ICAO SEARCH EXAMPLE

The following demonstrates the ICAOSearch process of retrieving a facility ICAO given the StartCursor search filter and facility Ident. It's a situation that could arise if the user needs the latitude and longitude of a particular VOR when the Ident is known, something that cannot be done simply with A:Vars. A solution would be to 1) perform ICAOSearch to retrieve the VOR's ICAO, then, 2) transfer the ICAO to the WaypointVor or Facility Group to gain access the VOR's Lat and Lon.

As an example, "What's the Lat and Lon of the Corvallis, Oregon, USA VOR (Ident = "**CVO**")?"

STEP 1 - ICAOSearch. ICAOSearch has two required inputs, the search filter IcaoSearchStartCursor, and the facility Ident which is entered using the variable IcaoSearchEnterChar.

Before entry begins, all strings and enums in the ICAOSearch Group are blank and zero:

1.1 – First, enter the ICAOSearch filter, IcaoSearchStartCursor. ICAOSearch always begins by entering IcaoSearchStartCursor. In this example, it is '**V**' for VOR. A keyboard direct entry statement would be in the form of:

```
<On Key="AlphaNumeric">
      <Visible>(L: AlphaNumericEntryEnable, enum) 101 ==</Visible>
      (M:Key) chr (>C:fs9gps:IcaoSearchStartCursor)
</On>
```

An equivalent statement:

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

After the ICAO search filter has been entered, ICAOSearch immediately returns the first ICAO that matches the criterion. In this case, CurrentIcaoType is "**V**", and the first VOR Ident in the database is **1CD** (CurrentIdent). The associated ICAO is **VCY**_ _ _ _**1CD**_ _. The Region is **CY**, and is part of the 12 character ICAO. There is only one VOR with the Ident **1CD** in the fs9gps database, hence MatchedIcaosNumber = 1.

```
MatchedIcaosNumber: 1

          ICAO      111
Index     123456789012
0         VCY     1CD
```

At this point, only StartCursor has been entered. No Ident search string, or portion of the Ident, has been entered yet.

1.2 – Next, begin entering the Ident using IcaoSearchEnterChar. "**C**" is entered.

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

IcaoSearchEnterChar is the heart of the ICAOSearch. It is how Ident is entered. The ICAOSearch engine subsequently searches the database and returns an ICAO that matches the ICAO Type defined by StartCursor and the Ident defined by EnterChar.

After "**C**" is entered, the following automatically occurs:

- The CursorPosition advances one place to Position 1, ready for the next character of the Ident to be entered.
- The first VOR Ident in the fs9gps database that begins with "**C**" is the **CA** VOR. Its ICAO is **V**_ _**SOCACA**_ _ _. From this ICAO one can tell that it is an ILS or LOC because Region is blank and the owning airport, **SOCA**, is listed in character positions 4 through 7. MatchedIcaosNumber is 1, meaning that there is only one VOR in the database with Ident = **'CA'**.

```
MatchedIcaosNumber: 1

          ICAO      111
Index     123456789012
0         V   SOCACA
```

'V SOCACA' is the ILS/DME 08 at Rochambeau Airport, Cayenne, French Guiana.

1.3 – IcaoSearchEnterChar. "**V**" is entered.

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

After "**V**" was entered, the following automatically occurs:

- "**C**", previously entered, and "**V**" are concatenated to form "**CV**".

- The CursorPosition advances one place to Position 2, ready for the next character of the Ident to be entered.
- ICAOSearch returns three VORs whose Ident is **'CV':**

```
MatchedIcaosNumber: 3

          ICAO      111
Index     123456789012
0         V   EGDCCV
1         V   LFKCCV
2         VYB     CV
```

1.4 – IcaoSearchEnterChar. Lastly, "**O**" is entered.

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

After "**O**" is entered, the following automatically occurs:

- "**CV**" and "**O**" are concatenated to form "**CVO**". Entry of the "**CVO**" Ident is now complete. Even though keyboard direct entry always is a "one letter at a time" entry process, use of a shift register in the <On> statement is not necessary. The gps module automatically concatenates, thereby enabling continuous typing.
- The CursorPosition advances one place to Position 3.
- ICAOSearch returns two VORs whose Ident is **'CVO':**

```
MatchedIcaosNumber: 2

          ICAO      111
Index     123456789012
0         VHE     CVO
1         VK1     CVO
```

VHE_ _ _ _**CVO**_ _ is the ICAO of the CAIRO VOR located in Cairo, Egypt, not CORVALLIS VOR located in Corvallis, Oregon, USA. Both share the same Ident, "**CVO**". ICAOSearch located the two VORs with that Ident, but **VHE**_ _ _ _**CVO**_ _ comes alphabetically before **VK1**_ _ _ _**CVO**_ _, the ICAO of the Corvallis VOR, so it has Index value 0. If an ICAO transfer is made at this point, the Lat and Lon of the Cairo VOR will be accessed.

ICAOs returned from ICAOSearch are Indexed, and a pointer, IcaoSearchMatchedIcao, must be defined to access the ICAOs. The default index pointer is always 0, the first item in the list. Setting IcaoSearchMatchedIcao to 1 accesses the second VOR:

```
1 (>C:fs9gps:IcaoSearchMatchedIcao)
```

```
GPS Viewer 1.2
GET  SET  SLEN  1234567890123456789012  STRING  NUMBER
 G         2    3                       IcaoSearchCursorPosition, enum
 G         3    CVO                     IcaoSearchCurrentIdent, string
 G    S    12   VK1      CVO            IcaoSearchCurrentIcao, string
 G         1    V                       IcaoSearchCurrentIcaoType, string
 G         2    K1                      IcaoSearchCurrentIcaoRegion, string
 G         2    2                       IcaoSearchMatchedIcaosNumber, enum
 G    S    2    1                       IcaoSearchMatchedIcao, enum
```

STEP 2 - Transfer the ICAO to the WaypointVor Group. After setting the proper Index pointer, ICAO Transfer into the WaypointVor Group can be performed. The appropriate xml:

```
(C:fs9gps:IcaoSearchCurrentIcao) (>C:fs9gps:WaypointVorIcao)
```

```
GPS Viewer 1.2
GET  SET  SLEN  1234567890123456789012  STRING  NUMBER
 G    S    12   VK1      CVO            WaypointVorICAO, string
 G         3    CVO                     WaypointVorIdent, string
 G         9    CORVALLIS               WaypointVorName, string
 G         4      44.49956 95           WaypointVorLatitude, degrees
 G         7    -123.29366 95           WaypointVorLongitude, degrees
 G         2    3                       IcaoSearchCursorPosition, enum
 G         3    CVO                     IcaoSearchCurrentIdent, string
 G    S    12   VK1      CVO            IcaoSearchCurrentIcao, string
 G         1    V                       IcaoSearchCurrentIcaoType, string
 G         2    K1                      IcaoSearchCurrentIcaoRegion, string
 G         2    2                       IcaoSearchMatchedIcaosNumber, enum
 G    S    2    1                       IcaoSearchMatchedIcao, enum
```

Now, WaypointVorIcao = IcaoSearchCurrentIcao = **VK1**_ _ _ _**CVO**_ _, and variables WaypointVorLatitude and WaypointVorLongitude return the desired Lat and Lon of the CORVALLIS VOR.

RESOLVING MULTIPLE ICAO MATCHES

Because Idents are not unique, ICAO Searches often return multiple ICAO matches (IcaoSearchMatchedIcaosNumber > 1), as in the case of the 'CVO' VOR search.

There are a few ways to select the index pointer of the desired facility. The gps_500 gauge typically displays ICAOSearch results (or the Idents from) on the screen and the user selects the desired facility by manipulating a scroll bar and cursor.

Alternatively, code in the users gauge can determine the correct index pointer based on selection criteria. In the VOR example, the nearest VOR may be desired. The following code snippet demonstrates one way to select the nearest VOR. It involves an ICAO Transfer into the Facility Group to retrieve the Lat and Lon of each VOR returned by the ICAO search.

```
 9 <!-- LOOP THROUGH ICAO SEARCH RESULT TO FIND NEAREST FACILITY -->
10 (@c:IcaoSearchMatchedIcaosNumber, enum) 0 &gt; (L:ResolveMultipleICAOs, bool) 1 == and
11     if{
12         (L:ResolveMultipleICAOInit, bool) 1 ==
13             if{
14                 (A:PLANE LATITUDE, degrees) (>@c:GeoCalcLatitude1, degrees)
15                 (A:PLANE LONGITUDE, degrees) (>@c:GeoCalcLongitude1, degrees)
16                 99999 (>L:FacilityNearestDistance, nmiles)
17                 0 (>L:ICAOTransferCycleCounter, enum)
18                 3 (>L:NumberOfCyclesToSkip, enum)
19                 0 (>L:NearestFacilityIndex, enum)
20                 0 (>L:NumberOfFacilitiesChecked, enum)
21                 0 (>L:ResolveMultipleICAOInit, bool)
22             }
23
24         (L:NumberOfFacilitiesChecked, enum) (>@c:IcaoSearchMatchedIcao, enum)
25         (@c:IcaoSearchCurrentIcao) (>@c:FacilityICAO)
26
27         (L:ICAOTransferCycleCounter, enum) ++ (>L:ICAOTransferCycleCounter, enum)
28         (L:ICAOTransferCycleCounter, enum) (L:NumberOfCyclesToSkip, enum) &gt;
29             if{
30                 (@c:FacilityLatitude, degrees) (>@c:GeoCalcLatitude2, degrees)
31                 (@c:FacilityLongitude, degrees) (>@c:GeoCalcLongitude2, degrees)
32                 (@c:GeoCalcDistance, nmiles) (>L:FacilityCurrentDist, nmiles)
33                 (L:FacilityNearestDistance, nmiles) (L:FacilityCurrentDist, nmiles) &gt;
34                     if{
35                         (L:FacilityCurrentDist, nmiles) (>L:FacilityNearestDistance, nmiles)
36                         (@c:IcaoSearchMatchedIcao) (>L:NearestFacilityIndex, enum)
37                     }
38                 (L:NumberOfFacilitiesChecked, enum) ++ (>L:NumberOfFacilitiesChecked, enum)
39                 0 (>L:ICAOTransferCycleCounter, enum)
40                 (@c:IcaoSearchMatchedIcao, enum) 1 + (@c:IcaoSearchMatchedIcaosNumber, enum) &gt;=
41                     if{
42                         0 (>L:ResolveMultipleICAOs, bool)
43                     }
44             }
45     }
```

- **Line 10.** The Loop is executed if there are multiple ICAOs returned by ICAOSearch and `L:ResolveMultipleICAOs` mouse Area has been clicked.
- **Lines 12 through 22.** Loop parameters are initialized.
- **Line 24.** The Index Pointer is set.
- **Line 25.** The ICAO Transfer.
- **Lines 27 and 28.** The cycle skip code used after the ICAO Transfer. This is a cycle counting technique.
- **Lines 30 through 32.** GeoCalcDistance is calculated using the aircraft position as the Lat1, Lon1 reference point and the current Facility location as Lat2, Lon2. The result is stored as an L:Var.
- **Lines 33 through 37.** The current Facility distance is checked to see if it is the shortest and, if so, then its Index Pointer is stored as `L:NearestFacilityIndex`, which is the objective.
- **Lines 38 and 39.** The cycle counter is reset to zero and the Index Pointer is incremented to prepare for the next Facility in the ICAOSearch list.

- **Lines 40 through 43.** The loop terminates after the number of Facilities checked equals MatchedIcaosNumber.

The Loop is triggered by a click area that sets both `L:ResolveMultipleICAOs` and `L:ResolveMultipleICAOInit` to 1.

ICAO SEARCH AIRPORTS – A Special Case

If interested in Airports only, then ICAO Search is not necessary in order to find the unique Airport ICAO. The reason is that the 3 to 4 character Airport Ident is unique itself, and a full Airport ICAO involves simply concatenating the Airport Start Cursor, "A", with the Ident, as follows:

```
'A______' 'KLAX' scat
```

That is the full ICAO for KLAX Airport. The first part of the statement is an 'A' followed by six spaces. The Airport Ident could be entered via Direct Keyboard Entry or Mouse or Code (discussed in later chapters).

Defining the ICAO using Direct Keyboard Entry for the Ident might look something like the following:

```
'A______'
(L:IdentChar1, enum) chr scat
(L:IdentChar2, enum) chr scat
(L:IdentChar3, enum) chr scat
(L:IdentChar4, enum) chr scat
```

Avoiding the ICAO Search using this shortcut is a special case that is safe only for Airports.

ICAO TRANSFER

The ICAO Transfer is a simple technique used to move from one gps group into another in order to access additional information (i.e., variables) regarding a specific Airport, VOR, NDB, or Intersection / Waypoint - information that is contained in the second group, but not in the first.

It is an important technique to understand as far as FS9 goes. Although useful while working with the FSX gps module, some of the need for the ICAO Transfer has been alleviated because, with FSX, the Nearest Groups (the 'first' group) are populated with many more variables from the Waypoint Groups (the 'second' group) to begin with. Makes life much easier.

An example is the best way to explain the technique.

ICAO TRANSFER EXAMPLE – NearestAirport > WaypointAirport

A good example of ICAO Transfer in fs9gps is the access of additional airport information following a NearestAirport search. Suppose the user wants a list of all frequencies from the nearest airport. The solution begins with a NearestAirport search, the results of which are shown below. In this example, all of the variables that are available in the NearestAirport Group are displayed.

```
NEAREST AIRPORT SEARCH
  41.8364 :Current Lat    10 :Max Items   10 :Items Num
 -88.2098 :Current Lon    75 :Max Distance

Current  ICAO     111 ---------------------- Current -------------------------
Line     123456789012 Ident Kind Rwy*  Dist  Brg Best Appr  Com     Freq Length
0        A      KDPA   KDPA    1   14   4.6  338   13   ILS  twr  120.90   7573
1        A      LL10   LL10    1  179   6.1  177    0                0.00   2580
2        A      1C5     1C5    1  178   9.2  157    8   GPS  CTF  122.90   3363
3        A      85LL   85LL    2  359   9.4  194    0                0.00   2300
4        A      06C     06C    1  109  10.4   28    0        CTF  123.00   3800
5        A      IS23   IS23    3  179  11.0  341    0                0.00   3000
6        A      LL22   LL22    1   89  11.3  123    0                0.00   2800
7        A      KARR   KARR    1   89  12.5  252   13   ILS  twr  120.60   6491
8        A      2IS0   2IS0    1  179  13.4  142    0                0.00   2900
9        A      LL51   LL51    2  359  14.2  192    0                0.00   2000
```

FS9 NearestAirport Group information available from a NearestAirport search is:

- 12 Character ICAO Identification
- Airport Ident
- Airport Kind (Class) Code (hard surface = 1, soft = 2, water = 3, etc)
- Longest Runway Direction
- Distance to the Airport from the reference point (aircraft)
- True Bearing to the Airport from the reference point
- Best (most precise) Approach Code

- Best (most precise) Approach Name
- Com Frequency Name (but just one, the principal airport control, usually 'twr' or 'CTF' if there is either, but never Ground or other)
- Com Frequency Value (but just one, and if there are multiple Tower frequencies, only the first)
- Longest Runway Length

However, in FS9, there is much more airport information, such as all airport frequencies, located in the WaypointAirport Group. Unfortunately, with FS9, the WaypointAirport Group variables are not accessible simply by performing a NearestAirport search.

To display the list of all frequencies of the nearest airport, the user must 'transfer' into the WaypointAirport Group where they are located. This is a simple process of sending the ICAO of the nearest airport obtained from the NearestAirport search to WaypointAirport. The xml instruction, which should be inserted in the <Update> section is:

```
0 (>C:fs9gps:NearestAirportCurrentLine) // Index pointer for the nearest

(C:fs9gps:NearestAirportCurrentICAO)
(>C:fs9gps:WaypointAirportICAO)
```

Now, WaypointAirport has a specific ICAO to work with, and all variables subsequently accessed in the WaypointAirport Group, such as the list of frequencies, return information about only that airport.

```
WAYPOINT AIRPORT FREQUENCIES
ICAO     111    12 :Frequencies Num
123456789012
A      KDPA  :Waypoint Airport ICAO

--- Waypoint Airport Frequency ----
Freq        Name  Limit  Value Type
0       Approach      0   133.50    1
1           ATIS      1   124.80    1
2      Clearance      0   119.75    1
3      Departure      0   133.50    1
4            FSS      0   122.10    1
5            FSS      0   122.30    1
6         Ground      0   121.80    1
7          Tower      0   120.90    1
8          Tower      0   124.50    1
9         Unicom      0   122.95    1
10       ILS 02L      0   111.70    2
11        ILS 10      0   109.50    2
```

A few comments:

- In this example, why doesn't one go directly to WaypointAirport for the frequency list to begin with? Because WaypointAirport does not have the ability to determine which airport is the nearest. WaypointAirport must always be told (e.g., by defining WaypointAirportICAO) which specific airport you are interested

in. This is one area in which FSX gps is easier to work with. Much airport information is available in the NearestAirport Group to begin with, unlike in FS9.

- The ICAO must be used for the transfer into another group. It is the unique database element identifier. Note that the NearestAirport to WaypointAirport transfer will not work using Ident:

```
0 (>C:fs9gps:NearestAirportCurrentLine)

(C:fs9gps:NearestAirportCurrentIdent)
(>C:fs9gps:WaypointAirportIdent)
```

- The airport frequencies accessed in WaypointAirport are an indexed list containing, in this example, 12 separate frequencies. To display or otherwise utilize any one of them requires an Index pointer. In the WAYPOINT APT FREQUENCIES DISPLAY LOOP section of the code below, line 88 is the Index pointer for the frequency list, and lines 89 through 93 are the screen display instructions. Refer to the Search> Index> Display section for more discussion.

The xml for Example 1:

```
1  <Gauge Name="NEAREST AIRPORT - AIRPORT FREQUENCIES" Version="1.0">
2  <Size X="800" Y="800" />
3
4      <Macro Name="c">C:fs9gps</Macro>
5      <Macro Name="C">C:fs9gps</Macro>
6
7  <Update Frequency="18" Hidden="No">
8  <!-- SET SEARCH VARIABLES -->
9      10 (>@c:NearestAirportMaximumItems, enum)
10     75 (>@c:NearestAirportMaximumDistance, nmiles)
11
12 <!-- SET REFERENCE POINT VARIABLES (AIRCRAFT POSITION) -->
13     (A:PLANE LATITUDE, degrees) (>@c:NearestAirportCurrentLatitude, degrees)
14     (A:PLANE LONGITUDE, degrees) (>@c:NearestAirportCurrentLongitude, degrees)
15
16 <!-- ICAO TRANSFER: NearestAirport to WaypointAirport -->
17     0 (>@c:NearestAirportCurrentLine)
18     (@c:NearestAirportCurrentICAO) (>@c:WaypointAirportICAO)
19
20 </Update>
21
22     <Element Name="BACKGROUND RECTANGLE">
23         <Position X="0" Y="0"/>
24         <Rectangle Width="800" Height="800" FillColor="white" Bright="Yes"/>
25     </Element>
```

```
<Element Name="NEAREST APT DISPLAY LOOP">
    <Position X="10" Y="5"/>
    <FormattedText X="800" Y="700" Font="Courier New" FontSize="12" LineSpacing="12"
    Color="#101010" Bright="Yes" >
        <Color Value="blue"/>
        <Color Value="darkgreen"/>
        <String>
        \{clr2}NEAREST AIRPORT SEARCH\n
        \{clr3}%((C:fs9gps:NearestAirportCurrentLatitude, degrees))%!9.4f! :Current Lat
        %((C:fs9gps:NearestAirportMaximumItems, enum))%!5d! :Max Items
        %((@c:NearestAirportItemsNumber))%!4d! :Items Num\n
        %((C:fs9gps:NearestAirportCurrentLongitude, degrees))%!9.4f! :Current Lon
        %((C:fs9gps:NearestAirportMaximumDistance, nmiles))%!5d! :Max Distance
        \n\n\{clr2}
        Current  ICAO     111 ---------------------- Current -------------------------\n
        Line     123456789012 Ident Kind Rwy*  Dist  Brg Best Appr  Com    Freq Length\n
            %((@c:NearestAirportItemsNumber) s2 0 !=)
                %{if}
                    %(0 sp1)
                    %{loop}
                        %(l1 (>@c:NearestAirportCurrentLine))
                            \{clr}%((@c:NearestAirportCurrentLine))%!-5d!
                            %((@c:NearestAirportCurrentICAO))%!16s!
                            %((@c:NearestAirportCurrentIdent))%!6s!
                            %((@c:NearestAirportCurrentAirportKind))%!5d!
                            %((@c:NearestAirportCurrentLongestAirportDirection, degrees))%!5d!
                            %((@c:NearestAirportCurrentDistance, nmiles))%!6.1f!
                            %((@c:NearestAirportCurrentTrueBearing, degrees))%!5d!
                            %((@c:NearestAirportCurrentBestApproachEnum))%!4d!
                            %((@c:NearestAirportCurrentBestApproach))%!6s!
                            %((@c:NearestAirportCurrentComFrequencyName))%!5s!
                            %((@c:NearestAirportCurrentComFrequencyValue, mhz))%!8.2f!
                            %((@c:NearestAirportCurrentLongestRunwayLength, feet))%!7d!\n
                            %(l1 ++ s1 l2 &lt;)
                    %{next}
                %{end}
        </String>
    </FormattedText>
</Element>
```

```
<Element Name="WAYPOINT APT FREQUENCIES DISPLAY LOOP">
    <Position X="10" Y="190"/>
    <FormattedText X="800" Y="700" Font="Courier New" FontSize="12" LineSpacing="12"
    Color="#100000" Bright="Yes" >
        <Color Value="blue"/>
        <Color Value="darkgreen"/>
        <Color Value="gray"/>
        <String>
        \n
        \{clr2}WAYPOINT AIRPORT FREQUENCIES\n
        \{clr4}ICAO     111
        \{clr3}%((@c:WaypointAirportFrequenciesNumber))%!6d! :Frequencies Num\n
        \{clr4}123456789012\n
        \{clr3}%((@c:WaypointAirportICAO))%!12s! :Waypoint Airport ICAO
        \n\n\{clr2}
        ----- WaypointAirportFrequency -----\n
        Freq        Name  Limit  Value  Type\n
            %((@c:WaypointAirportFrequenciesNumber) s2 0 !=)
                %{if}
                    %(0 sp1)
                    %{loop}
                        %(l1 (>@c:WaypointAirportCurrentFrequency))
                            \{clr}%((@c:WaypointAirportCurrentFrequency))%!-3d!
                            %((@c:WaypointAirportFrequencyName))%!13s!
                            %((@c:WaypointAirportFrequencyLimit))%!5d!
                            %((@c:WaypointAirportFrequencyValue, mHz))%!9.2f!
                            %((@c:WaypointAirportFrequencyType))%!6d!\n
                            %(l1 ++ s1 l2 &lt;)
                    %{next}
                %{end}
        </String>
    </FormattedText>
</Element>
```

WAYPOINT AIRPORT GROUP

The WaypointAirport Group contains all variables associated with specific airports in fs9gps. The ICAO Identification must be specified before variables can be accessed, then all subsequent variables accessed in WaypointAirport return information about that airport until the ICAO is changed.

Frequencies, Transitions, Approaches and Runways are indexed variables (lists) requiring an Index Pointer to access specific items. The rest are non-indexed.

❑ WaypointAirportICAO (12 character string) [Get, Set]

The 12 character ICAO Identification for the specific airport.

❑ WaypointAirportIdent (3 to 4 character string) [Get]

The airport IDENT code. Note that this is not the same as the 12 character ICAO. Idents are three to four characters long and often begin with the first letter of the Region code.

❑ WaypointAirportKind (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

❑ WaypointAirportLongestRunwayDirection (degrees) [Get]

Direction (magnetic) of the longest runway.

❑ WaypointAirportType (enum) [Get]

A number representing Airport Type. Referred to as AirportPrivateType in the Microsoft online ESP SDK.

WaypointAirportType

#	Type	#	Type
0	UNKNOWN_TYPE_AIRPORT = 0	2	MILITARY_TYPE_AIRPORT = 2
1	PUBLIC_TYPE_AIRPORT = 1	3	PRIVATE_TYPE_AIRPORT = 3

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

❑ WaypointAirportName (string) [Get]

Name of the airport. WaypointAirportName is the only 'name' in fs9gps that can be searched using NameSearch.

❑ WaypointAirportCity (string) [Get]

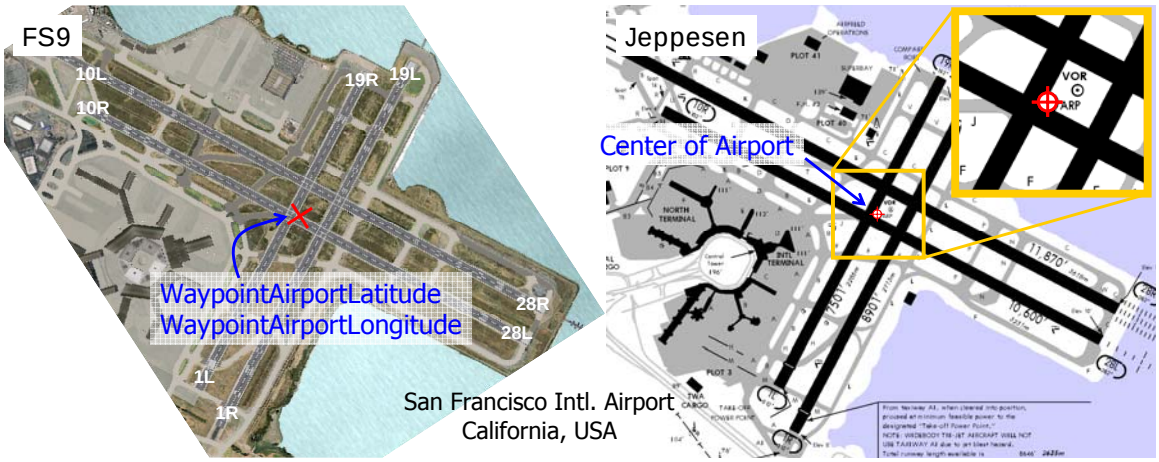
Airport city, and in the case of many airports in the North America, state or province.

❑ WaypointAirportRegion – not available

Does not exist. Although WaypointAirportRegion is listed in the SDKs as an FS9 and FSX gps variable, Region is absent in the WaypointAirportICAO, and therefore, AirportRegion is not a live variable in the WaypointAirport Group.

❑ WaypointAirportLatitude,
❑ WaypointAirportLongitude (degrees, radians) [Get]

The latitude and longitude of the airport. WaypointAirportLatitude and Longitude is the center of the runway, or in the case of large airports, the center of the airport facility. The units of Lat/Lon can be degrees (formatted +/-ddd.dddd where S16 degrees 30 minutes would be written as -16.5000) or radians (d.dddd).



❑ WaypointAirportElevation (feet) [Get]

Airport elevation, asl.

❑ WaypointAirportFuel1 (string) [Get]

If Avgas is available at the airport, WaypointAirportFuel1 = 'Avgas'. If Avgas is not available, WaypointAirportFuel1 is blank.

❑ WaypointAirportFuel2 (string) [Get]

If Jet fuel is available at the airport, WaypointAirportFuel2 = 'Jet'. If Jet fuel is not available, WaypointAirportFuel2 is blank.

❑ WaypointAirportBestApproachEnum (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

❑ WaypointAirportBestApproach (string) [Get]

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

❑ WaypointAirportRadarCoverage (enum) [Get]

This variable is not activated in FS9.

❑ WaypointAirportAirspace (string) [Get]

This variable is not activated in FS9.

❑ WaypointAirportTowered (bool) [Get]

Tower present = 1. No Tower = 0.

❑ WaypointAirportCurrentFrequency (enum) Get, Set]

Index pointer for the airport frequency list. The first frequency in the list is accessed by setting WaypointAirportCurrentFrequency=0.

❑ WaypointAirportFrequenciesNumber (enum) [Get]

Number of frequencies at the airport. Includes both Com and Nav (i.e., ILS & LOC) frequencies.

❑ WaypointAirportFrequencyName (string) [Get]

Name of the frequency. The communication frequency names are:

- ❑ Approach
- ❑ ATIS, ASOS, AWOS
- ❑ CTAF
- ❑ Unicom
- ❑ Multicom
- ❑ Clearance
- ❑ Ground
- ❑ Tower
- ❑ Departure
- ❑ FSS

Navigation frequency names are either ILS or LOC and include the runway number (e.g. ILS-24L)

❑ WaypointAirportFrequencyLimit (enum) [Get]

Frequency limited designation.

- ❑ 0 = No Limit. Transmit and Receive capability. This is the most common FrequencyLimit value in the fs9gps database.
- ❑ 1 = RX_ONLY. Receive Only. ATIS, ASOS, AWOS which transmit weather and airport information.
- ❑ 2 = TX_ONLY. Transmit only.
- ❑ 3 = PART_TIME.

❑ **WaypointAirportFrequencyValue (MHz) [Get]**

Radio frequency, usually expressed as MHz.

❑ **WaypointAirportFrequencyType (enum) [Get]**

1 = Communication frequency

2 = Navigation frequency (ILS or LOC)

❑ **WaypointAirportCurrentRunway (enum) [Get, Set]**

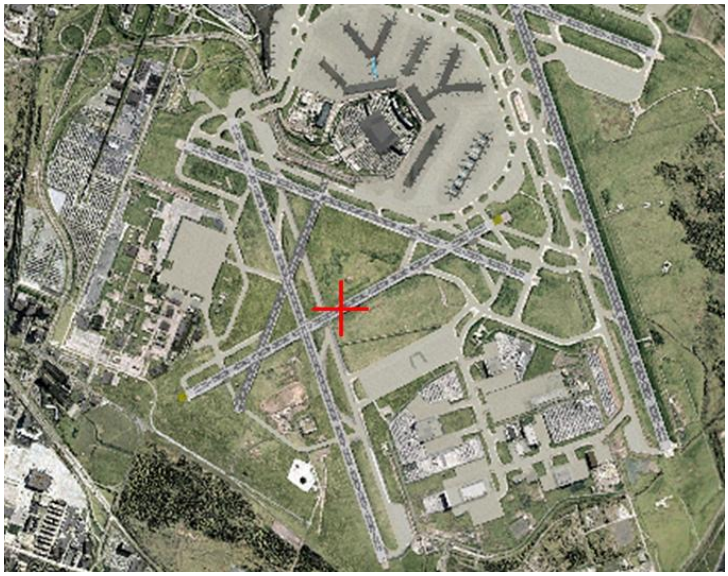
Index pointer for the airport runway list. The first runway in the list is accessed by setting WaypointAirportCurrentRunway=0.

❑ **WaypointAirportRunwaysNumber (enum) [Get]**

Number of runways at the airport. Every runway has two directions, but in the Waypoint Airport Runways list, it counts as one runway.

❑ **WaypointAirportRunwayLatitude**
❑ **WaypointAirportRunwayLongitude (degrees or radians) [Get]**

Latitude and longitude of the center point of the runway. The example below shows WaypointAirportRunwayLatitude and Longitude (41.98961 and -87.90514 degrees) for runway 04L-22R at Chicago O'Hare International Airport.



❑ WaypointAirportRunwayElevation (feet) [Get]

Elevation (asl) of the center point of the runway.

❑ WaypointAirportRunwayDirection (degrees) [Get]

Compass direction of the runway. Only the first of the pair of directions for each runway is returned. In the example above, WaypointAirportRunwayDirection of Runway 04L-22R is 39.44 degrees.

❑ WaypointAirportRunwayDesignation (string) [Get]

The designation, or name, of the runway. In the example above, "04L-22R".

❑ WaypointAirportRunwayLength (feet) [Get]

Length of the runway.

❑ WaypointAirportRunwayWidth (feet) [Get]

Width of the runway.

❑ WaypointAirportRunwaySurface (enum) [Get]

A number representing runway surface type.

RunwaySurfaceType

#	Surface Type	#	Surface Type	#	Surface Type
0	UNKNOWN = 0	105	GRAVEL = 105	112	SAND = 112
1	CONCRETE = 1	106	OIL_TREATED = 106	113	SHALE = 113
2	ASPHALT = 2	107	STEEL = 107	114	TARMAC = 114
101	GRASS = 101	108	BITUMINUS = 108	115	SNOW = 115
102	TURF = 102	109	BRICK = 109	116	ICE = 116
103	DIRT = 103	110	MACADAM = 110	201	WATER = 201
104	CORAL = 104	111	PLANKS = 111		

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

❑ WaypointAirportRunwayLighting (enum) [Get]

A number representing airport lighting type. Note that this is not list of available lighting systems for a runway, such as VASI and REIL.

RunwayLightingType

#	Lighting Type	#	Lighting Type
0	UNKNOWN = 0	3	FULL_TIME = 3
1	NONE = 1	4	FREQUENCY = 4
2	PART_TIME = 2		

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

RunwayLighting Types 2 and 4 may not exist, at least not in the fs9gps database. I have checked all runways at all airports in the fs9gps database within Europe and the USA and found no Type 2 or 4 RunwayLighting types.

❑ WaypointAirportCurrentApproach (enum) [Get, Set]

Index pointer for the airport approach list. The first approach in the list is accessed by setting WaypointAirportCurrentApproach=0.

❑ WaypointAirportApproachesNumber (enum) [Get]

The number of approach procedures for the selected airport.

❑ WaypointAirportApproachName (string) [Get]

The name of the selected approach, such as, "ILS 22R", "NDB 27R", or "RNAV 09L".

❑ WaypointAirportApproachGps (bool) [Get]

A designation indicating that the approach can be flown by the GPS receiver. ApproachGps = 1 = approach is approved for GPS use. ApproachGps = 0 = approach is not approved for GPS use. For these approaches, the GPS receiver can be used for supplemental information only.

❑ WaypointAirportApproachTransitionsNumber (enum) [Get]

The number of transitions available for the selected approach.

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

Index pointer for the approach transitions list. The first transition in the list is accessed by setting WaypointAirportCurrentTransition=0.

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

The name of the current transition, such as, "Vectors", or "PAPPI" (a waypoint approach fix).

- **WaypointAirportApproachTransitionLatitude**
- **WaypointAirportApproachTransitionLongitude (degrees or radians) [Get]**

The latitude and longitude of the center of the runway to which the selected approach applies.

- **WaypointAirportApproachTransitionSize (nmiles) [Get]**

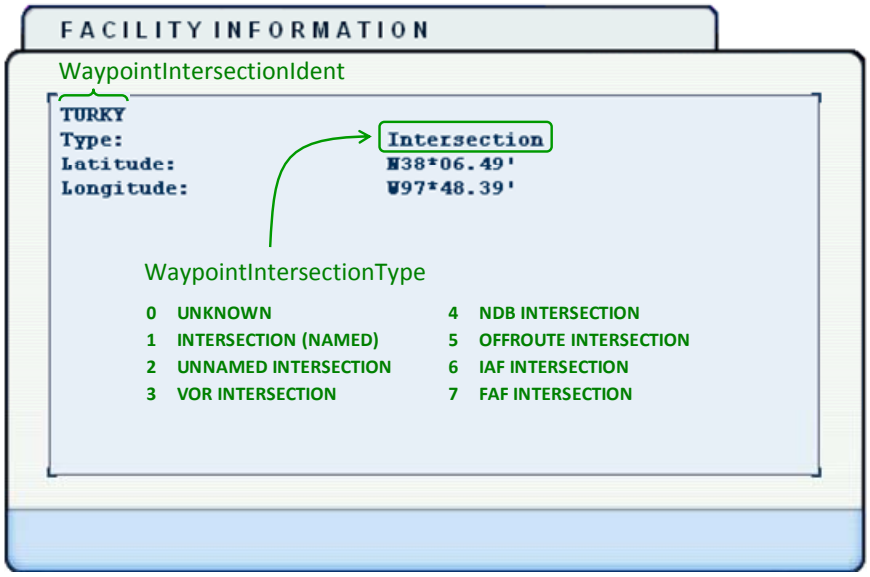
Size (radius, I assume) of the selected approach transition. It appears that all ApproachTransitionSize values in the fs9gps database are preset to 27.00 nmiles.

WAYPOINT INTERSECTION GROUP

The WaypointIntersection Group contains all variables associated with Waypoints and Intersections in the fs9gps database. The ICAO Identification must first be specified, then all variables accessed in WaypointIntersection return information about that Intersection or Waypoint until the ICAO is changed.

All variables in WaypointIntersection are non-indexed; there are no 'lists' of items associated with a specific Intersection (compared to WaypointAirport, where, for example, there are lists of different runways, approaches, transitions, and frequencies).

The screen shot below shows the Facility Information page of an example Intersection in FS9, indicating the associated gps variable names.



The following is a snapshot of all WaypointIntersection variables for the same Intersection.

```
GPS Viewer 1.2                  111111111112
GET  SET  SLEN  12345678901234567890  STRING NUMBER
 G    S    12   WK3KHUTTURKY          WaypointIntersectionICAO, string
 G         5    TURKY                 WaypointIntersectionIdent, string
 G         0                          WaypointIntersectionCity, string
 G         2    K3                    WaypointIntersectionRegion, string
 G         6      38.1081280          WaypointIntersectionLatitude, degrees
 G         7     -97.8065501          WaypointIntersectionLongitude, degrees
 G         2    1                     WaypointIntersectionType, enum
 G         3    GNP                   WaypointIntersectionNearestVorIdent, string
 G         2    2                     WaypointIntersectionNearestVorType, enum
 G         6    326.689753            WaypointIntersectionNearestVorTrueRadial, degrees
 G         6    320.689753            WaypointIntersectionNearestVorMagneticRadial, degrees
 G         10   158.043825            WaypointIntersectionNearestVorDistance, nmiles
```

❑ WaypointIntersectionICAO (string, SLEN=12) [Get, Set]

WaypointIntersectionICAO is the ICAO for the specific Intersection.

❑ WaypointIntersectionIdent (string) [Get]

The 4 to 5 character Intersection Ident.

❑ WaypointIntersectionType (enum) [Get]

An enum representing Intersection Type.

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 great majority of Intersections in the fs9gps database are Type 1 = Named, followed by Type 2, 3 and 4, which is relatively unusual. It does not appear that there are any Type 0, 5, 6, or 7 Intersections in the database.

❑ WaypointIntersectionCity (string) [Get]

WaypointIntersectionCity is not populated in the fs9gps data base. It always returns a blank string.

❑ WaypointIntersectionRegion (string) [Get]

The two character Region code.

❑ WaypointIntersectionLatitude
❑ WaypointIntersectionLongitude (degrees or radians) [Get]

The latitude and longitude of the Intersection. The units of Lat/Lon can be degrees (formatted +/-ddd.dddd where S16 degrees 30 minutes would be written as -16.5000) or radians (d.dddd).

❑ WaypointIntersectionNearestVorIdent (string) [Get]

The Ident of a VOR. It is usually NOT the nearest VOR, however.

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

An enum representing the VOR Type of the VOR returned as "Nearest". It is the correct Type for that VOR, but the VOR is usually not the nearest to the Intersection.

VOR Type Bit Map Table

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

- **WaypointIntersectionNearestVorTrueRadial (degrees) [Get]**
- **WaypointIntersectionNearestVorMagneticRadial (degrees) [Get]**
- **WaypointIntersectionNearestVorDistance (NMiles) [Get]**

The WaypointIntersectionNearestVor variables represent the radial bearings and distance from a VOR, but not necessarily the nearest VOR.

NOTE: WaypointIntersectionNearestVor appears to be a software bug. Exercise caution when using WaypointIntersectionNearestVor variables.

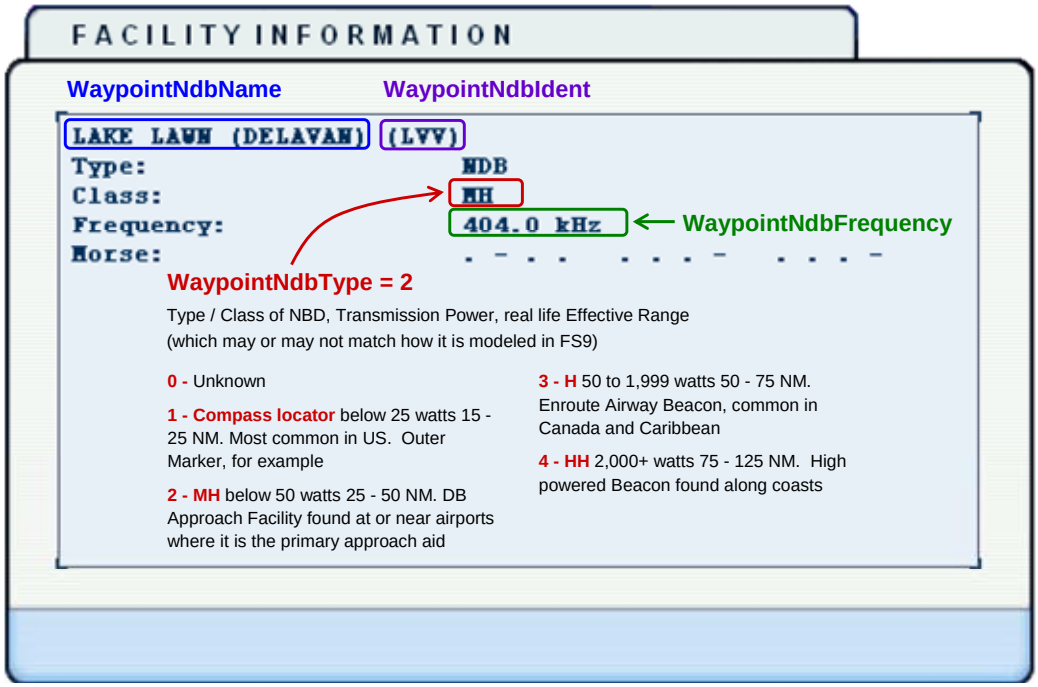
The NearestVOR search works properly, however. Use that approach to find the nearest VOR to an Intersection.

WAYPOINT NDB GROUP

The WaypointNdb Group contains all variables associated with specific Non-Directional Beacons in the fs9gps database. The ICAO Identification must be specified before variables can be accessed, then all subsequent variables accessed in WaypointNdb return information about that NDB until the ICAO is changed.

All variables in WaypointNdb are non-indexed; there are no 'lists' of items associated with a specific NDB (compared to WaypointAirport, where, for example, there are lists of different runways, approaches, transitions, and frequencies).

The figure below is a screen shot of the Facility Information page of an example NDB in FS9, indicating the associated gps variable names.



The following is a snapshot of all WaypointNdb variables for the same NDB.

```
GPS Viewer 1.2                  11111111112
GET  SET  SLEN  12345678901234567890  STRING NUMBER
 G    S    12   NK5      LVV          WaypointNdbICAO, string
 G         3    LVV                   WaypointNdbIdent, string
 G         2    2                     WaypointNdbType, enum
 G         19   LAKE LAWN (DELAVAN)   WaypointNdbName, string
 G         0                          WaypointNdbCity, string
 G         2    K5                    WaypointNdbRegion, string
 G         6      42.6988333          WaypointNdbLatitude, degrees
 G         7     -88.5931666          WaypointNdbLongitude, degrees
 G         6    947.998733            WaypointNdbElevation, feet
 G         9    404                   WaypointNdbFrequency, khz
 G         2    0                     WaypointNdbWeatherBroadcast, enum
 G         2     0.0                  WaypointNdbMagneticVariation, degrees
```

❑ WaypointNdbICAO (string, SLEN=12) [Get, Set]

The ICAO for the specific NDB. Some NDBs are nav fixes in fs9gps approach procedures. These NDBs include the "owning" airport Ident in ICAO character positions 4 through 7. See discussion in ICAO Idents.

❑ WaypointNdbIdent (string) [Get]

The 1 to 5 character NDB Ident

❑ WaypointNdbType (enum) [Get]

A number representing NDB Type (Class).

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.

❑ WaypointNdbName (string) [Get]

Name of the NDB. Interestingly, some NDBs also contain the city name in parenthesis following the NDB name – all part of the variable WaypointNdbName. I do not understand the rules/reasons that some do and some do not. In the example above, Lake Lawn is the NDB name, Delavan is the city. NDB Names are not searchable using NameSearch.

❑ WaypointNdbCity (string) [Get]

WaypointNdbCity is not populated in the fs9gps data base. It always returns a blank string.

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

The two character Region code.

- **WaypointNdbLatitude**
- **WaypointNdbLongitude (degrees or radians) [Get]**

The latitude and longitude of the NDB. The units of Lat/Lon can be degrees (formatted +/-ddd.dddd where S16 degrees 30 minutes would be written as -16.5000) or radians (d.dddd).

- **WaypointNdbElevation (feet) [Get]**

Elevation (asl) of the NDB facility.

- **WaypointNdbFrequency (kHz) [Get]**

Radio frequency of the NDB. Commonly expressed in kHz.

- **WaypointNdbWeatherBroadcast (gps boolean) [Get]**

The ESP SDK indicates that WaypointNdbWeatherBroadcast is a gps boolean:

- 0 = Unknown
- 1 = No
- 2 = Yes

However, having scanned most NDBs in the fs9gps database, so far I have found all NDBs have WaypointNdbWeatherBroadcast = 0. Consequently, this variable may not represent an active feature in FS9.

- **WaypointNdbMagneticVariation (degrees) [Get]**

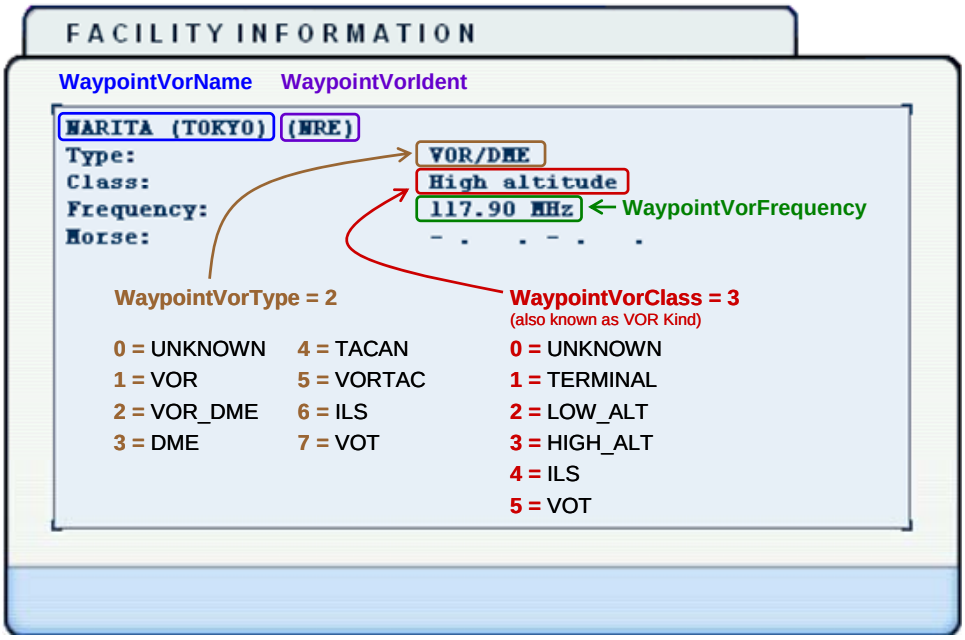
WaypointNdbMagneticVariation is the magnetic variation at the NDB location. Magnetic Variation is the difference between the compass magnetic indication and True North, measured in degrees longitude. This is an integer number in the fs9gps database.

WAYPOINT VOR GROUP

The WaypointVor Group contains all variables associated with specific VOR, VORTAC and VOR-DME beacons in the fs9gps database. The ICAO Identification must be specified before variables can be accessed, then all subsequent variables accessed in WaypointVor return information about that VOR until the ICAO is changed. ILS and LOC are not part of the WaypointVor Group even though the ICAO Type for ILS and LOC is 'V'. These two Nav facilities belong to the WaypointAirport Group.

All variables in WaypointVor are non-indexed; there are no 'lists' of items associated with a specific VOR (compared to WaypointAirport, where, for example, there are lists of different runways, approaches, transitions, and frequencies).

The figure below is a screen shot of the Facility Information page of an example VOR in FS9, indicating the associated gps variable names.



The following is a snapshot of all WaypointVor variables for the same VOR.

```
GPS Viewer 1.2                    11111111112
GET  SET  SLEN  12345678901234567890  STRING NUMBER
 G    S    12   VRJ      NRE          WaypointVorICAO, string
 G         3    NRE                   WaypointVorIdent, string
 G         2    2                     WaypointVorType, enum
 G         2    3                     WaypointVorClass, enum
 G         14   NARITA (TOKYO)        WaypointVorName, string
 G         0                          WaypointVorCity, string
 G         2    RJ                    WaypointVorRegion, string
 G         6      35.7823473          WaypointVorLatitude, degrees
 G         6     140.3625360          WaypointVorLongitude, degrees
 G         6    153.999351            WaypointVorElevation, feet
 G         10   117.900000            WaypointVorFrequency, mhz
 G         2    0                     WaypointVorWeatherBroadcast, enum
 G         2      7.00                WaypointVorMagneticVariation, degrees
```

Note that for the Narita VOR, the City is part of the WaypointVorName and is displayed in parenthesis (Tokyo).

❑ WaypointVorICAO (string, SLEN=12) [Get, Set]

WaypointVorICAO is the ICAO for the specific VOR.

❑ WaypointVorIdent (string) [Get]

The 2 to 3 character VOR Ident.

❑ WaypointVorType (enum) [Get]

A number representing VOR Type.

VOR Type

#	VOR Type	#	VOR 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

❑ WaypointVorClass (enum) [Get]

A number representing VOR Class, also known as VOR Kind.

VOR Class (Kind)

#	VOR Class	#	VOR Class
0	UNKNOWN = 0	3	HIGH_ALT = 3
1	TERMINAL = 1	4	ILS = 4
2	LOW_ALT = 2	5	VOT = 5

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

❑ WaypointVorName (string) [Get]

Name of the VOR. Interestingly, some VORs also contain the city name in parenthesis following the VOR name – all part of the variable WaypointVorName. I do not understand the rules/reasons that some do and some do not. In the example above, Narita is the VOR name, Tokyo is the city. VOR Names are not searchable using NameSearch.

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

WaypointVorCity is not populated in the fs9gps data base. It always returns a blank string.

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

The two character Region code.

- **WaypointVorLatitude**
- **WaypointVorLongitude (degrees or radians) [Get]**

The latitude and longitude of the VOR. The units of Lat/Lon can be degrees (formatted +/-ddd.dddd where S16 degrees 30 minutes would be written as -16.5000) or radians (d.dddd).

- **WaypointVorElevation (feet) [Get]**

Elevation (asl) of the VOR facility.

- **WaypointVorFrequency (kHz) [Get]**

Radio frequency of the VOR. Commonly expressed in MHz.

- **WaypointVorWeatherBroadcast (gps boolean) [Get]**

The ESP SDK indicates that WaypointVorWeatherBroadcast is a gps boolean:

- 0 = Unknown
- 1 = No
- 2 = Yes

However, having scanned most VORs in the fs9gps database, so far I have found all VORs have WaypointVorWeatherBroadcast = 0. Consequently, this variable may not represent an active feature in FS9.

- **WaypointVorMagneticVariation (degrees) [Get]**

WaypointVorMagneticVariation is the magnetic variation at the VOR location. Magnetic Variation is the difference between the compass magnetic indication and True North, measured in degrees longitude. This is an integer number in the fs9gps database.