatically invoke NameSearchAdvanceCursor and automatically concatenate keystroke entries, allowing for continuous typing.

```
<On Key="AlphaNumeric">
<Visible> (L:NameSearchEntry, enum) 101 == </Visible>
(M:Key) chr (>@c:NameSearchEnterChar)
</On>
```

❑ **Mouse.** Click spots are used to mimic the use of knobs to enter the Name, as in a Garmin GNS 500 or the FS9 gps_500 gauge. Note that in the example below, NameSearchAdvanceCursor and NameSearchAdvanceCharacter are used, but NameSearchEnterChar is not needed. When entering a Name with the mouse, as the user advances the cursor,

```
1 (>C:fs9gps:NameSearchAdvanceCursor),
```

the character currently selected by AdvanceCharacter is automatically entered into EnterChar. Additionally, as the cursor continues to advance, the character selected using AdvanceCharacter is concatenated with the previous characters, thus building the Name string. The string is passed to EnterChar each time a character is selected (each time AdvanceCharacter is 'clicked').

Garmin-type GNS knob, Upper Left click spot:

```
<Area Left="470" Top="307" Width="25" Height="12">
   <Cursor Type="Hand" />
      <Click Kind="LeftSingle" Repeat="No">
         -1 (>C:fs9gps:NameSearchAdvanceCursor)
      </Click>
</Area>
```

Upper Right click spot:

```
<Area Left="500" Top="307" Width="25" Height="12">
   <Cursor Type="Hand" />
      <Click Kind="LeftSingle" Repeat="No">
         1 (>C:fs9gps:NameSearchAdvanceCursor)
      </Click>
</Area>
```

Lower Left click spot:

```
<Area Left="480" Top="336" Width="15" Height="12">
   <Cursor Type="Hand" />
      <Click Kind="LeftSingle" Repeat="Yes">
         -1 (>C:fs9gps:NameSearchAdvanceCharacter)
      </Click>
</Area>
```

Lower Right click spot:

```
<Area Left="500" Top="336" Width="15" Height="12">
   <Cursor Type="Hand" />
      <Click Kind="LeftSingle" Repeat="Yes">
         1 (>C:fs9gps:NameSearchAdvanceCharacter)
      </Click>
</Area>
```

- **Code Entry**. Code may be used to enter the Name:

`'Garde' (>C:fs9gps:NameSearchEnterChar)`

returns the ICAO A _ _ _ _ _ _ EDOC' (Gardelegen Airport, Gardelegen, Germany).

While,
`'Garder' (>C:fs9gps:NameSearchEnterChar)`, or
`'Garderm' (>C:fs9gps:NameSearchEnterChar)`, or
`'Gardermo' (>C:fs9gps:NameSearchEnterChar)`, or
`'Gardermoe' (>C:fs9gps:NameSearchEnterChar)`, or
`'Gardermoen' (>C:fs9gps:NameSearchEnterChar)`

all return the ICAO A _ _ _ _ _ _ ENGM' (Gardermoen Airport, Oslo, Norway)

EnterChar and AdvanceCursor/AdvanceCharacter automatically concatenates the character entry to form CurrentName. Up to 80 characters can be entered into CurrentName even though Airport Names are not nearly that long.

	Cursor Advance	Concatenation
Keyboard Direct Entry	Automatic	Automatic
Mouse Entry	by Mouse	Automatic
Code Entry	Not Necessary	Not Necessary

❑ NameSearchBackupChar (enum) [Set]

NameSearchBackupChar is used for backspace when entering a Name via keyboard direct entry.

For example, gps_500 lines 3951 – 3955:

```
<On Key="Backspace">
    <Visible>(C:fs9gps:enteringInput)</Visible>
      110 119 (C:fs9gps:enteringInput) rng 131
      (C:fs9gps:enteringInput) == ||
            if{
                  -1 (>C:fs9gps:NameSearchBackupChar) @InitBlinker quit
            }
</On>
```

Another, simpler example:

```
<On Key="Backspace">
    <Visible>(L:AlphanumericEntryEnable, bool)</Visible>
    (M:Key) (>C:fs9gps:NameSearchBackupChar)
</On>
```

Neither (M:Key) nor a number actually need to be entered in order to create a backspace. All of the following create a single backspace at a time as the keyboard backspace key is entered:

```
<On Key="Backspace">
    <Visible>(L:AlphanumericEntryEnable, bool)</Visible>
    (M:Key) -1 (>C:fs9gps:NameSearchBackupChar)
</On>
```

```
<On Key="Backspace">
    <Visible>(L:AlphanumericEntryEnable, bool)</Visible>
    (M:Key) 1 (>C:fs9gps:NameSearchBackupChar)
</On>

<On Key="Backspace">
    <Visible>(L:AlphanumericEntryEnable, bool)</Visible>
    (M:Key) 0 (>C:fs9gps:NameSearchBackupChar)
</On>

<On Key="Backspace">
    <Visible>(L:AlphanumericEntryEnable, bool)</Visible>
    (M:Key) -200 (>C:fs9gps:NameSearchBackupChar)
</On>

<On Key="Backspace">
    <Visible>(L:AlphanumericEntryEnable, bool)</Visible>
    (>C:fs9gps:NameSearchBackupChar)
</On>
```

Note the use of `<Visible>` tags. In the examples above, only when `L:AlphanumericEntryEnable` = 1 will keyboard direct entry for the Name work. The ability to "turn off" keyboard direct entry is necessary in order to retain normal FS9/X keyboard entry instructions such as "G" = Landing Gear Toggle when Name entry is not needed.

❑ NameSearchCursorPosition (enum) [Get]

NameSearchCursorPosition is the current cursor position of the Name entry. The first character entered is CursorPosition 0. The second is CursorPosition 1, and so forth.

CursorPosition is active for all types of Name entry – keyboard, mouse, code.

❑ NameSearchCurrentName (string) [Get, Set]

NameSearchCurrentIdent is the airport Name constructed (concatenated) as the user enters Name characters. It grows (SLEN increases by one) with each keystroke or mouse AdvanceCursor entry.

❑ NameSearchCurrentMatch (enum) [Get]

NameSearchCurrentMatch is the number of ICAOs that were matched to the Airport Name during the NameSearch. It is always either 0 or 1.

❑ NameSearchCurrentIcao (string) [Get]

NameSearchCurrentIcao is the ICAO associated with the current Airport Name. As an Airport Name is being entered via direct keyboard entry or mouse one character at a time, an Airport Name is progressively 'concatenated' and a NameSearch performed as each new character is entered.

❑ NameSearchCurrentIcaoType (string) [Get]

NameSearchCurrentIcaoType is always 'A', for Airport.

❑ NameSearchCurrentIcaoRegion (string) [Get]

NameSearchCurrentIcaoRegion is always a blank string because Airport ICAOs do not contain a Region Code.