

# TopSky plugin for EuroScope

---

*- version 2.2.1 -*

Developer Guide

## Table of Contents

<b>1</b>	<b>EuroScope setup.....</b>	<b>3</b>
1.1	General Settings .....	3
1.1.1	Display options .....	3
1.1.2	TAG display options .....	3
1.1.3	Miscellaneous options .....	3
1.2	Display settings dialog .....	3
1.3	Symbology dialog .....	4
1.3.1	Colors .....	4
1.3.2	Symbols .....	4
1.4	Plug-ins Dialog.....	4
1.5	Conflict Alert Settings Dialog .....	5
1.5.1	STCA Lower altitude.....	5
1.5.2	STCA Higher level.....	5
1.5.3	MTCA options .....	5
1.6	Plugin data files and internal settings .....	6
1.7	TAG editor .....	7
1.7.1	General.....	7
1.7.2	Tagging levels .....	7
1.7.3	Filtering .....	7
1.8	Flight lists setup.....	7
1.9	CPDLC.....	7
1.9.1	Tag items .....	7
1.9.2	Tag functions .....	8
<b>2</b>	<b>External data files.....</b>	<b>9</b>
2.1	TopSkyAirspace.txt .....	10
2.2	TopSkyAreas.txt.....	12
2.3	TopSkyAreasManualAct.txt.....	18
2.4	TopSkyCPDLC.txt .....	19
2.5	TopSkyCallsigns.txt & TopSkyCallsignsLocal.txt .....	22
2.6	TopSkyMaps.txt & TopSkyMapsLocal.txt.....	23
2.7	TopSkyMSAW.txt.....	32
2.8	TopSkyRadars.txt.....	34
2.9	TopSkySettings.txt & TopSkySettingsLocal.txt .....	37
2.10	TopSkySSRcodes.txt.....	38
2.11	TopSkySTCA.txt .....	47
2.12	TopSkySymbols.txt .....	50
2.13	TopSkyViews.txt .....	52
<b>3</b>	<b>Sound files.....</b>	<b>53</b>
<b>4</b>	<b>Cursor files .....</b>	<b>53</b>

## 1 EuroScope setup

This chapter is meant as a guide for users who either didn't get the plugin as a part of a package including all the settings files, or who want to create their own setup.

To load the plugin and setup EuroScope to make the most out of it, use the following settings in the EuroScope settings dialogs. Settings not mentioned shouldn't make any difference. Remember to save the settings when exiting ES. Use of the plugin file must be saved to each profile file separately for it to be loaded automatically.

### 1.1 General Settings

#### 1.1.1 Display options

- |                                   |     |
|-----------------------------------|-----|
| - Show route when accepting       | Off |
| - Lock show route when accepting  | Off |
| - Rotate radar target symbol      | Off |
| - Rotate flight plan track symbol | Off |
| - Show CLAM warnings              | On  |
| - Show RAM warnings               | On  |

The plugin uses its own route display and the default plugin track labels don't offer any way of hiding the ES route display so the first two should be selected off. If both settings are left "on" and the track labels are not modified, the radar screen will soon be filled with aircraft routes with no way to remove them. The CLAM and RAM warning settings also affect the plugin. If they are selected off, the plugin can't show the warnings either.

#### 1.1.2 TAG display options

- |  |     |
|--|-----|
| - Allow correlated aircraft tag untagged | On  |
| - Allow concerned aircraft tag untagged  | Off |
| - Allow assumed aircraft tag untagged    | Off |
| - Show detailed over untagged            | On  |

These settings are needed to display the correct type of track label for each aircraft.

#### 1.1.3 Miscellaneous options

- |   |    |
|---|----|
| - Keep scratch pad content after direct | On |
|---|----|

The plugin stores the manual alerts in the scratch pad. If this setting is selected off and a direct clearance is given, any manual alerts are removed as a result.

### 1.2 Display settings dialog

- |                                 |     |      |
|---------------------------------|-----|------|
| - Number of history dots        | 0   |      |
| - Do not display aircraft below | 0   | feet |
| - Do not display aircraft above | 0   | feet |
| - Show leader lines             | Off |      |

The plugin draws its own custom history dots and leader lines so the ES default ones must be selected off. Similarly, the plugin has its own altitude filtering settings, so the ES filters should be off.

## 1.3 Symbology dialog

### 1.3.1 Colors

Whether to use the transparency settings is up to the user, and their colors should be set as desired. The plugin colors determine the track label color but the EuroScope defined color is used to draw the leader line between the position symbol and the label. Some EuroScope colors have a corresponding plugin color setting as listed below. They should be set to the same value.

<u>EuroScope color</u>	<u>Plugin color</u>
------------------------	---------------------

Aircraft colors:

- |                                    |                        |
|------------------------------------|------------------------|
| - Datablock/non concerned          | Color_Unconcerned      |
| - Datablock/notified               | Color_Coordination     |
| - Datablock/assumed                | Color_Assumed          |
| - Datablock/redundant              | Color_Redundant        |
| - Datablock/information            | Color_Warning          |
| - Datablock/detailed background    | Color_Flight_Highlight |
| - Datablock/active item background | Color_Field_Highlight  |

List colors

- |                     |                  |
|---------------------|------------------|
| - Other/list header | Color_Foreground |
|---------------------|------------------|

Background colors:

- |                                     |                       |
|-------------------------------------|-----------------------|
| - Sector/active sector background   | Color_Active_Sector   |
| - Sector/inactive sector background | Color_Inactive_Sector |

If adjusted from the default ones, the background colors should be chosen so that all plugin colors can be seen. Medium grey colors work well with the default plugin colors of plugin type **A**, dark colors with **B**, but anything can be used as long as the plugin colors are taken into account and changed accordingly.

“Other/normal menu item” and “Other/disabled menu item” should be different colors to be able to see disabled menu items in the setup menus.

### 1.3.2 Symbols

The “Aircraft primary radar only”, “Aircraft flight plan track”, “Aircraft coasting”, “History dot” and all entries beginning with “Aircraft corr.” or “Aircraft uncorr.” are drawn by the plugin and should be set to “MOVETO 0 0” only. All other symbols are drawn by ES and should be set as desired.

## 1.4 Plug-ins Dialog

Load the plugin file (TopSky.dll). After loading it, highlight its name (“TopSky plugin”) in the list and move “Standard ES radar screen” from the “Forbidden to draw on types” box to the “Allowed to draw on types” box.

## 1.5 Conflict Alert Settings Dialog

The plugin has its own STCA and MTCD systems and doesn't use the ES default ones. Selecting the below settings will disable the warnings from the ES default systems.

### 1.5.1 STCA Lower altitude

- Show lower altitude STCA Off

### 1.5.2 STCA Higher level

- Show higher level STCA Off

### 1.5.3 MTCA options

- Warn if vertical separation is less 0 feet
- Warn if horizontal separation is less 0 nm

## 1.6 Plugin data files and internal settings

The plugin comes with some pre-defined setups, but it is possible to adjust hundreds of different settings using the [settings files](#). This makes it possible to update the existing setups and to create completely new ones. Even though the ranges of acceptable values have been limited to prevent major problems, care should be taken when adjusting any settings as the results can be unexpected.

When creating a new setup, only the **TopSkyCallsigns.txt** and **TopSkyCallsignsLocal.txt** data files can be used from an old setup, as they contain only general callsign data that can be used anywhere. All the other data files contain information specific to the original setup, and should not be used.

Some setup-specific plugin settings are listed below. At least these settings should be looked at when creating a new setup, but many of the others are important too. However, never change a setting without understanding what it does.

Setup_COOPANS	Define which of the two plugin types to activate ( <b>A</b> or <b>B</b> )
Airspace_C_Flag	Define the list of clearance flag airports
Airspace_C_No_Flag	
Airspace_P_Flag	Define the list of P-RNAV airports (applicable to plugin type <b>B</b> only)
Airspace_P_No_Flag	
Airspace_ASSR_Type	Define the transponder code ranges
Airspace_ASSR_StartCode	
Airspace_ASSR_EndCode	
Airspace_No_DUPE_Codes	Define the transponder codes that do not produce a DUPE alert
Airspace_SIGMET_Areas	Define airspace information for the SIGMET/NOTAM functions
Airspace_SIGMET_Pages	
Airspace_NOTAM_Add	
Airspace_NOTAM_Remove	
Label_RTE_Origin	Define airport identifiers for the ROUTE label item
WXR_Latitude	Define the weather radar tile center point (for performance reasons, look into the <code>_Min</code> and <code>_Max</code> settings as well)
WXR_Longitude	

## 1.7 TAG editor

To make the plugin work as it's supposed to, there are some rules that need to be followed when creating the tag families:

### 1.7.1 General

- For every correlated and flight plan track tag type:
  - o The *untagged* and *tagged* tags must contain the “**Dummy item – not detailed tag**” item
  - o The *detailed* tag must contain the “**Dummy item – detailed tag**” item
- Tag items marked “**(0-1)**” are only displayed when the track is unconcerned or notified, items with “**(2+)**” when the track is in any other state. Other items are shown in all states.
- A couple of the indicator items have an “**(inactive)**” version as well. Both can be used in the same tag, as only one of them will be shown (the inactive version can then be used to toggle the indicator back to active state if required).

### 1.7.2 Tagging levels

- When a tag item has an “**(unselected track)**” variant, it should be used in the *untagged* and *tagged* tagging levels, and the normal variant in the *detailed* tagging level. For tag items without specific variants, the normal variant can be used in all tagging levels.
- Items beginning with “**ET**” should be used in the *detailed* tagging level (they are only displayed in the extended label which is a sublevel of the *detailed* tagging level in the plugin, displayed by the “**Open extended tag**” function). Items with “**(not ET)**” will not be shown in the extended label. Other items are displayed in both the normal *detailed* tag and the extended label.

### 1.7.3 Filtering

- Items beginning with “**List**” should only be used in flight lists, other items only in tags (“List” items are not hidden when a label is subject to filtering, the other ones are)
- Tags should be constructed only from TopSky plugin items. If this is not possible, the plugin's label filtering functionalities may not work as they can only toggle the visibility of its own tag items.

## 1.8 Flight lists setup

Of the plugin tag items, only the ones beginning with “**List**” should be used in flight lists (“List” items are not hidden when a label is subject to filtering, the other ones are).

## 1.9 CPDLC

The following tag items (all variants – List, ET, etc. unless otherwise specified) and functions have CPDLC related features, and should be incorporated in the relevant tag definitions and flight lists to provide the intended functionality.

### 1.9.1 Tag items

- |            |   |
|------------|---|
| - AHDG     | Displays heading uplink status                  |
| - ASSR     | Displays squawk code uplink status              |
| - CALLSIGN | Displays CPDLC connection status                |
| - CFL      | Displays level uplink status                    |
| - CPDLC_E  | Displays received CPDLC emergency messages      |
| - CPDLC_W  | Displays message and connection status messages |
| - List CLR | Displays DCL/PDC request status                 |
| - NPT      | Displays direct-to downlink status              |

- NSSR                      Displays squawk code uplink status
- PFREQ                    Displays pilot monitored frequency after “monitor” uplink has been used
- PRFL                      Displays level request downlink status
- List RFL                  Displays pilot requested level
- SI                          Displays frequency change uplink status

### 1.9.2 Tag functions

- Acknowledge PFREQ                      Acknowledges the information  
(use with PFREQ tag item)
- CPDLC Warning functions                      Acknowledges warning or opens a menu  
(use with CPDLC\_W tag item)
- Open AHDG menu                      Used to send/answer heading uplinks/downlinks  
and send direct-to uplinks
- Open ASP menu                      Used to send/answer speed uplinks/downlinks
- Open Callsign menu                      Contains many CPDLC related items
- Open CFL menu / Open CFL/PEL menu                      Used to send level uplinks
- Open CPDLC Current Message Window (\*)                      Opens the window  
(use with the specified items, PRFL also with “List RFL”)
- Open CPDLC Emergency Acknowledgement menu                      Opens the menu  
(use with CPDLC\_E tag item)
- Open PDC menu                      Used to send DCL/PDC
- Open RFL menu                      Used to reply standby/unable to level downlinks
- Send CPDLC squawk SSR                      Sends squawk code uplink



## 2 External data files

This chapter gives guidance on developing the external data files used by the plugin for various features. The data files must be located in the same folder as the plugin dll. Even though the plugin does its best to check the data for errors, some errors may get through and cause all kinds of issues, possibly leading to ES crashing, so it's important to be careful to provide correctly formed data when creating the files. Errors found in the data files by the plugin can be seen in the "Plugin Status" submenu. It is found in the "Status" or "STS" menu depending on the plugin version.

To reload a data file, left-click on the "Reload" button. Some of the data can also be visually checked for correctness by left-clicking the "View" button. The areas will then be shown on the radar screen.

## 2.1 TopSkyAirspace.txt

The file contains the QNH value to transition level tables for the QNH/TL Window and any custom runway approach lines (i.e. starting point not on the runway threshold and/or approach course not aligned with the runway centerline) and lists of custom level menu values.

For the QNH/TL tables, the file is read one line at a time and the first matching line will be used. The following example shows a possible setup:

// Made up lines...	Comment
QNHTL:XBZZ:80	QNHTL
QNHTL:XB,XC:80,978,75,996,70	QNHTL

For the runway approach lines, any lines defined here will override the corresponding lines automatically created from the sector file data. The following line shows an example:

APPLINE:EFHK:15:153.4:HEL	AppLine
---------------------------	---------

For the custom level lists, the file is read one line at a time and the first matching line will be used. The following example shows a possible setup:

LEVELS:EETN:22,25,32,35,50	LEVELS
LEVELS*:15,20,30	LEVELS

### QNHTL

#### QNHTL:ICAOlist:TL

#### QNHTL:ICAOlist:TL<sub>1</sub>,QNH<sub>1</sub>,TL<sub>2</sub>,QNH<sub>2</sub>, ... ,TL<sub>n</sub>,QNH<sub>n</sub>,TL<sub>n+1</sub>

Defines a variable transition level based on the QNH value for the specified airports.

- ICAOlist                      Comma-separated list of airport ICAO designators. Either complete designators or one to three first letters of the designator.
- TL<sub>n</sub>                            Transition level value (text string, will be displayed exactly as written)
- QNH<sub>n</sub>                         QNH value (integer value, see also below)

The first definition sets a fixed transition level, while the second one creates a table of transition levels based on QNH values.

The QNH value must be in the same format as in the METAR. If the METAR reports the QNH in hPa (Qxxxx), the values in the list must be in hPa as well. If the METAR reports the QNH in inHg (Axxxx), the values in the list must be in inHg\*100.

The TL/QNH list must contain one or more pairs of TL and QNH values followed by a TL value in the end.

The transition level is found in the following way: the actual QNH value found in the METAR is compared against the values in the list, from left to right. If the actual QNH is less than the list value, the corresponding TL value (the one before the QNH value) is used. If not, the next QNH value in the list is

checked and so on. If the actual QNH value is equal to or greater than all the values in the list, the TL value after the last QNH value in the list ( $TL_{n+1}$ ) is used.

## *AppLine*

**APPLINE:AirportCode:Runway:AppCourseT:PointName**

**APPLINE:AirportCode:Runway:AppCourseT:Lat:Lon**

Defines a runway approach line with a specified approach course and end point (either as a point name or coordinates). By default, approach lines are created for all runways in the active sector file with the approach courses set on the extended runway center lines and ending at the runway thresholds. If a runway needs some other approach line instead, it can be defined here. Any runway approach line defined here will override the plugin created default one for the same runway.

- AirportCode            Airport ICAO code
- Runway                Runway identifier
- AppCourseT            Approach track (degrees true, decimal value)
- PointName            Fix, VOR, NDB, airport code or runway (must be found in the active sector file)
- Lat                    Endpoint Latitude (decimal degrees or sector file format)
- Lon                    Endpoint Longitude (decimal degrees or sector file format)

*Note: the syntax to define a runway threshold as a PointName is the 4-letter ICAO airport designator followed by a forward slash and the runway identifier.*

## *Levels*

**LEVELS:AirportCode:LevelList**

Defines a custom level list to be used in all plugin level menus (AFL, CFL and RFL). The list here overrides the default list values up to the highest value in the custom list, which after the default list takes over. The airport code is compared against the login callsign, and if a matching list is found, it will be used. In case a matching list is not found, the code then compares the lists against the aircraft's departure airport until more than 1/3 of the way to the destination, and then against the destination airport.

- AirportCode            Airport ICAO code. "\*" matches any airport, other wildcards are not supported
- LevelList                Comma-separated list of levels (in 100's of feet, range 1-999)

## 2.2 TopSkyAreas.txt

This file contains the areas for the APW and SAP functionality as well as the MTCD and STCA inhibit areas. The following example area is used to show the syntax (optional lines in grey color).

CATEGORYDEF:D:7:0:5:0:0:0	CategoryDef
// EF D101 Isosaari	Comment
AREA:T:EFD101	Name
CATEGORY:D	Category
ACTIVE:NOTAM:EFIN:EF D101	Active
ACTIVE:NOTAM:EFIN:EFD101	Active
LABEL:N059.55.08.817:E025.07.08.496:D101	Label
LIMITS:0:390	Limits
N059.54.15.000 E025.15.06.000	Coordinate
N059.53.27.000 E024.59.49.000	Coordinate
N059.56.36.000 E025.10.10.000	Coordinate

### CategoryDef

**CATEGORYDEF:**Name:ActBorderColor:ActFillColor:ActFillPattern:PreBorderColor:PreFillColor:PreFillPattern

**CATEGORYDEF:**Name:ActBorderColor:ActFillColor:ActFillPattern:PreBorderColor:PreFillColor:PreFillPattern:LabelName:LabelMapText:LabelUserText:LabelLevels:LabelTimes

This line defines the display colors for an area category. The default border color is *Active Map* for unfilled and *Active RD Map* for filled areas. The default fill color is *Active RD Infill Map*, and the fill percentage that was used in previous plugin versions is 50. If the label settings are left out, the default settings are used.

- Name                      Name for the category (text string)
- ActBorderColor           Active area border color
  - 0                      use default color
  - 1-20                custom color (*Active Map Type X*)
- ActFillColor            Active area fill color (values as in ActBorderColor)
  - 0                      use default color
  - 1-20                custom color (*Active Map Type X*)
- ActFillPattern          Active area fill pattern
  - 0                      no fill
  - 5, 10, 20, 25, 30, 40, 50, 60, 70, 75, 80, 90, 100      percentage to fill
- PreBorderColor          Pre-active area border color (values as in ActBorderColor)
- PreFillColor            Pre-active area fill color (values as in ActFillColor)
- PreFillPattern          Pre-active area fill pattern (values as in ActFillPattern)
- LabelName               Show area name in area label (1=yes, 0=no)
- LabelMapText            Show map text in area label (1=yes, 0=no)
- LabelUserText           Show user text in area label (0=no,1=manually entered/pre-defined,2=also activity period specific,3=also NOTAM id)
- LabelLevels             Show area level limits in area label (1=yes, 0=no)
- LabelTimes              Show area activation times in area label (1=yes, 0=no)

## Name

### AREA:AreaType:AreaName

The first line for each area definition must be a *name* line. Type S areas will appear in the Safety Nets Status Window, type M areas in the MTCD Status Window, the other types in the Airspace Management Window.

- AreaType                      Area type (one of the following):
  - T                                      (TSA area)
  - 1, 2, 3, 4 or 5                      (TSA area)
  - 1F, 2F, 3F, 4F or 5F              (TSA area)
  - M                                      (MTCD inhibit area)
  - S                                      (STCA inhibit area)
- AreaName                      Area name to identify it in the areas window (text string)

TSA area types 1F-5F are filled, 1-5 are not (the number defines the area border color). For area type T the colors and filling are defined in a *category* line. If a *category* line is not defined for a type T area, the area is not filled and a default border color is used. For compatibility with earlier plugin versions, area types D and R are also supported (they are TSA areas that are filled)

Area type T and the category definitions are meant to be used with plugin type **A** and area types 1-5 and 1F-5F with plugin type **B**, but their use is not restricted by the code.

Of the line types described below, area types M and S only support *active* (partly, only “ACTIVE:1”), *bound*, *limits* and *coordinate*. TSA areas support all line types.

## Category

### CATEGORY:Name

The *category* line is optional. It defines the display colors for type T areas.

- Name                              Category name (must have been defined earlier in the file)

## Label

### LABEL:Lat:Lon:LabelText

The *label* line is optional. It displays the area name and/or altitude limits on the radar display.

- Lat                                  Latitude for the label (decimal degrees or sector file format)
- Lon                                  Longitude for the label (decimal degrees or sector file format)
- LabelText                      Label text (text string)

## Group

### GROUP:Name

The *group* line is optional. It defines the area's group name if desired (only used in plugin type **A** )

- Name                      Group name

## Usertext

### USERTEXT:Text

The *usertext* line is optional. It defines the default user text for the area.

- Text                      User text string

## Active

### ACTIVE:1

**ACTIVE:SchedStartDate:SchedEndDate:SchedWeekdays:StartTime:EndTime[:Lower:Upper:UserText]**

**ACTIVE:NOTAM:Icao:Text**

**ACTIVE:RWY:ARR:ArrRwyList:DEP:DepRwyList**

The *active* line is optional. If there are no *active* lines defined, the area will not be automatically activated. An area can contain more than one *active* line; the plugin will check all of them to set the activation status.

The first option will activate the area automatically without any time limits when the plugin is loaded. Note that this option cannot be used together with other *active* lines as it would override any other schedule.

The second option can be used to set activation schedules.

- SchedStartDate              First day to activate the area
  - month and day in the format MMDD (for recurring periods every year)
  - year, month and day in the format YYYYMMDD (for a single period)
- SchedEndDate              Last day to activate the area, formats as above
- SchedWeekdays              Days of the week to activate the area
  - list of numbers representing the days to activate the area, for example "145" means the area will activate on Mondays, Thursdays and Fridays
  - "0" (zero) to activate the area continuously from StartTime on SchedStartDate to EndTime on SchedEndDate
- StartTime                      Time to activate the area (UTC time in the format HHMM)
- EndTime                      Time to deactivate the area (UTC time in the format HHMM)
- Lower                      Lower limit for the area (feet, 0-99900)
- Upper                      Upper limit for the area (feet, 0-99900)
- UserText                      Displayed user text string

*Note: SchedEndDate and SchedWeekdays only limit the activation of the area. If the activation time extends past midnight, the area stays active until EndTime on the following day. Lower, Upper and UserText are optional fields. If any of them are specified, all the preceding fields must be included as well (i.e. if UserText is specified, both Lower and Upper must also be).*

The third option activates the area based on NOTAM information. The plugin then attempts to set the activation schedule based on the other contents of received NOTAMs.

- Icao ICAO location indicator that publishes activation NOTAMs for the area
- Text Text to search for in the NOTAM

The fourth option activates the area based on active runways. If all the specified runways are active, the area is activated. If even one of them is not, the area will be deactivated. The runway identifiers must be in the format "<4-letter ICAO code><runwayID>", for example "EFHK15".

- ArrRwyList Comma-separated list of runways. Enter "\*" to allow any runway.
- DepRwyList Comma-separated list of runways. Enter "\*" to allow any runway.

## Bound

### BOUND:C:Lat:Lon:Radius

The *bound* line is optional but highly recommended for areas that are circle-shaped. It increases the accuracy of the calculation while at the same time reducing the amount of calculations required, giving both an accuracy and a performance gain to the plugin.

The bound line should only be used for areas that are circles. The "*Lat*" and "*Lon*" coordinates (decimal degrees or sector file format) define the center point and the "*Radius*" (nautical miles, decimal number) the radius of the circle. The information is used to check if the aircraft is inside the area, but also the coordinate lines are still needed as they are used to draw the area on the screen (the coordinates will not be used for any calculations so make sure you only use this line for circular areas!).

## Limits

### LIMITS:Alt<sub>min</sub>:Alt<sub>max</sub>

The *limits* line is optional. It defines the default vertical limits of the area (in hundreds of feet). They can be changed as required in the area windows. When an area without default vertical limits is activated, its limits will be set to 0 and 999 (from 0ft to FL999). Those are also the minimum and maximum allowed values.

## Elevation

### ELEVATION:Min:Max

The *elevation* line is optional. It defines the minimum and maximum ground elevation in the area (in hundreds of feet). The default values are 0, available values are from -10 to 999 (-1000ft to 99900ft). The values are used to correct the vertical limits when the area is activated by NOTAM and AGL values are specified. The Min value is used to correct the lower limit and the Max value the upper limit. The Max value may not be lower than the Min value.

## *NoMSAW*

### **NOMSAW**

The optional *NoMSAW* line inhibits MSAW warnings inside the area.

## *NoAPW*

### **NOAPW**

The optional *NoAPW* line inhibits APW warnings for the area.

## *APW\_Buffer\_Lat*

### **APW\_BUFFER\_LAT:BufferU:BufferLI:BufferLV**

The optional *APW\_Buffer\_Lat* line can be used to override the default lateral buffers applied to the area for APW processing. The first value is used above the setting value “APW\_Buffer\_Lat\_SepLevel”, the others at or below it.

- BufferU                      High level buffer (nautical miles, decimal value, range 0.0-999.0)
- BufferLI                    Low level buffer for IFR flights (nautical miles, decimal value, range 0.0-999.0)
- BufferLV                    Low level buffer for VFR flights (nautical miles, decimal value, range 0.0-999.0)

## *APW\_Buffer\_Vert*

### **APW\_BUFFER\_VERT:BufferU:BufferLI:BufferLV**

The optional *APW\_Buffer\_Vert* line can be used to override the default vertical buffers applied to the area for APW processing. The first value is used above the minimum RVSM level, the others at or below it.

- BufferU                      High level buffer (feet, integer value, range 0-9999)
- BufferLI                    Low level buffer for IFR flights (feet, integer value, range 0-9999)
- BufferLV                    Low level buffer for VFR flights (feet, integer value, range 0-9999)



## *NoSAP*

### **NOSAP**

The optional *NoSAP* line inhibits SAP warnings for the area.

## *SAP\_Buffer\_Lat*

### **SAP\_BUFFER\_LAT:BufferU:BufferLI:BufferLV**

The optional *SAP\_Buffer\_Lat* line can be used to override the default lateral buffers applied to the area for SAP processing. The first value is used above the setting value “SAP\_Buffer\_Lat\_SepLevel”, the others at or below it.

- BufferU High level buffer (nautical miles, decimal value, range 0.0-999.0)
- BufferLI Low level buffer for IFR flights (nautical miles, decimal value, range 0.0-999.0)
- BufferLV Low level buffer for VFR flights (nautical miles, decimal value, range 0.0-999.0)

## *SAP\_Buffer\_Vert*

### **SAP\_BUFFER\_VERT:BufferU:BufferLI:BufferLV**

The optional *SAP\_Buffer\_Vert* line can be used to override the default vertical buffers applied to the area for SAP processing. The first value is used above the minimum RVSM level, the others at or below it.

- BufferU High level buffer (feet, integer value, range 0-9999)
- BufferLI Low level buffer for IFR flights (feet, integer value, range 0-9999)
- BufferLV Low level buffer for VFR flights (feet, integer value, range 0-9999)

## *Coordinate*

### **Lat Lon**

#### **COORD:Lat:Lon**

Each area definition must have at least three *coordinate* lines (three points being the minimum required to create a closed area). There is practically no upper limit for the number of coordinate points, but as the required calculations increase proportionally to the number of points, it's best to keep the areas simple. The first format option has a pair of latitude and longitude values with one or more spaces between them. There may also be one or more spaces in the beginning of the line before the latitude value so it should be relatively easy to create areas from the REGIONS part of a sector file. The second option uses the same format as the ESE file. The coordinates may be either in decimal degrees or in the sector file format.

## 2.3 TopSkyAreasManualAct.txt

This file can be used to temporarily clear all defined activation rules and/or add new ones to the areas defined in the TopSkyAreas.txt file. If the file is present, the plugin will display a message box to the user as a reminder about the non-standard configuration.

To clear all existing rules, use the following syntax:

### **CLEAR\_ALL**

#### **AreaName:CLEAR**

- AreaName                      Area name as specified in TopSkyAreas.txt

“CLEAR\_ALL” clears all rules from all areas, whereas the second option allows to clear rules for single areas. Note that if you are adding new rules as well, they must be defined after this line, otherwise they’ll get wiped out as well. The “CLEAR\_ALL” keyword should therefore generally be the first line in the file when it’s used, and the “AreaName:CLEAR” the first line that references the area in question.

To set the area to manual activation mode, use the following syntax:

### **MANUAL\_ALL**

#### **AreaName:MANUAL**

- AreaName                      Area name as specified in TopSkyAreas.txt

“MANUAL\_ALL” sets all areas to manual mode but retains all defined activation schedules, whereas the second option allows setting the mode for single areas. If any activation rules are added after these lines, the areas will revert back to automatic mode. Therefore the “MANUAL\_ALL” keyword should generally be the last line in the file if it’s used, and “AreaName:MANUAL” the last line that references the area in question.

When going through the file, if an area is set to manual mode and its last defined activation period is a single period (i.e. SchedWeekdays is “0”), the area is set to activate according to that period.

To add new rules, all the “ACTIVE” line types from TopSkyAreas.txt can be used. The only difference in the syntax is that in the place of the “ACTIVE” keyword, the area name must be used:

#### **AreaName:1**

**AreaName:SchedStartDate:SchedEndDate:SchedWeekdays:StartTime:EndTime[:Lower:Upper:UserText]**

**AreaName:NOTAM:Icao:Text**

**AreaName:RWY:ARR:ArrRwyList:DEP:DepRwyList**

- AreaName                      Area name as specified in TopSkyAreas.txt
- Other fields                      See TopSkyAreas.txt “ACTIVE” line types

## 2.4 TopSkyCPDLC.txt

This file contains data for the CPDLC system. It defines the departure clearance format(s) and pre-defined CPDLC free text items that are available and connects controller position IDs to CPDLC logins.

### *Departure clearance format definition*

```
DCL:EFHK:SID:CLRD TO <ades> OFF <drwy> VIA <sid> DCT <npt> SQUAWK <assr> <nfreq> <qnh> <rmk>
```

#### **Type:Adep:Subtype:Text**

- Type                      One of the following:
  - DCL              DCL type message (other than US)
  - DCL\_US        CPDLC DCL type message (used in USA)
  - PDC              PDC type message
- Adep                      Comma-separated list of departure airport ICAO codes  
                              (or "\*" to match any airport)
- Subtype                   One of the following:
  - AHDG            match when aircraft has an assigned heading
  - SID              match when aircraft has a SID
  - AHDG+SID       match when the aircraft has both
  - \*                always match
- Text                      The clearance message that is sent

Note that if both "AHDG" and "AHDG+SID" formats are defined for an airport with automatically assigned SIDs, the "AHDG" format will only be used when a SID isn't automatically assigned (an automatically assigned SID cannot be removed without changing the flight plan to prevent the automatic assignment).

As an airport typically provides either DCL or PDC (or PDC and DCL\_US) type clearances, both DCL and PDC types should not be defined here for any airport. They are requested using the same message format, and if both clearance types are defined, the plugin won't know which one to send. Having both PDC and DCL\_US for one airport is OK as they are requested with different messages.

The first definition line that matches both the *Adep* and *Subtype* will be used. The *Text* can contain some data fields that are automatically filled by the plugin. Some are highlighted if the pilot's CPDLC software supports it. The "<" character may not be used in the message definition for anything other than entering the defined data fields.

Data fields that are not highlighted:

- <year4>                  current UTC year
- <year2>                  last two digits of the current UTC year
- <month3>                current UTC month (three characters: "JAN", "FEB", etc.)
- <month2>                current UTC month (two digits)
- <day>                    current UTC day (two digits)
- <hour>                   hours of the current UTC time (two digits)
- <min>                    minutes of the current UTC time (two digits)

- <sec> seconds of the current UTC time (two digits)
- <wday3> current UTC weekday (three characters: "MON", "TUE", etc.)
- <number> running number of sent PDCs (three digits)
- <adep2> departure airport ICAO code

Data fields that are highlighted in DCL and DCL\_US clearances:

- <callsign> callsign of the aircraft
- <adep> departure airport ICAO code
- <ades> destination airport ICAO code
- <drwy> departure runway identifier
- <ahdg> assigned heading (three digits)
- <ahdg/trk> assigned heading ("HDG xxx") or track ("TRACK xxx")
- <sid> SID identifier
- <npt> next route point
- <sid+npt> SID identifier, a blank space and the next route point
- <freq> controller primary frequency
- <nfreq> "NEXT FREQ " + next frequency
- <rfreq> "REPORT READY ON " + next frequency
- <dfreq> "DEPARTURE FREQ " + departure frequency
- <cfl> cleared altitude/FL
- <rfl> requested altitude/FL
- <copx> exit coordination point
- <assr> assigned transponder code (four digits)
- <eobt> estimated off-blocks time (four digits)
- <aobt> actual off-blocks time (four digits)
- <aobtMDI> same as <aobt>, except displays "MDI" if equal to estimated off-blocks time
- <startup> either "START-UP APPROVED" or "REPORT READY ON " + next frequency
- <qnh> "QNH " + the QNH value (3-4 digits), nothing printed if unavailable
- <qnh4> same as <qnh> but always with four digits
- <qnhQ> "Q" + the QNH value (four digits), nothing printed if unavailable
- <rmk> remarks text

Highlighted items should have some text between them to work correctly unless the second item contains static text before the actual highlighted value. In cases where no text is desired between items, the underscore character "\_" works well and shows as a blank space in the current pilot clients. Creating the clearance formats usually requires some testing to get them right.

A header part will be automatically added in front of "DCL" type, by default it is

```
<adep2> PDC <number> . . . . CLD <hour><min> <year2><month2><day> <adep2> PDC
<number> <callsign>
```

The whole clearance including the header part is sent to the aircraft, but the header is not displayed in the plugin's message windows.

### *CPDLC free text definition*

#### **FREETEXT:ReplyType:Text**

- ReplyType                Reply type expected from the aircraft. One of the following:
  - WU                “WILCO” or “UNABLE”
  - AN                “AFFIRM” or “NEGATIVE”
  - R                “ROGER”
  - NE                no reply expected
- Text                    Message text that is sent

### *CPDLC login definition*

#### **LOGIN:Login:RadioCall:ControllerID**

- Login                    CPDLC login to match with the controller ID
- RadioCall                RTF callsign used by the controller
- ControllerID            Controller ID to match with the CPDLC login

As there is no way to match a controller to a CPDLC login automatically, this list is used. It follows that it's very important to use only agreed CPDLC logins for the CPDLC connection handovers to work properly.

## 2.5 TopSkyCallsigns.txt & TopSkyCallsignsLocal.txt

These files contain the radiotelephony callsigns to be displayed in the track labels. By default, if an identifier is found in both files, the information in the “TopSkyCallsigns.txt” file is used. The following example line shows the syntax:

AAB	Abelag Aviation	ABG	Callsign definition
-----	-----------------	-----	---------------------

The format of the files is the same as in the “ICAO\_Airlines.txt” file provided with EuroScope. Only one callsign must be defined per line. If the “TopSkyCallsigns.txt” file is not found, the plugin checks if “ICAO\_Airlines.txt” is found in the plugin folder, and if so, uses the information in that file.

### *Callsign definition*

**ThreeLetterID<tab>OperatorName<tab>Callsign**

- ThreeLetterID                      Three-letter designator in the flightplan
- OperatorName                      Name of the operator (not used, but must contain something)
- Callsign                              Radiotelephony callsign

As an alternative to having the callsign data, this file can contain the location of another file that has the data. In that case, this file should have only one line, containing the data file location. The location can be defined as absolute or relative. Relative locations starting with “.” or “..” are relative to the plugin folder. The data in the pointed file must be in the above format, and the pointed file must contain the callsign data, not a path to another file.

## 2.6 TopSkyMaps.txt & TopSkyMapsLocal.txt

These files contain the definitions for the maps in the Maps Window. The difference is that the maps in the “Local” file are automatically assigned to a folder named “LMAPS” and any map folder names in that file are disregarded. The following example area is used to show the syntax (optional lines in grey color):

COLORDEF:Approach:255:255:255	Color definition
SYMBOLDEF:Fix1	Symbol definition
MOVETO:-4:3	Moveto
LINETO:0:-4	Lineto
LINETO:4:3	Lineto
LINETO:-4:3	Lineto
// EFKU VOR app rwy 15	Comment
MAP:EFKU VOR 15	Name
FOLDER:IAP	Folder
COLOR:Approach	Color
LINE:N063.01.03.067:E027.47.04.942:N063.11.03.016:E027.39.18.377	Line
LINE:N063.01.22.882:E027.49.09.775:N063.00.43.220:E027.45.00.157	Line
LINE:N063.06.39.890:E027.45.04.332:N063.06.00.228:E027.40.53.958	Line
STYLE:Dot:1	Line style
LINE:N063.01.03.067:E027.47.04.942:N062.58.47.170:E027.48.49.840	Line
LINE:N062.58.47.170:E027.48.49.840:N062.54.47.000:E027.53.04.000	Line
SYMBOL:Fix1:N063.06.19.000:E027.43.04.000:ROVSU:20:-10	Symbol + label
SYMBOL:Fix1:N062.54.47.000:E027.53.04.000:VEKEM:20:-10	Symbol + label

The mandatory items for each map are a name, a folder it belongs to in the Maps Window, and a color with which to draw the items.

### *Color definition*

#### **COLORDEF:ColorName:R:G:B**

Every color used in the maps (with some exceptions listed in the COLOR line specification) must be defined using one of these lines.

- ColorName      Color name to be used in the Color lines (text string)
- R                Color's red component value (0-255)
- G                Color's green component value (0-255)
- B                Color's blue component value (0-255)

## Symbol definition

### **SYMBOLDEF:SymbolName**

The first line for each symbol definition must be a *symbol definition* line.

- SymbolName              Symbol name to use for this symbol in the Symbol lines (text string)

The symbol itself can consist of various elements, drawn by the following lines. The X and Y coordinates are relative to the symbol centerpoint, with the X axis having increasing values to the right and the Y axis having increasing values to the down direction. The commands are the same as in the EuroScope Symbology dialog with the exception of the possibility to draw elliptical arcs and the “:” separating the values here so the ES dialog can be used in most cases to test the results.

### **MOVETO:X:Y**

Sets the starting point for the next LINETO command

- X                          Number of pixels from the symbol centerpoint in the left(-)-right(+) direction
- Y                          Number of pixels from the symbol centerpoint in the up(-)-down(+) direction

### **LINETO:X:Y**

Draws a straight line from the previous position

- X                          Number of pixels from the symbol centerpoint in the left(-)-right(+) direction
- Y                          Number of pixels from the symbol centerpoint in the up(-)-down(+) direction

### **SETPIXEL:X:Y**

Paints the selected pixel

- X                          Number of pixels from the symbol centerpoint in the left(-)-right(+) direction
- Y                          Number of pixels from the symbol centerpoint in the up(-)-down(+) direction

### **ARC:X:Y:Radius:StartAngle:EndAngle**

### **ARC:X:Y:Radius<sub>X</sub>:Radius<sub>Y</sub>:StartAngle:EndAngle**

Draws a part of a circle

- X                          Centerpoint offset from the symbol centerpoint in the left(-)-right(+) direction
- Y                          Centerpoint offset from the symbol centerpoint in the up(-)-down(+) direction
- Radius                    Arc radius in pixels (to make a circular arc)



- Radius<sub>x</sub>                      Arc radius in relation to the X axis in pixels (to make an elliptical arc)
- Radius<sub>y</sub>                      Arc radius in relation to the Y axis in pixels (to make an elliptical arc)
- StartAngle                    Arc starting angle (integer degrees, 0 degrees is at positive X-axis, increasing counterclockwise)
- EndAngle                     Arc ending angle (integer degrees, 0 degrees is at positive X-axis, increasing counterclockwise)

**FILLARC:X:Y:Radius:StartAngle:EndAngle**

**FILLARC:X:Y:Radius<sub>x</sub>:Radius<sub>y</sub>:StartAngle:EndAngle**

Otherwise the same as ARC above but the result is filled

**POLYGON:X<sub>1</sub>:Y<sub>1</sub>: X<sub>2</sub>:Y<sub>2</sub>:...: X<sub>n</sub>:Y<sub>n</sub>**

Draws a filled polygon with n vertices

## *Name*

**MAP:MapName**

The first line for each map definition must be a name line. This will identify the map in the Maps Window.

- MapName                      Map name to identify it in the Maps window (text string)

## *Folder*

**FOLDER:FolderName**

Every map must belong to a folder. There is practically no limit to how many maps a single folder can contain. The following folder names get special treatment: "ARTCC HIGH", "ARTCC", "ARTCC LOW", "GEO", "SID", "STAR" and "FREE TEXT". Maps assigned to those folders will not be shown in the Maps Window. Instead, if sector file items with the same names are found in the active sector file (i.e. a SID named ABC1A exists in the sector file and a map named "ABC1A" in folder "SID"), the map is assigned as additional data for that sector file item and activated/deactivated together with it.

- FolderName                      Folder name to identify it in the Maps window (text string)

*Note: the folder name may not start with an empty space character, and may not contain the backslash ("\\") character. Folder names "AUTO" and "LMAPS" are reserved for plugin use.*

## Zoom

### **ZOOM:ZoomLevel**

A whole map or parts of it can be hidden based on the current zoom level. With a positive value set, when the radar screen is zoomed out so that there are less than the specified number of pixels per nautical mile, the lines of the map definition after this line are not read. There can be more than one *zoom* line in one map to hide parts of the map at different zoom levels. When the set value is negative, the following lines are not read when the radar screen is zoomed in more than the set value.

- ZoomLevel                      Radar screen zoom level (pixels per nautical mile, decimal value)

*Note: when there is more than one zoom line in a map, their order is important (for example "ZOOM:5" has to be before "ZOOM:10" to have any effect as with zoom below 10 pix/nm the "ZOOM:5" line will never be read if it's after the "ZOOM:10" line...)*

## Screen-specific

### **SCREEN-SPECIFIC**

When a map contains this line, each radar screen will toggle the map's visibility independently instead of affecting all other radar screens. Setting this is only possible for maps with no *active* lines.

## ASRdata

### **ASRDATA:ItemList**

Display of certain parts of the map can be decided based on information entered in the currently active ASR file. This line defines the "type" of the following lines in the map until the next *ASRdata* line. It is then checked against any definitions in the ASR file. "ASRDATA:\*" reads the following lines regardless of ASR definitions.

- ItemList                      Comma-separated list of items

The formats to define map data in the ASR file are as follows:

**PLUGIN:TopSky plugin:ShowMapData:<ItemList>**

**PLUGIN:TopSky plugin:HideMapData:<ItemList>**

<ItemList> is a comma-separated list of items. ShowMapData draws only those parts of plugin maps affected by *ASRdata* lines that contain at least one of the defined items, HideMapData hides such parts.

## Active

### ACTIVE:1

**ACTIVE:SchedStartDate:SchedEndDate:SchedWeekdays:StartTime:EndTime**

**ACTIVE:NOTAM:Icao:Text**

**ACTIVE:RWY:ARR:ArrRwyList:DEP:DepRwyList**

The *active* line is optional. If there are no *active* lines, the map will not be automatically activated. A map can contain more than one *active* line; the plugin will check all of them to set the activation status. The map will be placed in the "AUTO" folder, a folder name cannot be specified for automatically activating maps.

The first option will activate the map automatically without any time limits when the plugin is loaded. Note that this option cannot be used together with other *active* lines as it would override any other schedule.

The second option can be used to set activation schedules.

- SchedStartDate      First day to activate the map
  - month and day in the format MMDD (for recurring periods every year)
  - year, month and day in the format YYYYMMDD (for a single period)
- SchedEndDate      Last day to activate the map, formats as above
- SchedWeekdays    Days of the week to activate the map
  - list of numbers representing the days to activate the map, for example "145" means the map will activate on Mondays, Thursdays and Fridays
  - "0" (zero) to activate the map continuously from StartTime on SchedStartDate to EndTime on SchedEndDate
- StartTime          Time to activate the map (UTC time in the format HHMM)
- EndTime            Time to deactivate the map (UTC time in the format HHMM)

*Note: SchedEndDate and SchedWeekdays only limit the activation of the map. If the activation time extends past midnight, the map stays active until EndTime on the following day.*

The third option activates the map based on NOTAM information. The plugin then attempts to set the activation schedule based on the other contents of received NOTAMs.

- Icao                  ICAO location indicator that publishes activation NOTAMs for the map
- Text                  Text to search for in the NOTAM

The fourth option activates the map based on active runways. If all the specified runways are active, the map is activated. If even one of them is not, the map will be deactivated. The runway identifiers must be in the format "<4-letter ICAO code><runwayID>", for example "EFHK15".

- ArrRwyList          Comma-separated list of runways. Enter "\*" to allow any runway.
- DepRwyList          Comma-separated list of runways. Enter "\*" to allow any runway.

## Color

### **COLOR:ColorName[:FillColorName]**

Every map must have at least one *color* line. It sets the color to be used to draw the subsequent drawings. Each line, symbol, etc. within a map can be drawn with a different color simply by including a new *color* line when a color change is required. If the FillColorName is not specified, it is set to the same color as ColorName. All used color names (with some exceptions listed below) must be defined in the file using a *colordef* line.

- ColorName           Color to be used for drawing lines and texts
- FillColorName       Color to be used for filling the FILLARC and POLYGON objects

Some plugin colors can be used as map colors without having to define them in a *colordef* line (however if the color is defined in a *colordef* line, that definition will override the plugin color definition). The available color names are:

- Active\_Map
- Active\_Map\_Type\_1 ... Active\_Map\_Type\_20
- Active\_RD\_Infill\_Map
- Active\_RD\_Map
- Active\_Sector
- Active\_Text\_Map
- East\_NAT\_Map
- Inactive\_Sector
- Map\_1 ... Map\_4
- Map\_Border
- Map\_Hotspot
- Map\_Info
- Map\_Land
- Map\_Symbol
- Preactive\_Map
- Preactive\_Text\_Map
- Predisplay\_Map
- Rwy\_App\_Line\_Inuse
- Rwy\_App\_Line\_Not\_Inuse
- West\_NAT\_Map

## Style

### STYLE:StyleName[:Width]

The *style* line defines the line type for any subsequent *line* items within this map. It is not mandatory, a Solid type line with width 1 pixel will be drawn by default. As with the *color* line, a single map may contain any required number of *style* lines to draw different line styles within the same map. The width is only necessary for Solid type lines (it has no effect on the other types). If a width is not defined, a 1-pixel width is drawn by default.

- StyleName                      Style to be used (Solid, Dash, Dot, DashDot or DashDotDot)
- Width                              Width of line (pixels)

## Line

### LINE:Lat<sub>1</sub>:Lon<sub>1</sub>:Lat<sub>2</sub>:Lon<sub>2</sub>

### LINE:StartPointName:EndPointName

Draws a line from one point to another. Uses the previously defined line style (or solid line with 1-pixel width if no style defined).

- Lat<sub>1</sub>                              Latitude of starting point (decimal degrees or sector file format)
- Lon<sub>1</sub>                              Longitude of starting point (decimal degrees or sector file format)
- Lat<sub>2</sub>                              Latitude of end point (decimal degrees or sector file format)
- Lon<sub>2</sub>                              Longitude of end point (decimal degrees or sector file format)
- StartPointName              Fix, VOR, NDB, airport code or runway (must be found in the active sector file)
- EndPointName                Fix, VOR, NDB, airport code or runway (must be found in the active sector file)

*Note: the syntax to define a runway threshold as a StartPointName or an EndPointName is the 4-letter ICAO airport designator followed by a forward slash and the runway identifier.*

## FontSize

### FONTSIZE:Type:Size

### FONTSIZE:0

Each new map starts out with the default font size. It can be modified using the *FontSize* line. All texts after the line in that map use the new size. "FONTSIZE:0" sets the size back to the default value.

- Type                              Type of change
  - "=" sets a new size
  - "-" reduces the size from the default by the given amount
  - "+" increases the size from the default by the given amount
  - "\*" multiplies the size of the default by the given amount
- Size                                New font size (1-99)

The resulting font size is limited to values between 1 and 99.

## FontStyle

**FONTSTYLE:Weight,Italic,Underline,Strikethrough**

**FONTSTYLE:0**

Each new map starts out with the default font style. It can be modified using the *FontStyle* line. All texts after the line in that map use the new style. "FONTSTYLE:0" sets the style back to the default settings.

- Weight                      Font weight (0-1000)
  - some example values are 0=default weight, 400=normal, 700=bold
- Italic                      Italic (1=yes, 0=no)
- Underline                  Underline (1=yes, 0=no)
- Strikethrough              Strikethrough (1=yes, 0=no)

## Symbol

**SYMBOL:SymbolName:Lat:Lon[:Label:OffsetX:OffsetY]**

**SYMBOL:SymbolName:PointName[:Label:OffsetX:OffsetY]**

Draws a predefined symbol on the screen. Optionally, can display a text label with the label's centerpoint offset a given number of pixels from the symbol centerpoint.

- SymbolName                Name of symbol
- Lat                        Latitude of symbol centerpoint (decimal degrees or sector file format)
- Lon                        Longitude of symbol centerpoint (decimal degrees or sector file format)
- PointName                Fix, VOR, NDB, airport code or runway (must be found in the active sector file)
- Label                     Text label (text string)
- OffsetX                    Number of pixels to offset the label centerpoint in the left(-)-right(+) direction
- OffsetY                    Number of pixels to offset the label centerpoint in the up(-)-down(+) direction

*Note: the syntax to define a runway threshold as a PointName is the 4-letter ICAO airport designator followed by a forward slash and the runway identifier.*

## Text

**TEXT:Lat:Lon:Label[:OffsetX:OffsetY]**

**TEXT:PointName:Label[:OffsetX:OffsetY]**

Draws a text label.

- Lat Latitude of label centerpoint (decimal degrees or sector file format)
- Lon Longitude of label centerpoint (decimal degrees or sector file format)
- PointName Fix, VOR, NDB, airport code or runway (must be found in the active sector file)
- Label Text label (text string)
- OffsetX Number of pixels to offset the label centerpoint in the left(-)-right(+) direction
- OffsetY Number of pixels to offset the label centerpoint in the up(-)-down(+) direction

*Note: the syntax to define a runway threshold as a PointName is the 4-letter ICAO airport designator followed by a forward slash and the runway identifier.*

## Coordinate

**COORD:Lat:Lon**

**COORD:PointName**

Defines line or polygon vertex coordinates to be used later with *CoordLine* or *CoordPoly* lines.

- Lat Latitude (decimal degrees or sector file format)
- Lon Longitude (decimal degrees or sector file format)
- PointName Fix, VOR, NDB, airport code or runway (must be found in the active sector file)

*Note: the syntax to define a runway threshold as a PointName is the 4-letter ICAO airport designator followed by a forward slash and the runway identifier.*

## CoordLine

**COORDLINE**

Draws a sequence of lines from a set of coordinates having been defined by preceding *coordinate* lines. The lines use the previously defined line style (or solid line with 1-pixel width if no style defined).

## CoordPoly

**COORDPOLY:FillPattern**

Draws a polygon from a set of coordinates having been defined by preceding *coordinate* lines. The edge line is drawn with the previously defined line style (or solid line with 1-pixel width if no style defined).

- FillPattern Polygon fill pattern
  - 0 no fill
  - 5, 10, 20, 25, 30, 40, 50, 60, 70, 75, 80, 90, 100 percentage to fill

## 2.7 TopSkyMSAW.txt

The file is read one line at a time and the first line that contains the aircraft position returns the minimum safe altitude, so put specific small area lines at the top and large general areas to the end of the file. Be careful to always use the correct syntax to achieve any results and use the point (.) as the decimal separator. For backward compatibility, also the comma (,) is accepted to separate the fields. Only define one area per line. There are five types of area definitions that are accepted:

### *Lat/Lon box area*

**A:Lat<sub>min</sub>:Lat<sub>max</sub>:Lon<sub>min</sub>:Lon<sub>max</sub>:MSA**

An area bounded by the minimum and maximum latitude and longitude values

- Lat<sub>min</sub> Minimum Latitude (decimal degrees or sector file format)
- Lat<sub>max</sub> Maximum Latitude (decimal degrees or sector file format)
- Lon<sub>min</sub> Minimum Longitude (decimal degrees or sector file format)
- Lon<sub>max</sub> Maximum Longitude (decimal degrees or sector file format)
- MSA Minimum Safe Altitude within the area (feet, integer value)

### *Circle*

**C:Lat:Lon:R:MSA**

A circle of radius r with center point at (Lat,Lon)

- Lat Circle center point latitude (decimal degrees or sector file format)
- Lon Circle center point longitude (decimal degrees or sector file format)
- R Radius of circle (nautical miles, decimal value)
- MSA Minimum Safe Altitude within the circle (feet, integer value)

### *Lat/Lon box area list*

**L:Lat<sub>min</sub>:Lon<sub>min</sub>:ΔLat:ΔLon:N:MSA<sub>1</sub>:MSA<sub>2</sub>:...:MSA<sub>n</sub>**

A series of latitude-longitude bounded boxes. The boxes are in an east-west direction, with the first box being the westernmost.

- Lat<sub>min</sub> Latitude of the south edge of the boxes (decimal degrees or sector file format)
- Lon<sub>min</sub> Longitude of the west edge of first box (decimal degrees or sector file format)
- ΔLat Latitude size of one box (decimal degrees)
- ΔLon Longitude size of one box (decimal degrees)
- N Number of boxes
- MSA<sub>1</sub>-MSA<sub>n</sub> Minimum Safe Altitudes of the boxes (feet, integer values, must be n values)



## Polygon

**P:N:Lat<sub>1</sub>:Lon<sub>1</sub>:Lat<sub>2</sub>:Lon<sub>2</sub>:...:Lat<sub>n</sub>:Lon<sub>n</sub>:MSA**

A polygon with  $n$  vertices at given latitude-longitude points

- N                      Number of vertices
- Lat<sub>1</sub>                  Latitude of vertex 1 (decimal degrees or sector file format)
- Lon<sub>1</sub>                  Longitude of vertex 1 (decimal degrees or sector file format)
- Lat<sub>2</sub>                  Latitude of vertex 2 (decimal degrees or sector file format)
- Lon<sub>2</sub>                  Longitude of vertex 2 (decimal degrees or sector file format)
- Lat<sub>n</sub>                  Latitude of vertex n (decimal degrees or sector file format)
- Lon<sub>n</sub>                  Longitude of vertex n (decimal degrees or sector file format)
- MSA                   Minimum Safe Altitude within the polygon (feet, integer value)

## Sector

**S:Lat:Lon:TRdl<sub>1</sub>:TRdl<sub>2</sub>:R<sub>min</sub>:R<sub>max</sub>:MSA**

An area defined as being between two true bearings from a point (Lat,Lon) - clockwise direction from Rdl<sub>1</sub> to Rdl<sub>2</sub> - and between distances R<sub>min</sub> and R<sub>max</sub> from the point

- Lat                      Point latitude (decimal degrees or sector file format)
- Lon                      Point longitude (decimal degrees or sector file format)
- TRdl<sub>1</sub>                  Bearing 1 (degrees true, decimal value)
- TRdl<sub>2</sub>                  Bearing 2 (degrees true, decimal value)
- R<sub>min</sub>                   Minimum distance from point (nautical miles, decimal value)
- R<sub>max</sub>                   Maximum distance from point (nautical miles, decimal value)
- MSA                      Minimum Safe Altitude within the sector (feet, integer value)

## 2.8 TopSkyRadars.txt

The file contains the primary radar station definitions to be used for displaying raw video radar data. The following example shows the syntax (optional lines in grey color):

// Helsinki PSR	Comment
RADAR:Helsinki	Radar
POSITIONS:EFHF:EFHK:EFES	Positions
LOCATION:N060.18.56.400:E024.57.54.400	Location
ALTITUDE:335	Altitude
BEAMWIDTH:1.4	Beamwidth
PULSEWIDTH:1.0	Pulsewidth
MAXANGLE:50	Maxangle
RANGE:0.0:60.0	Range
NOTERRAIN	NoTerrain

### *Radar*

#### **RADAR:RadarName**

Each radar definition must start with a Radar line that defines the radar station name.

- RadarName                  Radar station name (text string)

### *Positions*

#### **POSITIONS:Pos<sub>1</sub>:Pos<sub>2</sub>:...**

Defines the list of controller positions that use the radar station. Only one radar station can be active so the first station in the file that contains a match will be used. The logic compares the positions against the login callsign from the beginning of the string ("EF" will be a match for either "EFES\_CTR" or "EFHK\_TWR" but not for "SAEF\_APP")

- Pos<sub>x</sub>                          Position login callsign (text string, full callsign or first x letters)

### *Location*

#### **LOCATION:Lat:Lon**

The location of the radar antenna.

- Lat                              Latitude value (decimal degrees or sector file format)
- Lon                              Longitude value (decimal degrees or sector file format)

## *Altitude*

### **ALTITUDE:Alt**

The radar antenna altitude above mean sea level. If not specified, a value of 0 is used.

- Alt                      Antenna altitude (feet AMSL, integer value)

## *Beamwidth*

### **BEAMWIDTH:Beamwidth**

Specifies the beamwidth of the radar in degrees. If not specified, a value of 1.5 will be used. The value affects how wide the radar targets will be (twice the beamwidth value).

- Beamwidth              Beamwidth of the radar (degrees, decimal value)

## *Pulsewidth*

### **PULSEWIDTH:Pulsewidth**

Specifies the pulse width of the radar in microseconds. If not specified, a value of 1.0 will be used. The value affects how deep the radar targets will be (approx. 0.08nm/microsecond).

- Pulsewidth              Pulse width of the radar (microseconds, decimal value)

## *Maxangle*

### **MAXANGLE:MaxAngle**

Defines the maximum elevation angle of the radar measured from horizontal level. If not specified, a value of 90 will be used (i.e. coverage all the way up to vertical)

- MaxAngle              Maximum vertical angle of the radar (degrees, integer value)

## *Range*

### **RANGE:MinRange:MaxRange**

Defines the minimum and maximum detection ranges of the radar station. If not specified, a minimum value of 0 and a maximum value of 999999 will be used.

- MinRange              Minimum detection range of the radar (nautical miles, decimal value)
- MaxRange              Maximum detection range of the radar (nautical miles, decimal value)

## *Ceiling*

### **CEILING:Ceiling**

Defines the maximum detection altitude of the radar. If not specified, a value of 999999 will be used.

- Ceiling                      Maximum detection altitude of the radar (feet AMSL, integer value)

## *NoTerrain*

### **NOTERRAIN**

Sets the radar station to disregard ESE file radar data and to have no terrain masking.

When *NoTerrain* or *terrain* lines are not defined, the displayed targets are based on the primary radar coverage defined in the ESE file. For reasonable results, a primary radar should be defined at the same position in the ESE file as the one in this data file.

With the *NoTerrain* definition, all targets within the radar's visibility area will be displayed regardless of the ESE file radar definitions.

## *Terrain*

### **TERRAIN:ARC:StartAngle:EndAngle:Distance:Elevation**

### **TERRAIN:LINE:StartLat:StartLon:EndLat:EndLon:Elevation**

Sets the radar station to disregard ESE file radar data and defines a terrain obstruction for the radar, either as a radial arc or a line segment of constant elevation value. The arc format is more performance-friendly so it's the preferred choice whenever possible.

When one or more *terrain* lines are defined, all targets within the radar's visibility area that are not masked by the defined terrain will be displayed regardless of the ESE file radar definitions.

- StartAngle                      Arc starting angle (degrees true, decimal value)
- EndAngle                        Arc ending angle (degrees true, decimal value)
- Distance                        Arc radius from radar station (nautical miles, decimal value)
- StartLat                        Line starting point latitude (decimal degrees or sector file format)
- StartLon                        Line starting point longitude (decimal degrees or sector file format)
- EndLat                         Line ending point latitude (decimal degrees or sector file format)
- EndLon                         Line ending point longitude (decimal degrees or sector file format)
- Elevation                        Terrain elevation (feet AMSL, decimal value)

## 2.9 TopSkySettings.txt & TopSkySettingsLocal.txt

These two files allow changing the plugin settings. The difference between them is that the settings in the first file are loaded every time, while the settings in the “Local” file are only loaded on user request (by performing a “Sign In”).

The available settings, their default values and acceptable ranges are described in an Excel spreadsheet provided together with this file.

The settings in the files can be either general or login callsign specific. General settings have to be located at the beginning of the file before any login callsign specific ones. Login callsign specific ones are defined by creating sections starting with a line that contains a text string in square brackets above them.

The login callsign specific settings are checked by comparing the text string against the login callsign. If the login callsign contains the text, any settings after that are loaded until a new line with text in square brackets is found which after the check is done again. If not, all the settings in that section are skipped.

Example TopSkySettings.txt file:

Setting1=0	
Setting2=123	
[_CTR]	New section start
Setting2=100	
Setting3=0	
[EFES_]	New section start
Setting2=200	
[ESSA_TWR]	New section start
Setting2=300	
Setting4=0	

The plugin reads the file as follows: first “Setting1” is set to “0” and “Setting2” to “123” regardless of the login callsign. If the login callsign is for example “EFES\_2\_CTR”, the first section (“\_CTR”) is a match, so “Setting2” will be changed to “100” and “Setting3” is set to “0”. The second section (“EFES\_”) is also a match, so “Setting2” is changed once again, this time to “200”. The last section (“ESSA\_TWR”) is not a match so the settings there won’t be applied. So, any one setting can appear in the file more than once, and be set more than once depending on how the file is laid out. As the file is always read in the order it is written, the more specific sections should be at the bottom (like in the example, any \_CTR callsign will get a different “Setting2” value than the other callsigns, but if it happens to be EFES\_CTR or EFES\_<anything here>\_CTR, the value is different from the other \_CTR callsigns).

When the plugin detects that the login callsign has changed, the settings files are automatically reloaded (the “Local” file only if a “Sign In” has been done).

## 2.10 TopSkySSRcodes.txt

The file contains the SSR code range and area definitions to be used when assigning transponder codes via the plugin. The following example shows the syntax (optional lines in grey color):

AREA:EF1	Area
RADIUS:15.0	Radius
N060.00.00.000 E025.00.00.000	Coordinate
// domestic secondary	Comment
RANGE:3201:3277	Range
ADES:EF	ADES
PROTECTION:EF1	Protection
PRIORITY:-1	Priority
IFR	IFR

The plugin's SSR code assignment system checks the flightplan, finds out which code ranges are available for that flightplan, checks for codes already in use, and then assigns one of the available codes. Transponder codes ending with "00" are not assigned.

The example area above designates the code range 3201-3277 available for IFR traffic with destinations starting with "EF" and not entering the area "EF1". In addition, the range is defined to have a priority level of -1, to be used only if there are no available codes found in higher priority ranges.

### Mode S global

#### MODE\_S\_GLOBAL

If this line appears anywhere in the file, it indicates that the identity of mode S capable aircraft will be maintained using mode S information everywhere. See also *Mode S* line to limit the availability to specified area(s).

### Group

#### GROUP:GroupName:Item<sub>1</sub>:Item<sub>2</sub>:Item<sub>3</sub>:...

The *group* line can be used as a shortcut to writing a large number of text entries. It can be used in line types where lists of text strings are used. To use a group in a line, enter "GROUP\_<groupname>" like any other text string. It will be automatically expanded to the list of text strings in the group definition.

- GroupName            Name for the group
- Item<sub>x</sub>                Text strings

*Note: the item separator to be used here is the colon (:), regardless of what's used in the target line type.*

## Area

### AREA:AreaName

Each area definition must start with an *area* line that defines the area name. The area must be defined in the file before it is referred to in a code range definition. The area names are case sensitive.

- AreaName                      Area name to use in the code assignment rules (text string)

## Radius

### RADIUS:Radius

If the area is a circle, it can be defined as a center point and a distance from it. In this case the area definition needs the *radius* line and one *coordinate* line (see below). All other area shapes need to be defined as polygons using three or more *coordinate* lines, and then the *radius* line shall not be used.

- Radius                          Area radius (nautical miles, decimal value)

## Mode S

### MODE\_S

This line indicates that inside this area the identity of mode S capable aircraft will be maintained using mode S information. The mode S conspicuity code (1000) can be assigned by the plugin when the entire planned route is within these areas.

## Coordinate

### Lat Lon

Each area definition must have either at least three *coordinate* lines (three points being the minimum required to create a closed area, a triangle), or one *coordinate* line and a *radius* line (see above). There is practically no upper limit for the number of coordinate points, but as the required calculations increase proportionally to the number of points, it's best to keep the areas simple. The latitude and longitude values must be either in decimal degrees or in the sector file format and there must be one or more spaces between them. There may also be one or more spaces in the beginning of the line before the latitude value.

## *Range*

### **RANGE:StartCode:EndCode**

This line is the only mandatory line for a code range definition and must always be the first line in a definition. It starts the definition by specifying the range of codes in it.

- StartCode                      First code in the range (4 octal digits, 0001-7777)
- EndCode                        Last code in the range (4 octal digits, 0001-7777)

## *Adhoc*

### **ADHOC**

This line causes the codes in this range to be assigned only to flightplans where the departure, destination or both are empty.

## *Priority*

### **PRIORITY:Level**

This line sets the priority level of this range. When more than one range of codes is suitable for a flight, codes are assigned based on the range priority level, with codes from lower priority ranges assigned only when no codes are available in higher priority ranges. The default priority level of a code range is zero.

- Level                            Priority level (-3 to +3, +3 being the highest priority)

## *Mandatory*

### **MANDATORY**

Available for backward compatibility. Sets the priority level to +1.

## *Secondary*

### **SECONDARY**

Available for backward compatibility. Sets the priority level to -1.

## *IFR*

### **IFR**

This line causes the codes in this range to be assigned only to IFR flightplans.



## *VFR*

### **VFR**

This line causes the codes in this range to be assigned only to VFR flightplans.

## *GAT*

### **GAT**

This line causes the codes in this range to be assigned only to GAT (general air traffic) flightplans. By default, a flightplan is categorized as GAT when “RMK/OAT” is not found in the remarks section.

## *OAT*

### **OAT**

This line causes the codes in this range to be assigned only to OAT (operational air traffic) flightplans. By default, a flightplan is categorized as OAT when “RMK/OAT” is found in the remarks section.

## *Direction*

### **DIRECTION:TTrk1:TTrk2**

This line limits the code assignment to flights having a track between the two specified true tracks (clockwise direction from Trk1 to Trk2). If *via* and/or *NotVia* lines are also present in the code range, the track to be checked is the outbound track from the specified point(s). If not, the tracks are checked against the aircraft’s planned track from its present position.

- TTrk1                      Start angle for the track range (degrees true, decimal value)
- TTrk2                      End angle for the track range (degrees true, decimal value)

## *ADEP*

### **ADEP:ICAOcode:ICAOcode:ICAOcode:...**

This line limits the code assignment to flights departing from one of the defined airports. The whole ICAO airport code is not needed; the match can also be done on the first one or more letters, e.g. entering “EF” will match all airports with ICAO designators beginning with “EF”. The *ADEP* line can contain one or more airport codes and one code range definition can also have more than one *ADEP* line if necessary.

- ICAOcode                      Airport ICAO code (complete or partial)

### *NotADEP*

#### **NOTADEP:ICAOcode:ICAOcode:ICAOcode:...**

This line limits the code assignment to flights not departing from any of the defined airports. Otherwise the format and limitations are the same as in the *ADEP* line.

- ICAOcode                  Airport ICAO code (complete or partial)

### *ADES*

#### **ADES:ICAOcode:ICAOcode:ICAOcode:...**

This line limits the code assignment to flights arriving at one of the defined airports. Otherwise the format and limitations are the same as in the *ADEP* line.

- ICAOcode                  Airport ICAO code (complete or partial)

### *NotADES*

#### **NOTADES:ICAOcode:ICAOcode:ICAOcode:...**

This line limits the code assignment to flights not arriving at any of the defined airports. Otherwise the format and limitations are the same as in the *ADEP* line.

- ICAOcode                  Airport ICAO code (complete or partial)

### *Local*

#### **LOCAL:ICAOcode:ICAOcode:ICAOcode:...**

This line limits the code assignment to local flights (ADEP=ADES) from one of the defined airports. The whole ICAO airport code is not needed; the match can also be done on the first one or more letters, e.g. entering “EF” will match all airports with ICAO designators beginning with “EF”. The *local* line can contain one or more airport codes and one code range definition can also have more than one *local* line if necessary. The *local* line cannot be used together with *ADEP* or *ADES* lines.

- ICAOcode                  Airport ICAO code (complete or partial)

### *Via*

#### **VIA:Point:Point:Point:...**

This line limits the code assignment to flights routing via at least one of the defined points. The point can be anywhere along the flightplan. One or more points can be defined in one *via* line and one code range definition can contain more than one *via* line if necessary.

- Point                      Point name (Fix, VOR, NDB or airport)

### *NotVia*

#### **NOTVIA:Point:Point:Point:...**

This line limits the code assignment to flights not routing via any of the defined points. Otherwise the format and limitations are the same as in the *via* line.

- Point                      Point name (Fix, VOR, NDB or airport)

### *AreaVia*

#### **AREAVIA:AreaName:AreaName:AreaName:...**

This line limits the code assignment to flights routing via at least one of the defined areas. The area(s) must have been defined earlier in the data file. One or more areas can be defined in one *AreaVia* line and one code range definition can contain more than one *AreaVia* line if necessary.

- AreaName                      Area name (text string)

### *Protection*

#### **PROTECTION:AreaName:AreaName:AreaName:...**

This line limits the code assignment to flights not routing via any of the defined areas. Otherwise the format and limitations are the same as in the *AreaVia* line.

- AreaName                      Area name (text string)

### *ADEParea*

#### **ADEPAREA:AreaName:AreaName:AreaName:...**

This line limits the code assignment to departing from one of the defined areas. The area(s) must have been defined earlier in the data file. One or more areas can be defined in one *ADEParea* line and one code range definition can contain more than one *ADEParea* line if necessary.

- AreaName                      Area name (text string)

### *NotADEParea*

#### **NOTADEPAREA:AreaName:AreaName:AreaName:...**

This line limits the code assignment to flights not departing from any of the defined areas. Otherwise the format and limitations are the same as in the *ADEParea* line.

- AreaName                      Area name (text string)

## *ADESarea*

### **ADESAREA:AreaName:AreaName:AreaName:...**

This line limits the code assignment to flights arriving in one of the defined areas. The area(s) must have been defined earlier in the data file. One or more areas can be defined in one *ADESarea* line and one code range definition can contain more than one *ADESarea* line if necessary.

- AreaName                                      Area name (text string)

## *NotADESarea*

### **NOTADESAREA:AreaName:AreaName:AreaName:...**

This line limits the code assignment to flights not arriving in any of the defined areas. Otherwise the format and limitations are the same as in the *ADESarea* line.

- AreaName                                      Area name (text string)

## *Unit*

### **UNIT:LoginCallsign:LoginCallsign:LoginCallsign:...**

This line limits the code assignment based on your network login callsign. The whole callsign is not needed; the match can also be done on the first one or more letters, e.g. entering “EFES” will match callsigns beginning with “EFES”. The *unit* line can contain one or more callsigns and one code range definition can also have more than one *unit* line if necessary.

- LoginCallsign                      Current login callsign (complete or partial)

## *NotUnit*

### **NOTUNIT:LoginCallsign:LoginCallsign:LoginCallsign:...**

This line limits the code assignment to network login callsigns other than the specified ones. Otherwise the format and limitations are the same as in the *unit* line.

- LoginCallsign                      Current login callsign (complete or partial)

## *Callsign*

### **CALLSIGN:Callsign:Callsign:Callsign:...**

This line limits the code assignment based on the aircraft's callsign. The whole callsign is not needed; the match can also be done on the first one or more letters, e.g. entering "FIN" will match callsigns beginning with "FIN". The *callsign* line can contain one or more callsigns and one code range definition can also have more than one *callsign* line if necessary.

- Callsign                      Aircraft callsign (complete or partial)

## *NotCallsign*

### **NOTCALLSIGN:Callsign:Callsign:Callsign:...**

This line limits the code assignment to aircraft callsigns other than the specified ones. Otherwise the format and limitations are the same as in the *callsign* line.

- Callsign                      Aircraft callsign (complete or partial)

## *ATYP*

### **ATYP:Atyp:Atyp:Atyp:...**

This line limits the code assignment based on the aircraft's type. The whole type is not needed; the match can also be done on the first one or more letters, e.g. entering "A3" will match types beginning with "A3". The *ATYP* line can contain one or more types and one code range definition can also have more than one *ATYP* line if necessary.

- Atyp                          Aircraft type designator (complete or partial)

## *NotATYP*

### **NOTATYP:Atyp:Atyp:Atyp:...**

This line limits the code assignment to aircraft types other than the specified ones. Otherwise the format and limitations are the same as in the *ATYP* line.

- Atyp                          Aircraft type designator (complete or partial)

## *Descr*

### **DESCR:List**

This line limits the code assignment based on the aircraft's description.

- List                      Allowed description letters. Any combination of the following can be used:
  - L              landplane
  - S              seaplane
  - A              amphibian
  - H              helicopter
  - G              gyrocopter
  - T              tilt-wing aircraft
  - ?              unknown

## *NotDescr*

### **NOTDESCR:List**

This line limits the code assignment to aircraft with descriptions other than the specified ones. Otherwise the format is the same as in the *descr* line.

- List                      Forbidden description letters

## *Remarks*

### **REMARKS:Text:Text:Text:...**

This line limits the code assignment based on the flightplan remarks. When all the specified text strings are found in the remarks section, the line is a match. The *remarks* line can contain one or more text strings and one code range definition can also have more than one *remarks* line if necessary (in this case it is enough that one of the lines is a match for the code range to be used).

- Text                      Text string to look for in the flightplan remarks

## *NotRemarks*

### **NOTREMARKS:Text:Text:Text:...**

Same as above but limits the code assignment to flightplans whose remarks section contains none of the specified text strings. When more than one *NotRemarks* line is used in a code range, the range is used when even one of the lines is a match.

- Text                      Text string to look for in the flightplan remarks

## 2.11 TopSkySTCA.txt

The file contains the definitions for final approach areas where smaller (usually 2.5nm) separation can be used, as well as STCA exclusion areas needed for parallel approaches and departures. The following example is used to show the syntax (optional lines in grey color):

FINALAPP:EFHK:04L	FinalApp
// EFHK 04's	Comment
SOIR:EFHK:04L:04R	SOIR
NOZ1:N060.18.37.790:E024.54.30.240	NOZ 1
NOZ1:N060.19.21.880:E024.53.08.680	NOZ 1
NOZ1:N060.12.36.560:E024.38.23.030	NOZ 1
NOZ1:N060.11.52.460:E024.39.44.310	NOZ 1
NOZ2:N060.18.49.470:E024.55.54.640	NOZ 2
NOZ2:N060.18.05.350:E024.57.16.110	NOZ 2
NOZ2:N060.11.20.250:E024.42.29.640	NOZ 2
NOZ2:N060.12.04.370:E024.41.08.450	NOZ 2

### *FinalApp*

#### **FINALAPP:AirportICAO:RwyID:Range:XTE:Lat:Lon:CourseT**

This line creates a final approach area where smaller separation values for STCA are used.

- AirportICAO      Airport ICAO code
- RwyID            Runway identifier
- Range            Maximum range from the approach end point (decimal value, range 0.0-99.0)
- XTE              Maximum cross-track error from the approach course (decimal value, range 0.0-99.0)
- Lat               Approach end point latitude (decimal degrees or sector file format)
- Lon               Approach end point longitude (decimal degrees or sector file format)
- CourseT          Approach course (degrees true, decimal value)

Only the ICAO code and RwyID are mandatory to be defined, so for example “FINALAPP:EFHK:04L” creates an approach area for that runway with default values. The default values for the other items are:

- Range            10.0 nm
- XTE              0.5 nm
- Lat, Lon          Runway threshold coordinates from the active sector file
- CourseT          Runway true bearing calculated from the threshold coordinates

Note that if one or more of the optional values needs to be changed, the line must include all the values up to the last changed value.

An STCA alert between two aircraft will be inhibited when all of the following are true:

- Both aircraft are planned to land on the runway in question
- Both aircraft are inside the same approach area
- Both aircraft have ground tracks within 10 degrees of the approach course
- The aircraft closer to the runway is not wake turbulence category HEAVY or SUPER, or a B757
- The aircraft further from the runway is in the same or higher wake turbulence category

## SOIR

### **SOIR:AirportICAO:RwyID1:RwyID2:Range:WidthIn:WidthOut**

This line starts the definition.

- |               |   |
|---------------|---|
| - AirportICAO | Airport ICAO code   |
| - RwyID1      | Identifier of the runway on the left side in the direction of flight            |
| - RwyID2      | Identifier of the runway on the right side in the direction of flight           |
| - Range       | Length of the NOZ (nm, decimal value, range 0.0-99.0)                           |
| - WidthIn     | Width of the NOZ toward the other runway (nm, decimal value, range 0.0-99.0)    |
| - WidthOut    | Width of the NOZ away from the other runway (nm, decimal value, range 0.0-99.0) |

Only the ICAO code and runway identifiers are mandatory to be defined. The default values for the other items are:

- |                      |         |
|----------------------|---------|
| - Range              | 10.0 nm |
| - WidthIn / WidthOut | 0.3 nm  |

Note that if one or more of the optional values needs to be changed, the line must include all the values up to the last changed value. A custom polygon can also be defined as the NOZ, see the NOZ1 and NOZ2 lines.

For arrivals, when one aircraft is within one NOZ and another within the other, and both aircraft have ground tracks within 10 degrees of the approach tracks for their runways, an STCA warning between those aircraft will be inhibited. For departures, the conditions are the same except that the 10-degree track error limit is only valid toward the other runway. Both runways also need to be set as active for arrival (for an arrival setup) or departure (for a departure setup) in the EuroScope runway setup. Both arrival and departure setups can be active simultaneously.

If the airport has more than two runways where simultaneous approaches are performed, each possible pair must be defined separately. For example, if an airport can do simultaneous approaches on any combination of runways 01L, 01C and 01R, three definitions are required (one for 01L/01C, one for 01L/01R and one for 01C/01R).

## Departure

### **DEPARTURE**

This optional line defines the setup as a departure setup. If not present, the setup is an arrival setup.



### *Approach course 1 and 2*

**CRS1:CourseT**

**CRS2:CourseT**

This optional line is used to define the approach course if it is different from the runway bearing. For it to have any effect, the setup must be an arrival setup and the NOZ for the corresponding runway must also be defined manually (see below).

- CourseT                      Approach course for the runway (degrees true, decimal value)

### *NOZ 1 and 2*

**NOZ1:Lat:Lon**

**NOZ2:Lat:Lon**

These optional lines are used to define vertices for custom polygons to override the default NOZ areas for the runways.

- Lat                              Point latitude (decimal degrees or sector file format)
- Lon                              Point longitude (decimal degrees or sector file format)

### *NTZ*

**NTZ:Lat:Lon**

This optional line is used to define vertices for a custom polygon to override the default NTZ area. It may only be used in an arrival setup. The NTZ is used to extend STCA alerting below its normal lower limit altitude when an aircraft is inside the NTZ.

- Lat                              Point latitude (decimal degrees or sector file format)
- Lon                              Point longitude (decimal degrees or sector file format)

## 2.12 TopSkySymbols.txt











This file makes it possible to change the default symbols drawn by the plugin. The following example symbol shows the syntax:

// distance marker	Comment
SYMBOL:MARKER	Type
ELLIPSE:0:0:2	Definition

### Symbol definition

#### SYMBOL:SymbolType

The first line for each symbol definition must be a type line.

- SymbolType      Symbol type (one of the following):
  - AIRPORT      Airport symbol
  - HOTSPOT      Area hotspot symbol
  - FIX            Fix symbol
  - NDB           NDB symbol
  - VOR           VOR symbol
  - MARKER      Runway approach line distance marker symbol
  - RPSH          History dot symbol
  - RPSC          Coasting track symbol
  - RPSF          FPASD track symbol
  - RPS1          Radar position symbol for PSR tracks
  - RPS2           Radar position symbol for SSR(A) tracks
  -  Uncontrolled SSR and SSR+PSR tracks
  - RPS2I         Radar position symbol for SSR(A) tracks with ident
  -  Uncontrolled SSR and SSR+PSR tracks with ident
  - RPS3         Radar position symbol for PSR+SSR(A) tracks
  -  Controlled SSR and SSR+PSR tracks
  - RPS3I         Radar position symbol for PSR+SSR(A) tracks with ident
  -  Controlled SSR and SSR+PSR tracks with ident
  - RPS4         Radar position symbol for SSR(S) tracks
  - RPS4I         Radar position symbol for SSR(S) tracks with ident

The symbol itself can consist of various elements, drawn by the following lines. The X and Y coordinates are relative to the symbol centerpoint, with the X axis having increasing values to the right and the Y axis having increasing values to the down direction. The commands are the same as in the EuroScope Symbology dialog with the exception of the possibility to draw elliptical arcs and the “:” separating the values here so the ES dialog can be used in most cases to test the results.

## **MOVETO:X:Y**

Sets the starting point for the next LINETO command

- X                      Number of pixels from the symbol centerpoint in the left(-)-right(+) direction
- Y                      Number of pixels from the symbol centerpoint in the up(-)-down(+) direction

## **LINETO:X:Y**

Draws a straight line from the previous position

- X                      Number of pixels from the symbol centerpoint in the left(-)-right(+) direction
- Y                      Number of pixels from the symbol centerpoint in the up(-)-down(+) direction

## **SETPIXEL:X:Y**

Paints the selected pixel

- X                      Number of pixels from the symbol centerpoint in the left(-)-right(+) direction
- Y                      Number of pixels from the symbol centerpoint in the up(-)-down(+) direction

## **ARC:X:Y:Radius:StartAngle:EndAngle**

### **ARC:X:Y:Radius<sub>X</sub>:Radius<sub>Y</sub>:StartAngle:EndAngle**

Draws a part of a circle or ellipse

- X                      Centerpoint offset from the symbol centerpoint in the left(-)-right(+) direction
- Y                      Centerpoint offset from the symbol centerpoint in the up(-)-down(+) direction
- Radius                Arc radius in pixels (to make a circular arc)
- Radius<sub>X</sub>            Arc radius in relation to the X axis in pixels (to make an elliptical arc)
- Radius<sub>Y</sub>            Arc radius in relation to the Y axis in pixels (to make an elliptical arc)
- StartAngle            Arc starting angle (integer degrees, 0 degrees is at positive X-axis, increasing counterclockwise)
- EndAngle             Arc ending angle (integer degrees, 0 degrees is at positive X-axis, increasing counterclockwise)

## **FILLARC:X:Y:Radius:StartAngle:EndAngle**

### **FILLARC:X:Y:Radius<sub>X</sub>:Radius<sub>Y</sub>:StartAngle:EndAngle**

Otherwise the same as ARC, but the result is filled

## **ELLIPSE:X:Y:Radius**

### **ELLIPSE:X:Y:Radius<sub>X</sub>:Radius<sub>Y</sub>**

Otherwise the same as FILLARC, but always draws a complete circle or ellipse

## **POLYGON:X<sub>1</sub>:Y<sub>1</sub>: X<sub>2</sub>:Y<sub>2</sub>:...: X<sub>n</sub>:Y<sub>n</sub>**

Draws a filled polygon with n vertices

## 2.13 TopSkyViews.txt

The file contains the definitions for the items in the View Window. Two types of definitions are allowed, enter only one definition per line:

### *Lat/Lon box*

**VIEW:ViewName:Lat<sub>min</sub>:Lon<sub>min</sub>:Lat<sub>max</sub>:Lon<sub>max</sub>**

An area bounded by the minimum and maximum latitude and longitude values. The resulting screen area covers at least the required coordinates, possibly more depending on the screen shape.

- ViewName            Name to identify the view in the View Window
- Lat<sub>min</sub>            Minimum Latitude (decimal degrees or sector file format)
- Lon<sub>min</sub>            Minimum Longitude (decimal degrees or sector file format)
- Lat<sub>max</sub>            Maximum Latitude (decimal degrees or sector file format)
- Lon<sub>max</sub>            Maximum Longitude (decimal degrees or sector file format)

### *Centerpoint and range*

**VIEW:ViewName:Lat:Lon:Range**

**VIEW:ViewName:PointName:Range**

An area defined by a range from a centerpoint. The resulting screen area will be centered on the required centerpoint and will show at least the required distance to every direction from the centerpoint.

- ViewName            Name to identify the view in the View Window
- Lat                  Centerpoint Latitude (decimal degrees or sector file format)
- Lon                  Centerpoint Longitude (decimal degrees or sector file format)
- PointName          Fix, VOR, NDB, airport code or runway (must be found in the active sector file)
- Range                The displayed range (nautical miles, decimal value)

*Note: the syntax to define a runway threshold as a PointName is the 4-letter ICAO airport designator followed by a forward slash and the runway identifier.*

### 3 Sound files

The plugin uses two sound files:

- TopSkySTCA-sound.wav                      STCA warning sound
- TopSkyCPDLC-sound.wav                      CPDLC/DCL message sound

The limitations for the used files are that the file must fit into available physical memory, be playable by an installed waveform-audio device driver and be located in the same folder as the plugin dll. If the file is not found, no sound is played.

### 4 Cursor files

Six custom cursors are used:

- arrowleft.cur                      General purpose cursor
- arrowright.cur                      Currently not used
- cross.cur                          Used when drawing vectors, selecting points, etc.
- move.cur                          Used when moving windows
- stop.cur                          Used when no input is allowed in that area
- resize.cur                          Used when resizing windows

If even one of the cursor files is not found in the plugin dll folder, none of the custom cursors will be used.