# PSXT Manual

## 2015 – 2025
## Ten years of real time live traffic!

**version 10.10.10**

June 23, 2025

by Nico Kaan

# 1 Introduction

*Important:*

*This Manual applies to four versions of PSXT:*

- *Two **standalone** versions **PSXT** (for MSFS 2020/2024), **PSXT_P3D** (for P3D v5 or v6), and*
- *Two versions that need the **RealTraffic** app: **PSXTraffic** (for MSFS 2020/2024) and **PSXTraffic_P3D** (for P3D).*

*In this manual the term PSXT is used generically and may refer to any of the four.*

With PSXT, you can fly with real-time traffic in P3D or MSFS 2020/2024. The traffic data is provided by a separate source called RealTraffic. RealTraffic streams aircraft position updates every two to three seconds, or occasionally every ten seconds. PSXT buffers the traffic for 30 seconds for vectorisation. Including internet delay, the aircraft you see are typically only about 45 seconds behind real time.

The traffic data originates from ADS-B transponder signals. This works very well while aircraft are airborne. On the ground, it depends on the airport's policy: at some airports, pilots are required to keep transponders on, while at others they must switch them off. Consequently, you may see little taxiing traffic at some airports, whereas at others, you can follow aircraft taxiing to and from stands.

The live aircraft injected (spawned) into your simulator represent actual aircraft present at that moment. They are controlled by live ATC. The aircraft (models and liveries) are spawned as realistically as possible, depending on your installed AI models and parameter settings.

PSXT determines the position of each live aircraft 100 times per second by calculating latitude, longitude, altitude, bank, and pitch. At every frame, PSXT sends the most recent calculated position to your simulator via SimConnect. Therefore, the number of positions visible depends on your simulator's frame rate.

Live airborne traffic is shown within a lateral range of 2 to 100 NM from the user aircraft. Live ground traffic and parked aircraft are shown at airports where you land, remain, or optionally fly over.

PSXT is aware of parking positions at each airport in your simulator. It places departing live aircraft at stands with the correct heading. Likewise, aircraft completing their flight are positioned precisely at their stand.

While on the ground, PSXT also generates static parked aircraft. It first uses data from RealTraffic and then supplements this using options defined in so-called **airport files**. These files are created offline by PSXT and RealTraffic and are updated weekly to reflect changes in airlines, aircraft models, seasonal schedules, etc.

Airport files include details about parking positions, airline usage, aircraft types, and corresponding time slots (in half-hour increments). PSXT tries to match current UTC time as closely as possible to these options. It will dynamically update parked aircraft for the next half-hour by adding and removing aircraft.

When you depart from an airport, all aircraft at that location (static and live) will be removed.
As your destination comes into range, PSXT places static parked aircraft at the appropriate parking positions, making the best possible match based on the current airport file.
PSXT controls gear, throttle, flaps, wheel rotation (aligned with speed), navigation lights, taxi lights, landing lights, cabin lights, logo lights, strobes, and beacon lights.

The performance impact of PSXT is minimal — it uses less than 0.5% of your CPU. Injecting AI aircraft into your simulator naturally affects FPS, but even mid-range PCs can achieve 40+ FPS.

**If you're new to PSXT, follow the Quick Start Guides.**
For questions and discussion, use the Support Forum at AVSIM and/or consult the FAQ.

## 1.1 System Requirements

- **Windows 10** or later, Windows **11** is the **preferred** OS.
- **Microsoft Visual C++ Redistributable for Visual Studio 2022**
  Install Microsoft Visual C++ Redistributable for Visual Studio 2022, version 17, architecture x64.
  It can be downloaded here at Microsoft (scroll down to **Download Visual C++ Redistributables**). Run **VC_redist.x64.exe**

## 1.2 Third-party software used

PSXT uses the following libraries:

- pugixml - an XML processing library
- curl and libcurl
- libssh2
- zlib and libzippp

## 1.3 Disclaimer

The data provided by RealTraffic is not always accurate. This may result in the following issues:

- Aircraft may appear to land in fields rather than on runways.
- Certain aircraft types or airlines consistently land off-runway due to faulty ADS-B data.
- Aircraft may vanish from the stream for over 30 seconds, causing them to disappear from your simulator before reappearing later.
- Taxiing aircraft may temporarily disappear if the ADS-B signal is blocked by buildings.

If you find the ground movement of AI aircraft unrealistic, you can disable ground traffic in RealTraffic.

Note also that some add-on airports may not exactly match real-world layouts and could lack updated taxiways or runways.

# 2 Installation

## 2.1 Download and install PSXT

Download the PSXT software for your Simulator from my site.
Unzip and run the installer with **administrator** rights.

**Install at your computer running MSFS/P3D.**

The default install locations are **C:\PSXT or C:\PSXT_P3D**

You may change that to another drive, but don't install it in the Community folder or in the Program Files folders, and preferably do not change the installation folder name!

After installation you will have a folder containing the files and a few sub-folders, containing.

The executable **PSXT.exe**. On your desktop you will find a shortcut to the executable.
Don't move/copy the executable outside the installation folder.

The **docs** sub-folder contains several documents. The PSXT_EULA is the End-User License Agreement. This is a legally enforceable contract between you and the Author. By accepting the installation, you agree to be bound by the terms of this EULA.

The **input** folder contains data needed by PSXT.

When you run PSXT the following files will be created:

The **output** folder, with two subfolders **info** and **errors** with info about the liveries and airports scans made by PSXT.

The **airports\learned** with its subfolders contains data that is learned on the fly at your end, and remains available with each update. It is not overwritten or deleted.

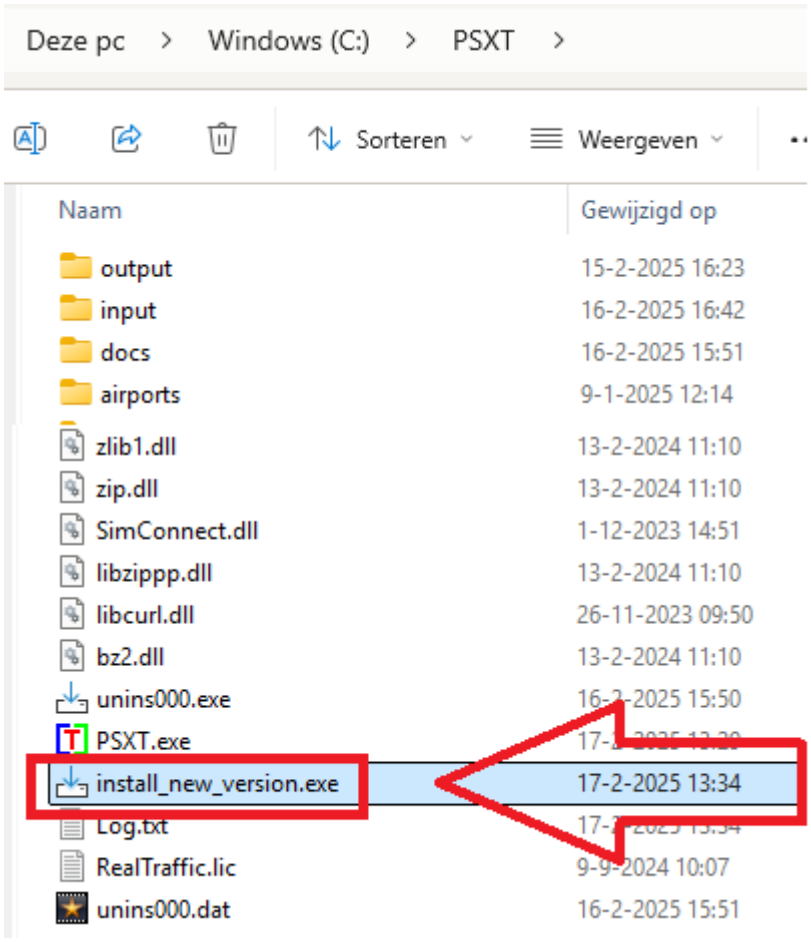
The **airports\my_thirdparty** holds the result of the scan for your airports (see chapter 6). An <icao>.txt file contains the link to the file in the archive at the PSXT web site. An <icao>.xml is an empty airport file, there is no file available in the archive. PSXT will have placed this file in the **TBLO** (to be learned offline) folder too (see 6.2). This empty airport file is used until a file in the archive becomes available.

The **airports\my_simaddon** (MSFS) or **airports\my_ORBX** (P3D) holds the result of the scan for your simaddon files or ORBX files (see chapter 6). A <icao>.txt file contains the link to the file in the archive at the PSXT web site. An <icao>.xml is an empty airport file, there is no file available in the archive. PSXT will have placed this file in the **TBLO** (to be learned offline) folder too (see 6.2). This empty airport file is used (till a file in the archive becomes available).

### 2.1.1 Update PSXT to a new version

PSXT will check at the start of the program for a new version. If so, it will automatically download and extract the installer program **“install_new_version.exe”** in the PSXT installation folder.


Exit PSXT, run the installer as administrator.




It will first uninstall your previous version and then install the new one *(so don’t uninstall the previous version yourself!)*. Language preference and license acceptance are skipped, you have already answered that at first install after downloading from my web-site.

### 2.1.2 Complete removal of PSXT from your PC

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

To completely remove all files, delete the installation folder and delete the folder

**`C:\Users\<user_name>\AppData\Roaming\PSXT`** or what have you

## 2.2 *[Optional]* Install your RealTraffic license *[standalone versions PSXT and PSXT_P3D only].*

Put your license string in the file **`RealTraffic.lic`** in your PSXT and/or PSXT_P3D installation folder(-s).

> *Note: if you already have the RealTraffic app installed and licensed, PSXT will grab the license from*
> ***`C:\Users\<user_name>\AppData\Roaming\InsideSystems\RealTraffic.lic`***

## 2.3 Using more computers running PSXTraffic at the same time *[non-standalone versions PSXTraffic and PSXtraffic_P3D only]*

If you want to run a complex set up with different computers each running the Flight Simulator generating part of the Simulator screen (with WideView for instance) you do the following.

Install PSXTraffic at all PCs. At one (the main) PC install RealTraffic too, don't install RealTraffic at the other PC's, and make sure there is **no** C:\Users\<yourname>\AppData\Roaming\InsideSystems\RealTraffic.jar still present.

Make sure each simulator has the same liveries and add-on airports.

**Important**: In this setup you must switch **Random off** in all PSXTraffic instances otherwise each PSXTraffic may (and will) react (randomly) different in choosing a livery or choosing a static parked aircraft location.

## 2.4 Install AI aircraft packages

Recommended packages are AIG, FSLTL and JustFlight (see the QuickStart Guid).

**Important if you are using AIG in MSFS:**
The lights of AI aircraft are also controlled by MSFS which may conflict with PSXT, and is sometimes completely wrong. You may see taxiing aircraft with strobes on for instance. To fix this, go to

\Community\aig-aigtraffic-modelbehavior\ModelBehaviorDefs\AIG\Customization.xml
and set in AIG_TC_PSXT_Check_Parameters , AIG_TC at **False**

# 3 Operations

The order in which you start the following three programs does not matter.

## 3.1 Start and configure Flight Simulator

- Set the day/time to system time.
- Fly in **1 x simulation rate** (no time compression)
  Rationale: Running with Time compression gives wrong utc times for new learned entries to the airport file, because the time in RT is no longer synchronous with your Sim time.

- Use Live Weather, you need the actual QNH to match with the QNH of RealTraffic!
- P3D: Make sure the Airline Traffic density and General aviation traffic density is set at **0%!**
- P3D: Select an aircraft as vehicle (Pedestrians, Avatars, are not allowed)
- MSFS 2020: See the Quick Start Guide
- MSFS 2024: See the Quick Start Guide

## 3.2 Start PSXT

Start the program using the shortcut on your desktop.

By default, PSXT and PSXTraffic for MSFS run in **MSFS 2020 mode**, as indicated by a ticked checkbox in front of the *Install path* line. To switch to **MSFS 2024 mode**, simply untick this box — PSXT/PSXTraffic will then restart in the new mode.

**Important!**

PSXT will search for your Simulator **install path** first. If not found you have to do something:

MSFS 2020:

Type a full path to your Community folder in the "Install path" text box of the UI and press Enter. You may give it up/to or with Community included, like:

**`C:\MSFS\Community`** or **`C:\MSFS`**

**Note: PSXT will always show your install path in the UI and record it in parameters.xml without \Community**

MSFS 2024: the same as 2020, but with the checkbox in frongt of the install path unchecked.

P3D:
Type a full path to the top-folder of your Simulator in the "Install path" text box of the UI and press Enter, like: **`D:\Program Files\Lockheed Martin\Prepar3D v5`**

You can change between P3D versions by adjusting this line.
Or if this method does not work, set hidden parameter (see 3.2.4) P3D_VERSION at 5 or 6

The black box below the header is divided in two sections each containing a vertical status bar. The left section relates to Flight Simulator, while the right section displays the status of RealTraffic.

### 3.2.1 Sim vertical status bar ("Sim")

Black: not yet initialised, Red: not connected, Yellow: connected but Sim is not running, Green: all ok!

### 3.2.2 Flight Simulator info block

This block displays the name and version of your Flight Simulator, along with the title of the user aircraft. The title appears in yellow if the wingspan is unknown. If the user aircraft is parked at a stand, the stand ID (as defined in the airport file) is also shown.
Additionally, the block shows the number of AI aircraft (liveries) available for matching, as well as the number of available sim add-on (MSFS) or ORBX (P3D) and third-party airport files. If any of these numbers are displayed in yellow, it indicates that not all airports have been offline learned (i.e. filled with parking options).

### 3.2.3 RT vertical status bar ("RT")

Black: not yet initialised, Red: connection lost, Yellow: connected but no traffic data receiving, Green: all ok! If no data is being received over the last minute the bar lights up in yellow.

### 3.2.4 RealTraffic info block

This block shows the version of RealTraffic and your license.

The word "Weather" shows if METAR messages are being received, if so the local qnh is shown. If there is no QNH data for that area, the word Weather is in yellow.

The word "Traffic" shows if traffic messages are being received. If so the number of live aircraft (*flying, taxing, not the live parked*) in your Simulator is shown.
If there is no traffic for that area, the word Traffic is in yellow.

*Note that the **live parked** are not included in the live count, that number is given in the second status line (12.2.2)*

## 3.3 Parameters

The light blue area in the middle contains the check- and text boxes for parameters that can be set. In the

**`C:\Users\<user_name>\AppData\Roaming\PSXT`** folder

a file **`parameters.xml`** will be created and updated.

You can always revert to the default settings by deleting the parameters.xml file.
Parameters that cannot be set via the user interface are called **"hidden" parameters** — you must assign values to these directly in the parameters.xml file.

**Important!**

**How do I enter a value in a text box in the UI?**

Place the cursor in the text box (indicated by light square brackets), type in or edit the value, and when you're ready, press the **Enter** key to activate the new value. Feedback about your input will be shown in the green line at the bottom of the window.

# 4 Meta parameters

### 4.1 Aerowinx PSX *[non-standalone versions only]*

Default: **false**

When checked, PSXTraffic will listen to the traffic data send by RealTraffic that is under control of Aerowinx PSX. This way PSX users that have MSFS or P3D as scenery generator will see live aircraft in that scenery too (and not only at the Aerowinx TCAS).

### 4.2 Port number (PSXTraffic and PSXTraffic_P3D)

Default: **10747**

With the hidden (see 3.2.4) parameter PORT_NUMBER you may set a different port number for the communication with the RealTraffic app (and change the one in RealTraffic accordingly!).

### 4.3 Log

Default: **on**

Adds logging of matching and parking to the Log.txt file (12.1).

### 4.4 Stay on top.

Default: **off**.

If on the PSXT window will always stay on top of your screen

### 4.5 Random *[non-standalone versions only]*

Default: **true**

If the hidden parameter RANDOM is set to false, there will be no randomness in livery matching and no randomness in the stands chosen for static aircraft.

You better leave this value at true. It is needed for users running PSXTraffic's at more than one computer at the same time (section 2.3) **only**.

### 4.6 GUID changes

Default: **true (recommended)**

If set to **false**, live aircraft in the simulator will not be updated (a so-called "livery change") to reflect new origin/destination parameters, updated callsign information, or reinitialised lighting. Although these updates are positive, they have the side effect of changing the object's **GUID**. As a result, if you are viewing that aircraft in spot mode (e.g. for a video stream), it may suddenly disappear from view — even though it still exists in the simulator at the same location.

Also note that if this setting is **false**, external programmes that retrieve data via SimConnect — such as **LittleNavMap** — will no longer receive up-to-date information.

### 4.7 Nav Lights while parked.

Default: **true**

Navigation lights of static and live **parked** aircraft will be on. This parameter is not airport specific.

### 4.8 Learn Soft and Learned files

Default: **true** (only enabled if Qf real of the airport file < 45)

PSXT comes with lots of airports in its on-line archive that are already learned offline with a set of ground traffic data for one week. Normally this will do.

However, for some airports this is not sufficient. **PSXT will always learn real options** for the airport on-the-fly, on top of what is already in the archive.

The learned options will be saved in an airport file <ICAO>.xml in the folder
**airports\learned\stock,** or
**airports\learned\simaddon,** or
**airports\learned\thirdparty\<publisher>**
depending on the publisher, and is not overwritten by each PSXT version update.

When an airport file is loaded, PSXT reads the file from the archive plus the learned options from the file in the learned hierarchy.

If a real live aircraft positions at a stand or an aircraft park at a stand, the **airline, opr** (optional, see 7.1) and **model** combination is added as an option to the airport file with the attribute **day**=**"n"** (n = the code of the day (utc), 1 ... 7, 1 = Monday), plus the current half hour (utc time) is recorded with the attribute **hh="m".** The hour is rounded, the range is 0 ... 47.

The entry in the airport file may look like:

```
<aircraft>
  <opt airl="KLM" type="E190" day="3" hh="17,18" reg="PH-EXA" cs="KLM1002" />
</aircraft>
```

The airline and aircraft type (model) codes must comply with ICAO standards.

If the standard operating procedures (SOP) at an airport require pilots to switch off their transponders after landing or only switch them on at take-off, no taxiing traffic will be visible. As a result, PSXT cannot perform real-time learning and the airport file will not be updated.

To compensate, PSXT will use aircraft that land and/or take off at the airport to update the airport file, by adding the same type of data as in real learning — but with the attribute soft="true" — to a randomly selected stand that best fits the aircraft. The hour coding is the same as in real learning.

Although the stand assignments in this case will not reflect real-world operations, you will at least see parked aircraft of the correct airlines and types at the correct time of day.

Light aircraft, business jets, and private (PVT) operators will be placed at non-cargo ramp positions — not at stands.

**Note that real learning will always take place, soft learning as long as the real part of the Quality factor is < 50 or the airport has < 20 parking positions, and Learn Soft is true.**

The Learn Soft value is stored in your Learned airport file. The value in learned file overrules (but does not change) the Learn Soft setting in the parameters.xml file (i.e the UI setting). This way you can set different values for different airports.

### 4.8.1 Custom made airport files

There are three ways to add custom information (manual additions):

a) You may add park options to the parking positions in the learned file.

**b)** You may add a new **parking position** to the learned file that is not existing in the file in the archive.
Specify it in the same way as in the airport files, plus attribute **manual="true"**

**c)** You may create a **complete** airport file as **alternative** for one in the archive.
Specify the parameters **airport_name** and **publisher** and add parameter **handmade="true"**.

*No need to add the attribute manual to parking positions, as in b).*

PSXT will read the airport altitude from the archived version. The number of positions and the Qf will be calculated and added automatically later.

Place the file in the corresponding folder in the learned hierarchy. For example, suppose you want to define the airport file for Aerosoft EDDF, then place it in airports\learned\thirdparty\Aerosoft.

PSXT will then take your file (only) if Learn Real and/or Learn Soft is true, and add learned options. In the airport line in the UI (h) will appear after the name of the publisher.

*Note that in this case you do not benefit from the offline learned data in the corresponding archive file*

### 4.9 Time Shift *[PSXT and PSXT_P3D only]*

With these boxes you may enter a time shift up to 4 weeks ago. PSXT will read Traffic and Weather data with that time shift.
Ranges are: days: 0 .. 28, hours: 0 .. 23 and minutes: 0 .. 59. All these values will be set to 0 if you close PSXT; in a forced restart the values remain active.
Or alternatively, with the **Sim Time** checkbox (default off) the time in your Simulator determines the time in RT, but again, no more than 2 weeks ago is possible.

### 4.10 Save *[PSXT and PSXT_P3D only]*

Default : **Off**
Whether all Time Shift parameters are kept between sessions or not depends on the setting of this checkbox. At a forced restart the values are always kept.

### 4.11 objectIds [MSFS 2020 only]

Default: false

**Especially for Spotters!**

If set to true, PSXT will – every minute – generate a list of the live aircraft with their objectIds in SimConnect. Listed are Airborne/Ground, Callsign, Airline, Type and whether or not Parked at a gate, and what gate.

With the objectId it is easier to spot an aircraft. Look in Little Nav Map for the callsign, pick up the objectId from the list and select the SimObject in Debug mode of MSFS: Debug -> Simobjects-> Containers -> Airplane -> Containers and click View.

The list is generated in the installation Folder of PSXT, where you can also find the Log.txt, and is named **Callsigns2ObjectIds.txt**

Do not set it to true if not needed, it costs cpu cycles.

# 5 Scanning liveries

PSXT needs to know which liveries are available in your Simulator in order to perform accurate matching. It will scan for your installed AI aircraft (models/liveries), looking for titles, airline codes, aircraft models and registration codes (tail numbers). By default, PSXT attempts to detect your liveries automatically, however, see the next section for more details.

For those interested, more in-depth information about the livery scan can be found in chapter 13.

PSXT will always perform a livery scan when the program is started for the very first time. You can manually trigger a scan at the next start by checking the box before "**Scan for liveries**" then exiting and restarting the programme. This is required whenever you add or remove models and/or liveries from your Simulator.

For **PSXT and PSXT_P3D**: If Flight Simulator is not yet connected when you check the box, PSXT will exit/restart and begin the scan.

For **PSXTraffic and PSXTTraffic_P3D**: If the RealTraffic app is not yet connected when you check the box, PSXT will also exit/restart and begin the scan automatically.

## 5.1 Automatic [MSFS: + Selector:]

Default: **On**

MSFS: in Automatic mode it will search for the AI aircraft packages **AIG** (aig-aitraffic-oci) , **FSLTL** (fsltl-traffic-base) and **FSTraffic** (justflight-aircraft-traffic-fleet) in your community folder. The order in which is defined by the **Selector value**, range 1 .. 6, default 1.

**Meaning**:
1: AIG, FSLTL, JF *(so AIG first, then FSLTL, and JF last)*
2: AIG, JF, FSLTL
3: FSLTL, AIG, JF
4: FSLTL, JF, AIG
5: JF, AIG, FSLTL,
6: JF, FSLTL, AIG

MSFS: After that it will search for a folder **user-aitraffic** in Community. In here you can put your aircraft from MAIW or FSX or whatever, that are not available in the first three packages.

The order (or hierarchy) is important when you are using Prioritize (5.2). Airline + model Liveries in a higher package have priority over the same (airline + model) liveries in a lower packages.

P3D: It will search for the addons.cfg and simobjects.cfg files for packages defining AI aircraft.

If you want to specify the paths to the folders with your liveries yourself, you should **uncheck** the **Automatic** check box.

Specify max 10 full paths to folders with AI aircraft. The parameters in **parameters.xml** file (see 3.2.4) are **lfolder0 … lfolder9**.

MSFS Examples:

```
<parameter name="lfolder0" value="D:\aig-aitraffic-oci-beta" />
<parameter name="lfolder1" value="D:\fsltl-traffic-base" />
<parameter name="lfolder2" value="D:\justflight-aircraft-traffic-fleet" />
```

P3D Examples:

```
<parameter name="lfolder0" value="D:\AI\FSPXAI" />
<parameter name="lfolder1" value="D:\AI\FAIB Boeing" />
```

**<parameter name="lfolder2" value="D:\AI\FAIB Airbus" />**

### 5.2 Prioritize

Default: **Off**

If **prioritize** is checked, a livery for an airline + type combination in a higher folder has priority over a livery for the same airline + type combination in a lower folder. Hence liveries in the AI package indicated by lfolder0 have priority over liveries in the AI package indicated by lfolder1*. (0 is defined to be higher than 1 and so on).*
The one in the lower folder will be neglected and it will not be added to the "working set" of liveries (the file \output**LiveriesScanned.xml**) from which PSXT performs the matching with a real live aircraft.

### 5.3 STUB

Default: **true**

If this parameter is true, liveries with a title ending with "_STUB" (mainly in the FSLTL package) will be **excluded** from the scan results.

# 6 Scanning add-on airports

PSXT must know what add-on airports you have installed. The purpose of the scan is to either generate empty **airport files** containing parking position data or to create shortcuts existing populated files from the on-line archive.

PSXT will automatically perform an airport scan at the very first start of the program. You can manually trigger a scan at the next start by ticking the checkbox next to "**Scan for addon airports**", then exiting and restarting. This is necessary whenever you add or remove airports in your Simulator.

For **PSXT and PSXT_P3D**:
If Flight Simulator is not yet connected when you check the box, PSXT will exit, restart and scan automatically.

For **PSXTraffic and PSXTTraffic_P3D**:
If the RealTraffic app is not yet connected when you check the box, PSXT will exit, restart and scan automatically.

Information about your third-party airport files is stored in the **`my_thirdparty`** subfolder.
For MSFS, information about the airport files from Microsoft and Asobo that come with MSFS are stored in **`my_simaddon.`** For P3D, information about the airport files from ORBX are stored **`in my_ORBX`**.

For MSFS, PSXT uses the BGL file path as well as the **Manufacturer** and **Creator** fields from the *manifest.json* file, for determining the **Publisher**. If that info is missing or just a few initials, PSXT cannot determine the Publisher, a situation that commonly occurs with freeware airports from sources such as flightsim.to.
For P3D, PSXT can only rely on the BGL file path to determine the publisher. Therefore, it is recommended to group your add-on airports into subfolders named after the Publisher.

You can view the BGL path, Manufacturer and Creator info found in the generated (empty) airport file. See FAQ #38 for more information.

MSFS only: Airports purchased from the Marketplace are encrypted, PSXT cannot read them, so avoid buying there!

## 6.1 Automatic (or Manual)

Default: **On**

Note: This option is given for Prepar3D only. For MSFS PSXT always scans the Community and Official Folders for airports (symbolic links included).

If you want to specify the paths to the folders with add-on airports yourself, you should **uncheck** the **Automatic** check box.

Specify max 10 full paths to folders with addon airports. The parameters in **parameters.xml** (3.2.4). are **`afolder0 … afolder9.`**

```
<parameter name="afolder0" value="D:\Addons\Airports" />
```

A folder search includes all sub-folders!

Information about the search is stored in the file **`output\info\AirportsScanLog.txt.`**

## 6.2 What to do with empty airports or airports that need an update?

Not all addon airport may already have been **off-line** learned (filled) by PSXT and RealTraffic, or newer versions have arrived. If you have these, you will see a yellow message in the UI.

Please zip the folders "**TBLO**" and "**TBLO2024**" (*you can find that in the PSXT installation folder*) and mail that please to leveld757@gmail.com

# 7 Aircraft parameters

## 7.1 Max aircraft matching depth

Default 4, range 1 … 4.

This parameter determines the maximum number of matching steps PSXT will perform when searching for a livery for an aircraft (both live and static). The lower the value, the better the match quality — but you may see fewer aircraft. **1** is the strictest (lowest) setting.

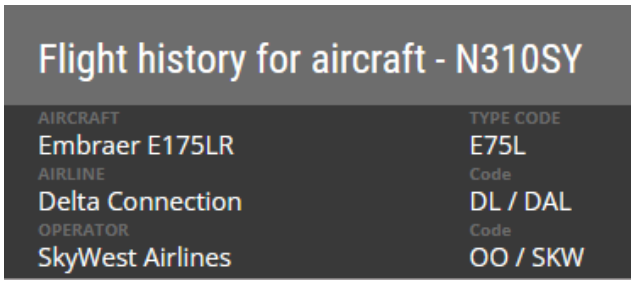
Liveries are matched using a best-effort process of up to six steps:

| Step | Match | Indicated in the Log with |
|---|---|---|
| 1 | Match of an aircraft by registration code | =1= |
| 2 | **PVT** aircraft only: match of a generic livery with airline ZZZ and model [a generic livery: icao_airline (MSFS) or atc_parking_codes (P3D)=**ZZZ**] *This will result in a so called white or house paint*<br><br>or a match of an aircraft by airline and/or operator* codes and model | =4=<br><br>=2= |
| 3 | Match an aircraft by airline and/or operator* and a similar model of the wanted model | =3= |
| 4 | Match of a generic livery with airline ZZZ and model. *This will result in a so called white or house paint* | =4= |

For light aircraft, business jets and helicopters PSXT will try these two extra steps, if matching is set at 3 or 4:

| | | |
|---|---|---|
| 5 | Match an aircraft by model (whatever airline) | =5= |
| 6 | Match an aircraft by a similar model of the wanted model (whatever airline) | =6= |

*operator: often operator is the same as airline, but there are operators that operate for an airline.




See this example from FlightRadar24, the **operator** is **SKW**, operating for **airline DAL.**

There is a special case:

=c= means "matched" from cache, an aircraft that re-appeared, after a hick-up in RT

## 7.2 Optional: checkboxes to include Light aircraft, Helicopters and Fighter Jets

PSXT always spawns business jets, large, and heavy models of live aircraft.

Ground vehicles, Gliders, Airships and Balloons are not spawn.

Optionally PSXT will also spawn light aircraft (GA), high-performance (Fighter Jets) aircraft (F16, F35 and so on) and helicopters.
*Note that your helicopter liveries should have Category=airplane in their aircraft.cfg file, and not Helicopter*.

If you do not have models from one of these categories, the checkbox becomes disabled.

# 8 Traffic injection parameters

PSXT always adds a basic flight plan — including departure and destination — to each live aircraft in your simulator.

In P3D, you can enable Flight Plan under the Information Options for vehicle labels to display the route (from → to) above each live aircraft.

External programmes such as Little Navmap and EFB also benefit from this, as they can display the flight plan information.

Note that this simple flight plan is purely informational — it does not affect aircraft behaviour or trigger automated flights. PSXT remains fully in control of each aircraft.

The following parameters all influence which traffic is displayed in your simulator.

## 8.1 Radius

Default: **40 nm**, range 2 … 100 nm.

All airborne live aircraft within a (horizontal) radius of the user aircraft will be injected.
All live ground traffic at airports where you land or park within the lateral range will also be injected.

If you are using FSUIPC, ensure that the Limit TCAS range setting on the Misc tab matches the value set in PSXT. Do not use the FSUIPC traffic limiter or any other traffic limiters, as they will interfere with PSXT's internal traffic management.

This value is stored in your learned airport file. The value in the learned file overrides — but does not change — the Lateral Range setting in the parameters.xml file (i.e. in the UI). This allows you to specify different values for different airports.

## 8.2 Auto range

Default: **on**.

The **Automatic** checkbox is useful for pilots flying from departure to destination in cockpit view. When enabled, PSXT dynamically adjusts the *Lateral Range* parameter according to the phase of your flight — from **10 NM** while on the ground to **40 NM** at cruising altitude. This ensures that only aircraft relevant to you as a pilot are injected into the simulator and shown on your TCAS, resulting in the smallest possible impact on FPS.

While on the ground, airborne traffic is limited to aircraft flying below **3000 ft AGL**, allowing you to see only arriving and departing traffic.

This mode is recommended when flying to or from busy airports such as **EGLL** or **KLAX**.
Note: it is either **automatic range** or **manual radius** — not both.

This value is stored in your **learned airport file**. The value in the learned file overrides — but does not change — the **Auto Range** setting in the parameters.xml file (UI). This allows different values to be set for different airports.

## 8.3 ah

Default: **49000** , range 18000 .. 49000

If the user aircraft flies above this **height** the auto range will be set to **100** nm, overruling other rules of the auto range algorithm.

### 8.4 Two hidden parameters control holding a live aircraft alive

There are sometimes/often **gaps** in the traffic data coming from RT. PSXT tries to overcome these gaps by doing the control of the aircraft itself for a while. A special case is a **taxing aircraft:**

If there is no data for it, PSXT will **hold** its position until another live aircraft is about to collide with it, if so PSXT will remove the holding aircraft.

PSXT will keep an aircraft holding for max 20 minutes (*customizable, see below*).

If new data becomes available for a holding aircraft PSXT will let the aircraft continue if the new position is less the 200 meters (*customizable, see below*) away, otherwise it will be removed.

With the following two hidden parameters one may customize the algorithm (*but try the default values first I'd say)*:

HOLDING_THRESHOLD_METERS, values 10 .. 4000, default **250**

and

MAX_HOLDING_MINUTES, values 1 .. 60, default **20**

# 9 Callsigns

## 9.1 Block live callsign

Default **empty**.

Enter the callsign of a real live aircraft. As a result, the real live aircraft will not be spawned in your Simulator, or if it already was in your Sim, it will no longer be visible in your Simulator.

You may as well enter an already spawned aircraft or one that will be spawned shortly.

*Note that lower case letters will be automatically transformed to upper case.*

*If you wipe out a blocked callsign, do not forget to press Enter as well. If all okay, you will see a message "no more blocking". However, it is easier to use the unblock push button of the next section.*

## 9.2 Unblock push button

Enable if there is a call sign to be blocked. By pushing this button you unblock the aircraft and it will show up again after say 30 seconds.

## 9.3 oid [MSFS 2020 only]

Especially for spotters!

Here you can ask the objectid of an aircraft by its callsign. The value of the objectid is shown next to the callsign in the UI (e.g. = 12101).

With the objectId it is easier to spot an aircraft in MSFS.

1. Look in LittleNavMap for the callsign,
2. Pick up the objectid from the UI,
3. Select the SimObject in MSFS's Devmode as follows:
   Debug -> Simobjects-> Containers -> Airplane -> Containers and click View.

# 10 Collision avoidance

At busy airports, it can sometimes be difficult to find a place in the departure queue. At London Heathrow, for example, there may be as many as 13 aircraft waiting for take-off in two parallel queues, each using different runway entry points. Occasionally, an aircraft may even land on that same runway...

You can activate near collision avoidance functions, or you can block one of these aircraft as described in section 9.1 and take its place.

The following features may make your live as a pilot easier…

### 10.1 Collision avoidance user aircraft with other aircraft (live + static) on the ground

Default: **true**

PSXT detects a near collision if two aircraft are at a distance from each other less than the sum of their wingspans divided by two.

If checked collisions between the user aircraft and live aircraft on the ground or static parked aircraft will be prevented by removing the other aircraft until it is safe.

### 10.2 Collision avoidance of the user aircraft with live aircraft in the air

Default: **true**

PSXT detects a near collision if two aircraft are at a distance from each other less than 4 miles, within an altitude difference of 1’000 feet and a “width” less than 700 ft.

PSXT removes/hides the live aircraft temporarily until it is safe to show it again.

As an extension PSXT also removes a live aircraft when during a landing, starting from 500 ft above ground, it detects a live aircraft right in front (entering or already on the runway), within 1 nm.

### 10.3 Collison avoidance live aircraft x static parked aircraft

Default: **false**

PSXT detects a near collision of two aircraft are at a distance from each other less than the sum of their wingspans divided by two. If checked collisions between live and static parked aircraft will be prevented by removing the static parked aircraft.

Note: PSXT will always removes a static parked aircraft from a stand if a live aircraft needs it, so also if collision avoidance live X park is false.

# 11 Airport parking

If an aircraft is on the ground within 5 NM of a known airport, an airport file for that location — containing information about which airline parks where, when, and with which aircraft type — is downloaded from my archive hosted on my website. PSXT uses the information collected during the airport scan (see Chapter 6) to determine whether a third-party add-on file, a sim add-on airport file, or a stock airport file is required.

An airport file is also loaded if the conditions set for within range (11.1.2) and height above ground (11.1.4) are met. This way it facilitates parking and traffic at airports you fly-over.

The downloaded file is available for inspection at **airports\park_airport.xml**.

PSXT uses the airport file for parking live and static aircraft.
Parking static aircraft at, and enabling live ground traffic for, an airport takes place:

- if you have entered that airport as your destination in the UI, or
- when you start at an airport, or
- if you land at an airport, not being one of your park airports, when you have come to a stop, or
- **for airports you have defined to be park airports** (see 11.1), in flight as soon as you are within reach of the airport, and if it is the one closest by if there are more airports within that reach.

If you fly away from an airport, the static parked aircraft will remain visible until you are out of range — or earlier, if another airport comes within range and is closer.

Every twenty minutes, necessary changes to parking positions are re-calculated — unless **ILO** is set to *true* (see Section 11.3). The twenty-minute cycle begins when PSXT is first loaded. The time
The words "Airport parking" are shown in yellow if schedules parking takes place.

*PSXT and PSXT_P3D:*

> PSXT will start parking aircraft from Real Traffic's 24 hour history, and if the number of parked aircraft requested by the user in 'Airport parking %' is not reached then it will continue with parking options from the airport file.

*PSXTraffic and PSXtraffic_P3D:*

> PSXT will start parking the number of parked aircraft requested by the user in 'Airport parking %' with parking options from the airport file.

### 11.1 airports\my_park_airports.txt

To begin with, **all** of your **third party addon** airports are taken as your park airports (in MSFS from 2020 and 2024).
However, in a file named **my_park_airports.txt,** to be put in folder **\PSXT\airports,** you may specify **more** airports, and (optionally) you may define a reserved stand (gate or ramp, excluded for static and live parked) for your aircraft at the airport.

Specify one airport (the ICAO code) per line. If you want to reserve a stand you must add that to the airport ICAO separated by a comma. Enter the name of the stand, as defined in the airport file, look for the **id** field as in <parkpos id="**A91.**

The airports and stands you enter are checked upon validity when reading the my_park_airports.txt file at start-up, and after pushing the re-read button (see 11.1.1). If there is an error airport parking will show in red. If info about what the problem is, is no longer shown, push the re-read button (again).

An example my_park_airports.txt:

EFHK
EHAM,B24
ESSA,S72

You can also let PSXT do that work for you, because:

- when you start at an airport that is not one of your park airports, it will be **automatically** added to your my_park_airports.txt, and if you are parked at a stand, the stand will be added too.
- when you land at an airport that is not one of your park airports, the airport will be added.
- when you start at an airport that is one of your park airports, parked at a stand your stand will be updated if applicable.
- when you exit your Sim or load a new flight while you are at a stand, the stand will be saved in your park airports.

In MSFS my_park_airports,txt is used in both modes (MSFS2020 and MSFS2024).

#### 11.1.1 Re-read button

If you push this button your my_park_airports.txt file will be read again. This is useful for

- adding a destination airport and or destination stand to your my_apark_airports.txt file while PSXT is running;
- getting info about the name of the airport or stand id that is not valid, indicated by a message in the messages line and "Airport parking:" will show in red.

#### 11.1.2 Destination

Default **empty**, valid ICAO codes for airports only

If empty airport parking takes place at the airports in your my_park_airports file.

If not empty airport parking will take place at the destination airport only, and only when the conditions set by range and height above ground (see next) are met.

The destination can be set when starting in the air, or when starting on the ground after the airport file has been loaded.

#### 11.1.3 Within range (nm)

Default **20**, range 8 .. 30

Parking at an airport starts if the user aircraft is within the range of the airport and below the height (see 11.1.4), and if the airport is in your park airports (see 11.1).

#### 11.1.4 height above ground (ft)

Default **18000**, range 2000 .. 45000

Parking at an airport starts if the user aircraft is within range of the airport (see 11.1.2) and below the height **above ground**, and if the airport is in your park airports (see 11.1).

### 11.2 **Wanted** parking percentage **%**

Default: **30**, range 0 .. 100

This parameter determines the wanted and maximum number of parking positions at the airport that may have a live or static parked aircraft. This value is the percentage of the number of parking positions at the airport. If the maximum has been reached and a new live want to park, the oldest static will disappear first, and if no static is available the oldest live aircraft will disappear.

Note: If you set the value at 0, only live aircraft may and will park (without a max)!

A live aircraft that has parked but has no longer a Real Traffic data connection will remain parked for maximum 3 hours (heavies) or 90 minutes (the rest), or it may of course start its return flight.

PSXT will automatically search for the best allocation of static parked aircraft, based on the parking options with day, hour, and real info found in the airport file. In the Log.txt you'll see a message whether there were sufficient parking options for the hour of the day or not.
PSXT will first search for options for the actual half hour of the day. If there are not sufficient options found it will search in the previous half hour, and so on, until it has searched 24 hours or it has found sufficient options. In that process it may cross the day boundary, and continue at the previous day.

Note: static parked aircraft have a callsign with a trailing star (*) character!

### 11.3 **Wanted generic** parking percentage

Default: **-1**, range -1 .. 100 **(-1 meaning not used**)

This is hidden parameter CUSTOM_PARKING_PERCENTAGE.

If >=0, this value will **overrule**, but not overwrite, **for every airport visited,** the value from the airport file (see **Fout! Verwijzingsbron niet gevonden.**).

### 11.4 Generic max park

Default: **-1**, range -1 .. 500 **(-1 meaning not used**)

This value will be, **for every airport visited**, the maximum number of parked (live+ static) aircraft.

This value is not stored in learned airport file.

### 11.5 Max Live

Default: **-1**, range -1 .. 300 (**-1 means not used) **)**

**Leave it at -1 if you are not using it. Note that you set lateral range first and then -if needed- Max Live.**

This function is to be used at non-high-end PC's flying at major airports with lots of traffic.
With this parameter the maximum total number of live aircraft (**parked and not parked**) can be limited. It may accidentally overshoot, but PSXT will correct that later.

This value is stored in your Learned airport file. The value in learned file overrules the value of Max Live in the UI, during the time the airport is loaded. This way you can set different values for different airports. When you leave the airport the default global setting is used again.

### 11.6 Straight push backs only

Default: **false**

If checked push backs will always be straight out. If not checked PSXT will try to make a 90 degrees curve, either to the left or to the right. This guess is not always right, resulting in a pirouette when leaving. So if you do not want that, check this box.

### 11.7 Park fake

Default: **false **)**

If checked, fake options will be used for static parking if there are not sufficient real and/or soft options.
You can use this option for airports for which RealTraffic provides no Traffic data, or for airports with a very low Qf. PSXT generates for each gate a fake option (in memory, not in the airport files).
Hence these airports will not look deserted when you arrive there.

You may see messages in the Log about these faked aircraft. The registration code and callsign have FAKE in it. Most of them will be light and business jets. If the ICAO code of an airports begins with K,

E, C, Z or R airliners that may fly in that region may be generated too; in other regions just a generic B738.

### 11.8 Week

Default: **false **)**

If true: In the search for options for static aircraft in the airport file, the **day no longer plays a role**. So there is **still a check upon max fit** but not at the day. For airports with a very low Qf this is a means to get more static aircraft parked at that airport (although not according to real live).

### 11.9 Park soft

Default: **true **)**

If checked, **soft** options from the airport file will be used for static parking. These aircraft have visited the airport at that time of the week but their stands are unknown and therefore chosen randomly.

### 11.10 ILO (Initial load only)

Default: **false **)**

If checked, PSXT will generate static aircraft at the first load of the airport, or after a change in percentage, Park Soft, Park Fake, or Max Fit by the user. No adjustments every twenty minutes.

Live parked aircraft will stay parked, until another live aircraft needs the stand or it starts a new flight.

### 11.11 Max fit

Default: **24.0**, range **0.5 .. 24.0 hours** (with one decimal, half hours only). ****)**

If 0.5, only real, and soft (depending on Park Soft in 11.3), options in the airport file for the **current day** of the week and **current half hour** will be used for parking of static aircraft.
For values > 0.5 it will search back for more time.
The Max fit value also determines how many parked aircraft from the RT server are taken into account; value 24 implies all, value 0.5 implies current half hour only, and so on.

### 11.12 PT only (PSXT and PSXT_P3D only)

Default: **false **)**

If true (on) PSXT will use the parked traffic (PT) info it receives from RealTraffic **only**! Hence parking options from the airport file are **not** used. This way you may get less aircraft, but all are at the right place for that time of day! This in combination with live parked traffic may be a good option for some airports.

In this mode all other parking controls *(to the left of the PT checkbox at the same line)* **are not used** and disabled. You will get all aircraft that are still parked at the airport up to 24 hours. You may probably see a lot within say 6 hours and only a few from 20 or so hours ago (staying overnight, or a chartered aircraft waiting to return, for instance).

Every 20 minutes the situation gets updated *(irrespective of the value of ILO*);

### 11.13 Flush static button

If pressed, PSXT will remove all static parked aircraft and set ILO (11.3) to true. This will leave you with live (parked) aircraft only.

*****)*** *The value is stored in your Learned airport file. The value in the learned file overrules (but not changes) the value of it in the parameters.xml file when the airport file is loaded. This way you can set different values for different airports.*

# 12 Session information

## 12.1 Log file

During each PSXT session a file **Log.txt** will be written in the PSXT installation folder, whilst saving up to 2 previous version (**Log0.txt** and **Log00.txt**).
Have a look into this file every now and then, and if you have problems add it to your post in the Forum. Info about matching and parking will be added if Log (4.3) is on.

## 12.2 Status lines

The black box below the parameters section provides information about parking of aircraft at your airport.

### 12.2.1 First line

The ICAO and IATA code (if available) of your airport. The airport name and the publisher from the airport addon.

**(h)**
If the file is handmade (see 4.8.1 section c) (h) will appear after the name of the publisher.

**wk=n** n is the week number of the airport file (from the archive) (1 .. 52)

***S***
If this shows **soft** static park options were used (11.3).

**qf: <real[/soft]>**
The quality factor of the airport file (0 .. 100) as well for real as for soft. If real >=45 there are no soft options so there is not shown a Qf soft. If there is no traffic data the Qf is empty.

**reserved stand: <>**
If specified for this airport (see ) the stand id will show up here. If it is not a valid id it will be in yellow.

**# updates <n>**
**G**ives the number of real and soft updates found during this session, they will be added to your learned file for that airport.

### 12.2.2 Second line

**# stands: <n>**
The total number of parking positions (stands), defined in the airport file.

**parked live: <n>**
The number of live aircraft that are parked at the airport.
*Note that the live parked are not included in the live count in the top right RT section of the UI.*

**average fit: <n.m>.**
The average time -in hours- of parked live aircraft at the airport.

**parked static: <n>**
The number of static aircraft that are parked at the airport.

**average fit: <n.m>.**
The average time -in hours- of parked static aircraft at the airport. This value -together with max fit from the first line- is an indication how "actual" the parked static aircraft are.
Not available in "week" mode.

**max park: <n>**
The max allowed total number of parked live + static (based on your specified percentage and the number of parking positions. Messages

### 12.3 Messages line

At the bottom line of the GUI several messages appear.
Information messages show up in yellow font against a green background**.**
Error messages show up in yellow font against a red background. Error messages are severe. After reading, you must exit PSXT with the Windows X button.

# 13 In-depth information about the Liveries scan

## 13.1 How are liveries in your Simulator found?

Liveries are found by scanning **`aircraft.cfg`** and **`sim.cfg`** files.

What information it is looking for is dependent on the Simulator.

MSFS:

In the **General** section of the aircraft.cfg file it searches for

**icao_type_designator** two to four letters/digits defining an ICAO valid model

In each **[Fltsim.x]** section of the aircraft.cfg file it searches for

| | |
|---|---|
| title | the title of the Livery |
| icao_airline | the 3 letter ICAO valid code of the airline.<br>special cases: PVT for private and ZZZ meaning generic (white livery) |
| atc_id | the registration code (tail number) of the aircraft |
| atc_parking_types | if it contains CARGO, it defines it to be a cargo aircraft |

P3D:

In the **General** section of the aircraft.cfg file it searches for

**atc_model** two to four letters/digits defining an ICAO valid model

In each **[Fltsim.x]** section of the aircraft.cfg file it searches for

| | |
|---|---|
| title | the title of the Livery |
| atc_parking_codes | the 3 letter ICAO valid code of the airline.<br>special cases: PVT (private) and ZZZ meaning generic (white livery) |
| atc_id | the registration code (tail number) of the aircraft |
| atc_parking_types | if it contains CARGO, it defines it to be a cargo aircraft |

IVAO:

A special case is **IVAO MTL** (P3D) **/ ivao_x-csl** (MSFS):
The folder with the MTL must be named IVAO_MTL (capitals) or ivao_x-csl (lower case) otherwise PSXT will scan the files as if they were normal liveries!

The model of the aircraft is taken from the folder name, example \Airplanes\IVAO_**B738**

In each **[Fltsim.x]** section of the aircraft.cfg file it searches for

| | |
|---|---|
| title | the title of the Livery |
| title | It searches for the 3 letter ICAO valid code of the airline in the title.<br>special case: ZZZ meaning generic (white livery) |
| atc_id | the registration code (tail number) of the aircraft |
| atc_parking_types | if it contains CARGO, it defines it to be a cargo aircraft |

For all simulators: if the registration code is defined, information found about airline, model and cargo may be overruled/adjusted by -presumably- better information in PSXT's database (if available ...).

## 13.2 Registration codes (tail numbers)

There are two possible ways to specify registration codes for liveries, **in-line** and **offline**. The in-line definitions are processed first.

### 13.2.1 In-line specification of registration code

The registration code is taken from the **atc_id=** line of the **[Fltsim.x]** section in the aircraft.cfg file (see 13.1). Registration codes must be 3 to 12 characters. One registration code (only) is recognized and recommended to make it unique.
If you want more registration codes to map to the same livery, you must use the offline method of section 13.2.2.

If a registration code is used by more than one livery, the first livery found will be kept.

The errors found in processing the inline regcodes can be found in the file **`output\errors\InlineRegcodes.txt.`**

### 13.2.2 Offline specification of registration code(-s)

In folder **input\my_regcodes**, one may put one or more text files. The name is not relevant but it must be unique, with extension **.txt**.

In the file one should specify the (exact) title of the livery followed by a semicolon, followed by a list of registration codes separated by commas. A livery title may not appear in more lines.

Liveries that have just one registration code in an offline file, will be treated as being **special**, they will only show up as live aircraft, not as static parked aircraft.

If one adds **;#** to a line, **all** registration codes will be treated as special, for example:
AIGAIM_easyJet Europe Airbus A320-200SL - Europcar;OE-IVC,OE-IVT;#

If you specify a **livery title only**, (so no registration codes) PSXT will treat it as a **don't use** livery.

30000 regcodes for AIG aircraft are already available in PSXT, hence you better focus on other AI packages then AIG.

An offline specified registration code **overrules** one defined in-line (*the inline regcodes are processed first*), and a newer offline definition overrules an earlier offline definition for the same registration code!

In the Log.txt file a summary is given of the results, and the details of replacements are summarized in **`output\info\OfflineRegcodes.txt.`**

## 13.3 Files about errors found in the scan

With these files it will be easier for you to track/pinpoint, and manually repair errors in your AI aircraft package(-s). You should do that if you want to make the best out of your AI aircraft, otherwise you can also choose to go on without paying attention to it.

- **`output\errors\Fixes.txt`**
  Some obviously wrong Aircraft model codes will be fixed automatically.
  Some obviously wrong airline codes (found in atc_parking_codes= ) are fixed as well.
  The fixes are applied internally, your **`aircraft.cfg`** file(-s) remains untouched.
- **`output\errors\InvalidAirlineCodes.txt`**
  This file contains invalid ICAO airline codes.
- **`output\errors\InvalidAircraftCodes.txt`**
  This file contains invalid ICAO aircraft model codes.
- **`output\errors\InlineRegcodes.txt`**
  It shows output\errors in the [fltsim.x] sections about non-unique registration codes.
- **`output\errors\InvalidAircraftCfgFiles.txt`**
  Contains invalid aircraft.cfg files. Problems may be:

  - [fltsim.x] numbering not consecutive.
  - missing atc_model keys, icao_designator etc.
- **output\errors\IncompleteFltsimSections.txt**
  The file contains the incomplete [fltsim.x] sections; a section is incomplete:
  - if the title key is not unique
  - if the texture key is invalid
  - if the model key is invalid
  - if the sim key is invalid
  - if there is more than one title key
  - if there is more than one atc_id key
  - if there is more than one atc_parking_codes key

  If incomplete, the livery is skipped.

### 13.4 Files with information about the scan

These files give info about non-erroneous issues from the search.

- **output\LiveriesScanned.xml**
  It shows the results of the liveries scan. These are the liveries that will be used in matching.
  *Warning: do not make any changes in this file, it is read by PSXT.*

- **output\info\AircraftCfgWithNoFltsimSection.txt**
  This file gives all the aircraft.cfg file(-s) that are "empty", having no [fltsim.x] section.
- **output\info\AirlineModelConversions.txt**
  It shows the conversions of the airlines + models found in your liveries (from the aircraft.cfg files) to the values found in the PSXT database. The corrected values are stored internally, your aircraft.cfg files remain untouched.
- **output\info\OfflineRegcodes.txt**
  It shows the results of processing the offline registration code files (13.2.2).