PSXseeconTraffic Tools

Manual

Version 1.0

October 7, 2021

Nico W. Kaan

1 Introduction 3
1.1 LiveryInfoGenerator (LIG) (*formerly named AILGenerator*) 3
1.2 AirportFilesGenerator (AFG) (formerly named ParkPosGenerator). 3
1.3 System Requirements 3
1.4 Forum 3

2 Installation 4
2.1 Download and install 4
2.2 Update to new version and removal 4
2.3 Version Check 4
2.4 Data files needed for LIG and/or AFG 4
2.5 Log.txt 4

3 How to use this tool 5
3.1 10 input folders 5
3.2 Buttons and checkbox 5

4 LiveryInfoGenerator (LIG) 6
4.1 How to use the input folders 6
4.2 What is the output? 6
4.3 How are liveries found? 6
4.3.1 P3D/FSX 6
4.3.2 MSFS 6
4.3.3 P3D/FSX and IVAO MTL 7
4.4 Registration codes 7
4.4.1 Wild cards in registration codes 7
4.5 Non-standard (additional) ICAO airline codes 8
4.5.1 Generic aircraft 8
4.5.2 Private aircraft 8
4.5.3 Government aircraft 8
4.6 Automatic repairs 9
4.6.1 [fltsim.x] numbering 9
4.6.2 Double and old registration codes in the line atc_id= will be removed. 9
4.6.3 Formatting 9
4.6.4 registration code added to title 9

5 AirportFilesGenerator (AFG) 10
5.1 How to use the input folders 10
5.2 What is the output? 10
5.3 Airport file format 10
5.3.1 Parameters 10
5.3.2 Parkpositions 11
5.3.3 Airlines for parking positions 11
5.3.4 Initial parking_percentage value 11

1 Introduction

Legend
These release Notes are as well for PSXseeconTraffic with FSX/P3D as with MSFS. PSXseeconTraffic is sometimes abbreviated to PSXT.

FSX and Prepar3D	MSFS
PSXseeconTraffic	PSXT_MSFS
PSXseeconTrafficTools	PSXT_MSFS_Tools
AILGenerator (AILG)	AILG_MSFS
ParkPosGenerator (PPG)	PPG_MSFS
LiveryInfoGenerator (LIG)	LiveryInfoGenerator
AirportFilesGenerator (AFG)	AirportFilesGenerator

PSXseeconTrafficTools contains two tools that support PSXseeconTraffic.

Note that these are the same tools as included in PSXseeconTraffic v27.0+.

1.1 **LiveryInfoGenerator (LIG)** (*formerly named AILGenerator*)

The main task of the Livery info generator is to find the titles, airline codes, aircraft types and registry codes of AI Aircraft-liveries in your flight simulator.

Information about these liveries is put in a file to be used by PSXseeconTraffic in the matching process with live aircraft.

1.2 **AirportFilesGenerator (AFG)** (formerly named ParkPosGenerator).

The main task of the Airport files generator is to generate airport files with information about the parking positions and the airlines that park there, for the stock and add-on airports in your flight simulator.

PSXseeconTraffic uses these files to position static and live aircraft exactly at gates.

Information automatically 'learned' by PSXseeconTraffic about who parks where, with what type and when, is dynamically added by PSXseeconTraffic.

1.3 System Requirements

PSXseeconTrafficTools is a 64bit Windows application, written in C++, for **Windows 10** or later.

In order to run it, you need Microsoft Visual C++ Redistributable for Visual Studio 2019. The standalone installer for x64 (**`VC_redist.x64.exe`**) can be downloaded here at Microsoft.

1.4 Forum

For questions & answers and discussions use my Support Forum at AVSIM.

2 Installation

This Chapter describes how to install, update and remove PSXseeconTrafficTools.

2.1 Download and install

Download the setup zip file from my site. Unzip and run the installer. You should install PSXseeconTrafficTools at the PC running FSX/P3D.

You better not install in the **C:\Program Files** folder.

2.2 Update to new version and removal

Download the setup file, unzip and run the installer. It will first uninstall your previous version and then install the new one.

Removing PSXseeconTrafficTools from your PC can be done running the uninstaller from the Programs list or the apps list, just like any other Windows program.

In order to completely remove all files, delete the installation folder too and delete the folder with the parameters.xml file from C:\Users\<you>\AppData\Roaming.

2.3 Version Check

At each start-up, but not more than once a day, PSXseeconTrafficTools will check if there is a new version available. If so the –normally- green version message in the main window will turn yellow, and the new version number will be indicated.

This is only an indication. It is up to you when you download and install the new version. It is recommend to always update. Thus it will be easier for me to answer any questions.

It may also happen that you get the message **"error 99: This version has expired"**. In that situation you cannot go further. You have to upgrade to a new version.

2.4 Data files needed for LIG and/or AFG

After installation, you will have a **\data** folder.

The **data** folder contains files with input data:

- **airlines.xml** contains ICAO airline codes
- **aircraft.xml** contains ICAO aircraft type codes
- **types.xml** sums up ICAO aircraft types of heavy and middle aircraft, helicopters and gilders. It also defines similar types. Types that are not listed are assumed to be light aircraft.
- **wingspan.xml** gives the wingspan (in meters) of aircraft types.
- **fixes.xml c**ontains aircraft type fixes that will be automatically made).

You may make changes/additions to these files. However, if you do, make a backup copy because these files will be overwritten with every new version. You better also report your changes in my Forum, so I can put them in.

2.5 Log.txt

In the file Log.txt runs of LIG and AFG are logged.

3 How to use this tool

3.1 10 input folders

In here you should specify full paths to the folder(-s) at your PC running your Flight Simulator (if not in AUTOMATIC mode). These folders have two controls:

- **Swap**
  You can re-order the folders contents with the Swap buttons (S). The higher the position of the folder, the higher its priority.

- **Delete**
  You can delete a folder entry with the X button.

3.2 Buttons and checkbox

- **Edit**
  With this button you enable the folders area for editing.
  When you are finished editing, push this button again in order to make the changes effective.

- **AUTOMATIC** (not available for FSX)
  If checked you do not have to fill in paths in the input folders but the folders for the Search will be found automatically.

- **Switch**
  At each push the program changes between the Livery Info Generator and the Airport Files Generator.

- **Search**!
  If all path names exist, this button will be enabled. The paths will be saved.
  Push this button if you're ready with the folders.
  The search will start…

4 LiveryInfoGenerator (LIG)

4.1 How to use the input folders

The higher the folder, the higher its priority. If the same livery appears in two folders, the livery of the lower priority folder will be skipped.

4.2 What is the output?

The output folder is a subfolder of PSXseeconTrafficTools named **liveries**.

In there a file named **AI_liveries.xml** with information about valid liveries found is stored.

These liveries may be used by PSXseeconTraffic in the matching process with real live aircraft.

4.3 How are liveries found?

LIG finds a livery based on information in the **`aircraft.cfg`** or **`sim.cfg`** file.

4.3.1 P3D/FSX

It searches in the [General] section for:

parameter	is used as
atc_model	ICAO aircraft type code

and it searches in (each) **[fltsim.x]** section for:

parameter	is used as
atc_id	Registration code(-s)
title	Title of livery
atc_parking_codes	ICAO Airline code(-s)
atc_parking_types	Cargo aircraft info (bool), if string "**CARGO**" is found

4.3.2 MSFS

LIG finds a livery based on the following information in the [General] section of the **`aircraft.cfg`** file:

parameter	is used as
icao_type_designator	ICAO aircraft type code

If the **`aircraft.cfg`** file does not have a [General] section but a [VARIATION] section instead, AIL will look up the aircraft.cfg info from the base_container.

and it searches in (each) **[fltsim.x]** section for:

parameter	is used as
atc_id	Registration code(-s)
title	Title of livery
icao_airline	ICAO Airline code(-s)
atc_parking_types	Cargo aircraft info (bool), if string "**CARGO**" is found

Note that all parameter values in MSFS has to be enclosed by double quotes, AILG_MSFS skips these quotes and stores only the real value.

LIG will also recognise parameter values that are **not** enclosed by double quotes in order to support AI packages that do not completely adhere to the MSFS standard. Be careful that you do not feed files with comments after parameter values.

4.3.3 P3D/FSX and IVAO MTL

The structure of the MTL is different from usual (described in the previous section). Here are the differences with the normal liveries:

- The aircraft type is taken from the subfolder name.
- The airline code is taken from the title in the [fltsim.x] section.

Note that the folder with the MTL should be named **IVAO_MTL** (capitals) otherwise LIG will scan the files as if they were normal liveries.

Also note that since IVAO_MTL does not support **`atc_parking_types`**, cargo aircraft cannot be detected.

4.4 Registration codes

Normally a livery has of course only one unique Registration code. However, you may fill in multiple Registration codes separated by comma's. Thus it is easier to separate different groups of liveries for a single type without having to install all liveries. For example to differ between Normal BAW 744 liveries and BAW One World 744 liveries. Or to distinguish all freighter liveries of a certain type from the Pax liveries of that same type, or 773–ER versions from 773 versions. Another major advantage is that if the callsign is missing an aircraft may be matched upon its registration code.

Registration codes may be 12 characters max.

The registration codes for lots of airlines in the format needed by PSXseeconTraffic can be found here.

4.4.1 Wild cards in registration codes

You may use **one** wildcard (the Asterisk sign **'*'**) in a registration code at either the fourth position (first position is 1) or the last position.

4.4.1.1 Last character wild card

Instead of

atc_id=A6-EBA,A6-EBF,A6-EBH,A6-EBK,A6-EBL,A6-EBM,A6-EBP,A6-EBR,A6-EBS,A6-EBT,A6-EBU,A6-EBV,A6-EBX,A6-EBZ,A6-ECB,A6-ECE,A6-ECF,A6-ECG,A6-ECH,A6-ECI,A6-ECJ,A6-ECK,A6-ECQ,A6-ECR,A6-ECS,A6-ECT,A6-ECU,A6-ECV,A6-ECW,A6-ECX,A6-ECY,A6-ECZ,A6-EGA,A6-EGB,A6-EGC,A6-EGD,A6-EGE,A6-EGF,A6-EGG,A6-EGH,A6-EGI,A6-EGJ,A6-EGK,A6-EGL,A6-EGM,A6-EGN,A6-EGO,A6-EGP,A6-EGQ,A6-EGR,A6-EGS,A6-EGT,A6-EGU,A6-EGV,A6-EGW,A6-EGY,A6-EGZ,A6-ENA,A6-ENB,A6-ENC,A6-END,A6-ENE,A6-ENF,A6-ENG,A6-ENH,A6-ENI,A6-ENJ

you may specify

atc_id=A6-EB*,A6-EC*,A6-EG*,A6-EN*

This is a lot less work, saves data in PSXseeconTraffic, speeds up livery search and is future proof while when Emirates adds a new 777 (A6-ENK), it will be recognised immediately without you having to change your aircraft.cfg file.

4.4.1.2 Fourth character wild card

You may also use a wildcard for the fourth character (start counting with 1); this is handy for US carriers.

For example:

atc_id=N72*AN

4.4.1.3 Special German Helicopters "wild card"

You may use code D-H to match all German Helicopters (with registration codes D-H***).

atc_id=D-H

4.5 Non-standard (additional) ICAO airline codes

4.5.1 Generic aircraft

Generic aircraft in house livery or white livery should be given the code **ZZZ**.
Generic codes used in previous versions such as BOE, AIB, EMB, BBA and AED are translated to ZZZ automatically.

4.5.2 Private aircraft

Private (GA) aircraft often do not have an airline code. I recommend to use a non (ICAO) existing airline code **PVT** (for Private) for these type of aircraft. So change the atc_parking_codes lines to

atc_parking_codes=PVT or for IVAO_MTL: title=<aircrafttypecode>PVT

in aircraft.cfg files that define a private aircraft livery. You can use this code for parking positions in an Airport file (ICAO.txt) of PSXseeconTraffic in order to see a PVT aircraft at a certain gate.
For the perfectionists there is also an option to use country specific private airline (General Aviation) codes. These codes are defined by Brian van Hout. If you want to add some, please contact him.

In this document GA Aircraft per country you will find the current set (73 codes).

4.5.3 Government aircraft

Government aircraft sometimes do not have an ICAO airline code. I recommend to use a non (ICAO) existing airline code **GOV** for these type of aircraft. So change the atc_parking_codes line to

atc_parking_codes=GOV or for IVAO_MTL: title=<aircrafttypecode>GOV

in aircraft.cfg files that define a government aircraft livery. You can use this code for parking positions in an Airport file (ICAO.txt) of PSXseeconTraffic in order to see a GOV aircraft at a certain gate.

For the perfectionists there is also an option to use country specific GOV/Military airline codes by Brian van Hout. If you want to add some, please contact him.

In this document Military and Govt Aircraft per Country v6 you will find all military airlines including 93 new non-ICAO codes (in red colour).

4.6 Automatic repairs

The following changes will be made if necessary. A backup file named **aircraft0.cfg** is written in the same folder as aircraft.cfg. An already existing aircraft0.cfg file is renamed to **aircraft00.cfg**.

4.6.1 [fltsim.x] numbering

If the aircraft.cfg file is not coded in UTF16, LIG will repair an in-consecutive [fltsim.x] section numbering in your aircraft.cfg file(-s).

4.6.2 Double and old registration codes in the line atc_id= will be removed.

e.g. in atc_id=PH-AKS,PH-AKR,PH-AKS one PH-AKS will be removed.

Old generic codes BOE, AIB,EMB,BBA and AED are changed into ZZZ which is even more generic and maybe used for any type.

4.6.3 Formatting

LIG will format all your valid (*without errors*) aircraft.cfg files. Formatting implies:

The file starts with the [fltsim.x] sections followed by other sections.
The [fltsim.x] sections will be in alphabetical order of the atc_parking_codes key (= the airline code); if they are equal the ordering will be done on the registration code.

Between [fltsim.x] sections one empty line will be inserted, extra empty lines will be removed.

White space surrounding the title of a livery will be removed.

4.6.4 registration code added to title

LIG will add the registration code of the livery, defined with **atc_id=regcode**, to the title of the livery (if not already present).

This will only take place for exactly defined liveries (with one registration code), so not with more than one registration codes or with a registration code with a wild card.
Note that this does not work for IVAO liveries.

Note: if the registration code is in the title, PSXseeconTraffic will be able to detect an **exact** livery match.

If there are changes due to the repair action a new aircraft.cfg file will be written after the following backups were made:

5 AirportFilesGenerator (AFG)

5.1 How to use the input folders

A path may not end with **\Scenery** or **\scenery**. If the folder exists the **Search!** Button will become enabled.

A folder search **includes all sub-folders**, so preferably specify:

D:\Program Files\Lockheed Martin\Prepar3D v5

and it will search Prepar3D v5 completely.

The priorities are: a stock airport will not overwrite a non-stock airport. An ORBX airport will overwrite a stock airport. An add-on airport will overwrite a stock airport and/or an ORBX airport. If you have add-on airports outside the Prepar3D v5 folder you should add extra line(-s) pointing to these add-ons, for example:

D:\AddOns

You can copy and paste text in the different folders. There is no need to work top down, you may leave "holes"; the order is not important for AFG (contrary to LIG).

5.2 What is the output?

The output folder is a subfolder of PSXseeconTrafficTools named **airports**.

The output folder is the place where for each found airport a file named <ICAO-code>.xml is stored.

They are generated by AFG and may be updated by PSXseeconTraffic, or by you manually, with aircraft options for each parking position (ICAO Airline + ICAO Aircraft Type.

5.3 Airport file format

Airport files are .xml files.

5.3.1 Parameters

For example:

```
<parameters>
   <parameter name="airport_name" value="Rotterdam" />
   <parameter name="supplier" value="P3D stock airport" />
   <parameter name="altitude" value="0" />
   <parameter name="num_parkpos" value="18" />
   <parameter name="parking_percentage" value="75" />
</parameters>
```

The parameters **altitude** and **airport_name** are taken from the .BGL file in your Flight Simulator.

The **supplier** is deduced (if possible) by AFG from the path-name of the BGL file, while most of the time the suppliers name is part of the path. Stock airports are indicated by supplier=<P3D/FSX> stock airport.

The **num_parkpos** gives the number of parking positions at the airport.

The **parking_percentage** determines how many (% of **num_parkpos**) of the parking positions will get an aircraft allocated by PSXT, the initial value is 35. You can change that via the PSXT UI too.

5.3.2 Parkpositions

There can be 0 or more parking positions (parkpos) in the file.

<parkpos id="B1" latitude="51.951576" longitude="4.435111" heading="147.2" radius="18.0" cargo="false" ramp=”false” />

Each **parkpos** has seven attributes. The heading is in degrees and the radius is in meters.

A **parkpos** can have zero or more aircraft options to be added by PSXT in learning mode, or by you manually, see PSXseeconTraffic Manual Chapter 6.

5.3.3 Airlines for parking positions

If the .**bgl** file has information about the airlines that park at a gate AFG will add a special section to the airport file. Unfortunately stock airports mostly do not have that info, but most add-on airports do.

For example a part from the **EGLL.xml** for Aerosoft London Heathrow:

```
<afcad>
   <parkpos id="L233" airls="AUA,CRX,DLH,EWG,GWI,SAS,SWR,TAP" />
   <parkpos id="L246" airls="AUA,CRX,GWI,SWR,TAP" />
   <parkpos id="L247" airls="AUA,DLH,GWI,SWR" />
   <parkpos id="L253" airls="BEE" exclude = “L305” />
   <parkpos id="L254" airls="ACA,AIC,ANZ,AVA,CRX,DLH,WR,TAP,THA,THY,UAL" />
   <parkpos id="L305" airls="FIN" exclude=”L253” />
</afcad>
```

An **exclude** parameter is only there if needed, it gives 1 or 2 adjacent positions that overlap with this gate.

5.3.4 Initial parking_percentage value

The initial **parking_percentage** is calculated based on the number of parking positions (**num_parkpos**) and the values of the following five parameters in the file

C:\users\<you>\appData\roaming\ParkPosGenerator\parameters.xml:

```
DEFAULT_PERCENTAGE_SMALL=75    // 0..100
DEFAULT_PERCENTAGE_MEDIUM=50  // 0 .. 100
DEFAULT_PERCENTAGE_LARGE=35    // 0 .. 100
BORDER_SMALL_MEDIUM=20          //  1 .. 19 parking positions,
                                // value included in SMALL
BORDER_MEDIUM_LARGE=100         //  20 .. 99 parking position,
                                // value included in MEDIUM
```

You may change these five parameters in the **parameters.xml** file to set it to your liking. So you can define what you mean by small, medium and large airports and set the percentages accordingly.