PSXT Manual

(PSXseeconTraffic)

version 31.15.5

August 8, 2022

by Nico W. Kaan

1 Introduction

With PSXT you can fly with real live traffic in P3D, FSX or MSFS. PSXT injects live aircraft within range of your aircraft in Flight Simulator.

The traffic information is provided by **RealTraffic**. The RealTraffic stream gives position updates of all aircraft around you every **two** seconds. To be able to vectorise the traffic, PSXT buffers the input for thirty seconds. The traffic you see is therefore thirty seconds + transmission lag **(= approx. 45 seconds) behind real time**. If you would consult apps such as FlightRadar24, or would listen to a real ATC channel, this must be considered.

The data is coming from ADS-B transponder data of aircraft. When airborne this gives very good results. On the ground it depends on airport policy, at some airports pilots must have their transponder on while on other airports they must switch it off. At those airports there will be not much taxiing traffic to see. But on the other airports you may follow the aircraft after landing taxiing to the gate or from the gate to the take-off runway.

The live aircraft injected in your simulator are representations of real aircraft that are there at that time. They are controlled by live ATC and cannot be controlled by artificial ATC programs for Flight Simulators, such as ProATC. Be aware: they will not make a go-around if you enter the runway!
The aircraft (models/liveries) are created in your Sim as real as possible, depending on how many different AI aircraft **you** have installed.

PSXT calculates a position for each live aircraft thirty times per second by calculating latitude, longitude, height, bank, and pitch. At each new frame PSXT pushes the latest calculated position in your Simulator.

Live airborne traffic will be shown within a lateral range from 2 to 100 nm around the user aircraft. Live ground traffic and live parked aircraft is shown at your origin or destination airport only.

PSXT knows the exact parking positions at each airport in your Flight Simulator, it places departing live traffic at gates at the exact position with the right heading. Even so, a live aircraft ending its flight at a gate will be placed exactly on spot.

When you start a session on the ground PSXT will generate static parked aircraft. PSXT will use the options defined in a so-called **airport file**. The airport file has already been created by PSXT for each airport. It contains information about the parking positions, what airline parks where with what type at what hours of the day, etc. PSXT will try to match the options with the current UTC hour as much as possible. It will gradually change the static aircraft to the situation of the next hour by removing and adding aircraft.

When you fly away from the departure airport all parked aircraft (static and live) will be removed. This happens when you are more then 40nm away without a destination set up, or much sooner if you do have a destination set.

When your destination airport comes into sight, PSXT places static parked aircraft at the parking positions of the destination airport. It will make the best actual fit, based on the information in the airport file.

PSXT **learns** from live aircraft parking at gates what airlines with what types of aircraft park at what gates at what time of the day and optionally at what day of the week . This info is added to the airport file so it will be able to better place static aircraft at parking positions later in time.

There's also a separate **Learner** application that learns without a need for a Flight Simulator connection. You can let it run for hours /days while you are doing other things.
However if you have a RealTraffic professional license you can run Learner together with PSXT and let it learn your destination airport while you are still preparing your flight or flying to it. This way you will not arrive at an empty airport when you visit that airport for the first time.

If PSXT is learning and adding info it will store (and make further updates in) the airport file in the subfolder `parked_updates`.

PSXT can coexist with third party traffic, such as from VATSIM, IVAO or MyTraffic. However, there is no collision avoidance between third party aircraft and aircraft by PSXT.

PSXT controls landing gear, throttles, spoilers, flaps, and all lights of live aircraft. Of course, only for those aircraft that support the respective SimConnect commands/events.

The load on your computer of PSXT is very low; it consumes less than half percent of your cpu cycles.

If you are new to PSXT and you want to see if you like it or not then you should follow the instructions in the Quick Start Guide for FSX, the Quick Start Guide for P3D or the QuickStartGuide for MSFS.

For questions & answers and discussions use my Support Forum at AVSIM or search the FAQ.

1.1 System Requirements

- **Windows 10** or later

- **Microsoft Visual C++ Redistributable for Visual Studio 2019**
  You need Microsoft Visual C++ Redistributable for Visual Studio 2019. The standalone installers for x86 (`VC_redist.x86.exe`) and x64 (`VC_redist.x64.exe`) can be downloaded here at Microsoft.

Optional, only needed if you want to run PSXT with FSX or P3D:

- **Microsoft SimConnect Client version 10.0.61259**
  The installation file SimConnect.msi can be found in

  ```
  \Prepar3D v5\redist\interface\FSX-SP2-XPACK\retail\lib\SimConnect.msi
  ```

  and in FSX at a similar location. Install the SimConnect Client at the computer running PSXT.

1.2 Disclaimer

The data coming from the RealTraffic is not always correct. This may lead to the following shortcomings:

- Sometimes aircraft do not land at the runway but in the fields …
- Some airlines/aircraft-types always seem to land besides the runway as-if their ADS-B equipment is not sending the right info.
- Aircraft may disappear from the RealTraffic stream for more than 30 seconds which will lead to disappearance from your Flight Simulator while popping up again (a little) later.
- The ADS-B signal may become blocked by buildings at airports which leads to a temporary stop of a taxiing aircraft.
- Some airlines perform much better than other airlines.

Note that if you do not like the movements of AI aircraft at the ground you can uncheck GND traffic in RealTraffic or check the PSXT option to have life traffic only if you are airborne.

Some add-on airports have runways that are not at the exact location.

2 Installation

2.1 Download and install PSXT

Download the PSXT software for your Flight Simulator (MSFS, P3D or FSX) from www.lekseecon.nl. Unzip and run the installer. Installing at the computer running your Simulator will give the smoothest results. For networked use of PSXT see chapter 12.

The default install locations are **`C:\PSXT_MSFS`**, **`C:\PSXT_P3D`** or **`C:\PSXT_FSX`**. You may change it to another location but don't change the installation folder names (PSXT_MSFS, etc).
MSFS: Don't install it in the Community folder!

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

The **`data`** folder contains files with input data for PSXT:

- **`airports_msfs.xml | airports_p3d.xml | airports_fsx.xml`**
  contains information about all stock airports.
- **`dbase.xml`** contains data needed for matching.
- **`fixes.xml`** contains aircraft type fixes that will be automatically made by PSXT.
- **`dontuseAIG.txt`** contains AIG livery titles that do not fly anymore (see 4.7).
- MSFS: **`airports_msfs_names.txt`**

The **`data`** folder has a subfolder, **`regcodesAIG`** that contains files, per airline, with specifications of registration codes for AIG liveries (4.3.2).

The **`liveries`** folder has a subfolder **`regcodes`**, here you can put your text files that specify registration codes for liveries ((4.3.2). Note that this liveries folder appears after you have run PSXT once.

The **`docs`** folder contains the Manual, Release Notes, FAQ, Quick Start Guide and EULA. The 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.

In the **`parked_updates`** folder PSXT will put airport files with added (learned) real live data about what airline is parking at what gate with what type at what hour of the day. Manually created or modified airport files or airport files created by others (like the ones at my web site) should go in here too.
MSFS: At first installation the folder will already be filled with files with parking options for a lot of airports world-wide with ground traffic ("real_only=true"). Parked static aircraft will show up immediately, PSXT_MSFS does not have to learn the airport first.

The **`SimConnect`** folder contains two files that are needed if you want to run PSXT networked. For the use of these see Chapter 12.

The **`stock_airports`** folder:

- MSFS: contains 16.417 airport files.
- P3D: contains 6309 airport files.
- FSX: contains 6450 airport files.

`PSXT_MSFS.exe` is the app for MSFS.
`PSXT_P3D.exe` is the app for all versions of P3D.
`PSXT_FSX.exe` is the app for Microsoft Flight Simulator X (FSX).

`Learner.exe` is an application that updates airport files with new real live parking options without the need for a Simulator connection (11.4) or to be used in parallel with PSXT to learn your destination airport on the fly.

MSFS**:** **`SimConnect.dll`** is needed to run PSXT_MSFS.

On your desktop you will find a shortcut to the executables of PSXT and Learner. You may pin these to Start and/or the Taskbar. Don't move/copy the executable outside its installation folder.

2.2 Update PSXT to a new version

Download the setup file, unzip, and run the installer. It will first uninstall your previous version and then install the new one. **Important**: The files in your `parked_updates` folder is safe, they will not be overwritten or removed.

2.3 Removal of PSXT if you no longer want to use it.

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 too and delete the folder
`C:\Users\<user_name>\AppData\Roaming\PSXT_MSFS` or
`C:\Users\<user_name>\AppData\Roaming\PSXT_P3D` or
`C:\Users\<user_name>\AppData\Roaming\PSXT_FSX`

2.4 Download and install RealTraffic

Preferably install RealTraffic at the same computer as PSXT.
You may install RT at a different computer in the same local area network. Then set a parameter (see 12.3) and set the IP address of the computer running PSXT in the Simulator Host IP address of the RT UI.

3 Operations

3.1 Start and configure Flight Simulator

- MSFS, Prepar3D version 4+ or Microsoft FSX SP2
- Set the day/time to system time
- Fly in 1 x simulation rate (no acceleration)
- Do not use Slew mode
- Use Live Weather, you need the actual QNH to match with the QNH of RealTraffic!
- FSX/P3D: Make sure the Airline Traffic density and General aviation traffic density is set at **0%!**
- FSX: Do not select "Aircraft cast shadows on the ground"
- P3D: Select an aircraft as vehicle (Pedestrians, Avatars, are not allowed)
- MSFS: Multiplayer OFF, Air Traffic OFF, Ground aircraft density: 0

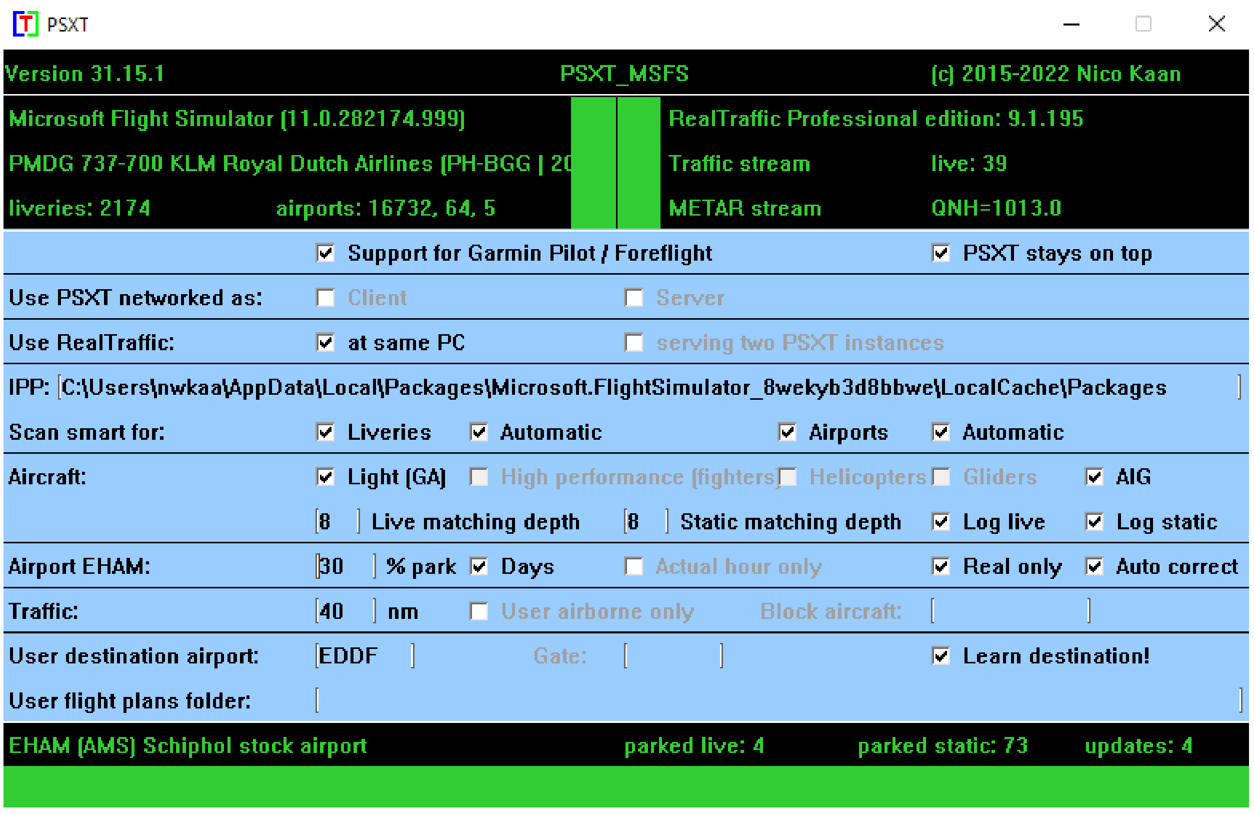
3.2 Start and configure RealTraffic

Select PSXT for P3D/FSX/MSFS for Simulator in use; all the necessary parameters will be set automatically.

Just to be sure: Broadcast traffic to 3rd party apps, Disable PSX Traffic injection, Show GND TFC and Show AIR TFC should all be checked. Auto select GND/AIR TFC should be unchecked, localhost as Simulator Host IP.

3.3 Start PSXT

Start the program via the shortcut on your desktop.


PSXT

Version 31.15.1 PSXT_MSFS (c) 2015-2022 Nico Kaan

Microsoft Flight Simulator (11.0.282174.999) | RealTraffic Professional edition: 9.1.195
PMDG 737-700 KLM Royal Dutch Airlines (PH-BGG | 20 | Traffic stream live: 39
liveries: 2174 airports: 16732, 64, 5 | METAR stream QNH=1013.0

[✓ Support for Garmin Pilot / Foreflight] [✓ PSXT stays on top]

Use PSXT networked as: Client Server

Use RealTraffic: [✓ at same PC] serving two PSXT instances

IPP: C:\Users\nwkaa\AppData\Local\Packages\Microsoft.FlightSimulator_8wekyb3d8bbwe\LocalCache\Packages

Scan smart for: [✓ Liveries] [✓ Automatic] [✓ Airports] [✓ Automatic]

Aircraft: [✓ Light (GA)] High performance (fighters) Helicopters Gliders [✓ AIG]

8 Live matching depth 8 Static matching depth [✓ Log live] [✓ Log static]

Airport EHAM: 30 % park [✓ Days] Actual hour only [✓ Real only] [✓ Auto correct]

Traffic: 40 nm User airborne only Block aircraft:

User destination airport: EDDF Gate: [✓ Learn destination!]

User flight plans folder:

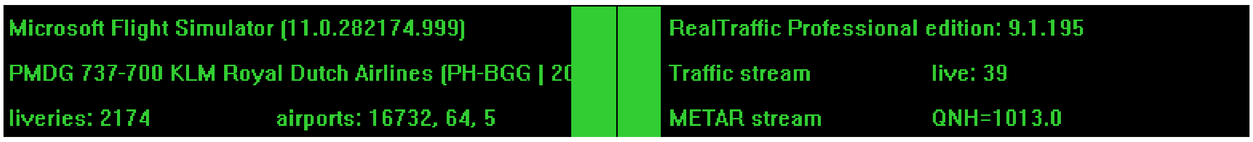
EHAM (AMS) Schiphol stock airport parked live: 4 parked static: 73 updates: 4

PSXT uses semantic versioning: major.minor.patch[-alfa|beta|RC]

Version 31.15.1 PSXT_MSFS (c) 2015-2022 Nico Kaan

At each start-up PSXT will check if there is a new version available. If so, the normally green version message in the main window will turn yellow for a patch update or orange for a minor update, and the new version number will be indicated.
For patches and minor updates there is no obligation to upgrade, for a major version update there is. In the latter case you will be notified by the error message "error 99: This version has expired".

The black box below the header is divided in two sections enclosing two vertical status bars. The left section is about Flight Simulator; the right is about RealTraffic.



```
Microsoft Flight Simulator (11.0.282174.999)              RealTraffic Professional edition: 9.1.195
PMDG 737-700 KLM Royal Dutch Airlines (PH-BGG | 20        Traffic stream          live: 39
liveries: 2174           airports: 16732, 64, 5           METAR stream            QNH=1013.0
```

3.3.1 Flight Simulator info block

This block shows the name and version of your Flight Simulator and the title of the user aircraft. The title is in yellow if the wingspan is not known.

Also shown is the number of ai aircraft (liveries) available for matching and the number of available airport files in your stock airports, addon airports and thirdparty_airports folders *(unless at a client in a networked config)*. If the colour of these numbers is yellow or red the parameters for airports are not set correctly and you better fix that first.

If PSXT is running you may navigate to another airport, load another aircraft (vehicle in P3D) or load a flight (scenery in P3D), PSXT will follow. It will disconnect from your Flight Simulator and reconnect 12 to 30 seconds later.

If you browse through the menus in Flight Simulator, for instance changing options, PSXT will hold and resume if you leave the menu.

3.3.2 RealTraffic info block

This block shows the version of RealTraffic and the license you have got.
The words "Traffic stream" show if the stream with traffic messages is connected. If so the number of live aircraft in your Simulator is shown.

The words "METAR stream" show if the stream with weather messages is connected, the QNH of the closest airport to the user aircraft is shown.

If no data is being received over the last minute the data lights up in yellow colour together with a message in the messages line.

3.3.3 Vertical status bars

In the middle there are two adjacent vertical status bars. The left one indicates the status of the data provider, the right one indicates the status of the Flight Simulator.

If the left vertical bar is red your Flight Simulator is not connected to PSXT. If it is yellow Flight Simulator is connected but your flight is not running. If it is green, it is connected and running.

If the right vertical bar is red your data provider is not connected to PSXT.
If it is yellow, it is connected but PSXT is not yet connected to the traffic messages and QNH streams.

3.4 Parameters

The light blue area in the middle contains the check- and text boxes for parameters that can be set. These parameters and some **hidden** parameters (hidden meaning not available in the User Interface) will be described in the following sections/chapters.

In the
`C:\Users\<user_name>\AppData\Roaming\PSXT_MSFS` folder
`C:\Users\<user_name>\AppData\Roaming\PSXT_P3D` folder
`C:\Users\<user_name>\AppData\Roaming\PSXT_FSX` folder

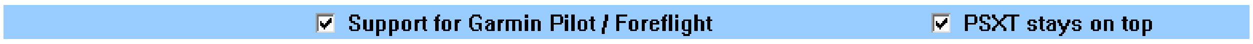
a file **`parameters.xml`** will be created and updated.

You can always go back to all default settings by exiting PSXT, deleting the **`parameters.xml`** file and restarting PSXT.

You must set hidden parameters in the **`parameters.xml`** file manually with a text editor.

Important: How to enter a new value in a textbox in the UI?
Put the cursor in the textbox (indicated by the light square brackets) edit the value, and after editing **press** the **Enter** key to make the new value active!
In the green line at the bottom feedback will be given about your new value.

There are two "meta" parameters in the UI:



☑ Support for Garmin Pilot / Foreflight ☑ PSXT stays on top

3.4.1 Garmin Pilot/Foreflight

Default: **off**

If checked PSXT will send all the data needed by RealTraffic (5 times per second) to support the Garmin Pilot/Foreflight apps. This way you can connect your Garmin Pilot app to RT and follow your flight!

If you are not using this feature, you better switch it off to save processing time.

3.4.2 PSXT stays on top

Default: **off**.

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

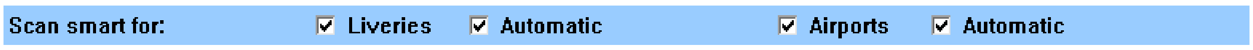
3.4.3 [HIDDEN] SIMCONNECT_TIMEOUT

Default: **15** (seconds), range 10 … 60 seconds.

If SimConnect does not respond to commands from PSXT for this number of seconds, it will conclude it has lost connection and it will try to reconnect to the Simulator.

4 Scanning liveries

4.1 Introduction


Scan smart for: ☑ Liveries ☑ Automatic ☑ Airports ☑ Automatic

PSXT must know what liveries are available for matching in your Simulator.

At the very first start of the program PSXT will always scan for your installed ai aircraft (liveries). The task of the scan is to find the titles, airline codes, aircraft types and registry codes (tail numbers) of AI Aircraft liveries in your flight simulator. The results are store in the file `\liveries\AI_liveries.xml` (note that this is a file for computer-computer communication, do not make changes in it).

At subsequent starts it depends on the value of the **checkbox Liveries (default checked)** if a scan is performed. If you are satisfied with the result of a scan, you can switch it off to speed up the next start PSXT.

This is the behaviour if **hidden** parameter **SCAN_SMART**=false. However, by default SCAN_SMART is true, which implies that if a scan was successful, the **Liveries** check box is switched off automatically. Another advantage of SCAN_SMART is that if it sees that the `\liveries\AI_liveries.xml` file is missing, it will set the **Liveries** checkbox to checked.

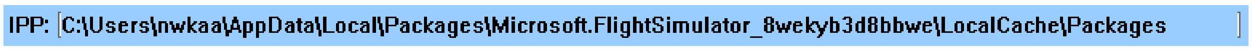
If SCAN_SMART is true, the UI shows "Scan smart for:" , otherwise it shows "Scan for:"

Be aware that if you add new AI aircraft to your Simulator you always must force a new scan by checking the Liveries checkbox!

By default, it tries to find your liveries automatically. That is only possible if it can find your installation path.

MSFS**: IPP (Install Packages Path)** see MSFS FAQ

If found, PSXT_MSFS will show the Install Packages Path in the **IPP** box in the UI**.**


IPP: C:\Users\nwkaa\AppData\Local\Packages\Microsoft.FlightSimulator_8wekyb3d8bbwe\LocalCache\Packages

If not, because you used a custom setting instead of the default location, you must enter a full path to where your Community and Official folders really are. Specify up to (but not included) the Community/Official folders, for example:

`D:\Games\MicroSoft Flight Simulator` if you have **`D:\Games\MicroSoft Flight Simulator\Official`** and **`D:\Games\MicroSoft Flight Simulator\Community`** on your PC.

*Don't forget to press **Enter** when you are finished typing …*

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

Specify max 20 full paths to folders with AI aircraft. The parameters in **parameters.xml** file (see 3.4) are **`lfolder0 … lfolder19`**.

Several examples:

```
<parameter name="lfolder0"
value="C:\Users\nwkaa\AppData\Local\Packages\Microsoft.FlightSimulator_8w
ekyb3d8bbwe\LocalCache\Packages\Community" />
```

```
<parameter name="lfolder0" value="D:\Games\MicroSoft Flight
Simulator\Community" />

<parameter name="lfolder0" value="E:\MSFS Addons\Liveries\aig-aitraffic-
oci-beta" />
```

4.2 How are liveries found?

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

In 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**
title	title of livery
atc_parking_codes	ICAO **airline** code(-s)
atc_parking_types	Cargo aircraft info (bool) if string "**CARGO**" is found

In MSFS it searches in the [General] section of the `aircraft.cfg` file for

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, PSXT_MSFS 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**
title	Title of livery
icao_airline	ICAO **airline** code(-s)
atc_parking_types	Cargo aircraft info (bool), if string "**CARGO**" is found

Special case **IVAO MTL** (P3D/FSX) **/ ivao_x-csl** (MSFS):

The structure of the MTL is different, therefore:

- The aircraft **type** is taken from the **subfolder** name, i.e., "IVAO_**A332**PW".
- The **airline** code is taken from the title in the [fltsim.x] section, i.e. "A332**LTU**".

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

PSXT requests via SimConnect calls the "**Static CG to Ground**" and the "**Static Pitch**" values of each aircraft in your Simulator to know the feet above the ground from centre of gravity and the pitch of the aircraft when on the ground. These values are also in the aircraft.cfg files, so if you see strange things, you should repair it over there.

4.3 Registration codes

4.3.1 In-line specification of registration code

The **registration code** is taken from the **atc_id=** line (see above). Registration codes may be 12 characters max. At most one registration code is recognized. If you want more registration codes to map to the same livery, you must the Offline method in the next section.

4.3.2 Offline specification of registration code(-s)

In subfolder **regcodes** of the **liveries** folder (NOTE: This folder appears after you have run PSXT once!), one may put one or more text files, name is not relevant but 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.

It is not necessary to place all regcodes for a livery at one line, you may split it up into two or more lines, but always in the fixed format, starting with the title.

In case of duplicate use of a registration code, the last one read will be taken and a message about it is generated!

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

This method has the advantage that you do not have to edit the [fltsim.x] sections, which is important when using AIGAIM. Another even more important advantage is that one can share these files. Files for AIG are already available in `data\regcodesAIG`.

Liveries that have just one registration code in an offline file, will be treated as 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;#

4.4 Non-standard ICAO airline codes

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

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

atc_parking_codes=PVT or for IVAO_MTL: title=< **type** >PVT

in aircraft.cfg files that define a private aircraft livery. You can use this code for parking positions in an airport file 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).

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

atc_parking_codes=GOV or for IVAO_MTL: title=< **type** >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 PSXT 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).

4.5 Files about errors found in the search

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.

- **`errors\Fixes.txt`**
  Some obviously wrong Aircraft type codes will be fixed automatically. Fixes are read from the file **`data\fixes.xml`**.
  Some obviously wrong airline codes (found in atc_parking_codes=) are fixed as well. The **`data\fixes.xml`** file contains examples taken from AIGAIM, ORBX and MAIW AI packs. They are converted to PVT or GOV.
  The fixes are applied internally, your **`aircraft.cfg`** file remains untouched.
- **`errors\InvalidAirlineCodes.txt`**
  This file contains invalid airline codes.
- **`errors\InvalidAircraftCodes.txt`**
  This file contains invalid aircraft type codes.
- **`Errors\InvalidAircraftCfgFiles.txt`**
  Contains invalid aircraft.cfg files. Problems may be:
  - [fltsim.x] numbering not consecutive.
  - missing atc_model keys, icao_designator etc.
- **`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 no or an invalid value for the atc_parking_codes key
  - 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.

4.6 Files with information about the search

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

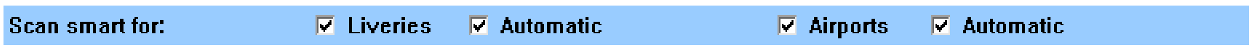
- **`info\AirlineData.txt`**
  Gives an overview by Airline, Type and Folder of the liveries by their registration code(-s), **before** conversion based on **`dbase.xml`**, see below about Airline Type and Registration conversions. This may help you to make corrections.
- **`info\AircraftCfgWithNoFltsimSection.txt`**
  This file gives all the aircraft.cfg file(-s) that are "empty", having no [fltsim.x] section.
- **`info\AirlineTypeConversions.txt`**
  It shows the conversions of the airlines + types 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.
- **`info\OfflineRegcodes.txt`**
  It shows the results of processing the offline registration code files (4.3.2).

4.7 How to suppress historic liveries from a fixed ai aircraft package

In some aircraft packages it is not possible to edit aircraft.cfg files. If you want to suppress certain historic liveries, that no longer fly these days, you can create a file **`data\dontuse.txt`** with the exact title of the livery per line. Then they will not show up, neither as static nor as live.
For AIGAIM such a file is already present in your package: **`data\dontuseAIG.txt.`**

5 Scanning add-on airports

5.1 Introduction



Scan smart for: ☑ Liveries ☑ Automatic ☑ Airports ☑ Automatic

PSXT must know what add-on airports are installed.

At the very first the start of the program PSXT will always scan for your installed addon airports.

The task of the scan is to generate airport files with information about the parking positions. The third-party airport files are stored in the subfolder **`thirdparty_airports`** and (MSFS only) the airport files from Microsoft and Asobo that come with MSFS in **`addon_airports`**.
Before it starts it will clean the **`addon_airports`** and **`thirdparty_airports`** folders!

At subsequent starts it depends on the value of the **checkbox Airports (default checked)** if a scan is performed. If you are satisfied with the result of a scan, you can switch it off to speed up the next start PSXT.

This is the behaviour if **hidden** parameter **SCAN_SMART**=false. However, by default SCAN_SMART is true, which implies that if a scan was successful, the **Airports** check box is switched off automatically. Another advantage of SCAN_SMART is that if it sees that there are no addon airport files available it will set the Airports checkbox to checked.

If SCAN_SMART is true, the UI shows "Scan smart for:" , otherwise it shows "Scan for:"

Be aware that if you add new airports to your Simulator, you always must force a new scan by checking the Airports checkbox!

By default, PSXT tries to find the airports automatically. That is only possible if it can find your installation path.

MSFS only: ➔ see the text about the MSFS textbox in 4.1

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

Specify max 20 full paths to folders with addon airports. The parameters in **parameters.xml** (3.4). are **`afolder0 … afolder19.`**

For example:

```
<parameter name="afolder0" value="D:\Games\MicroSoft Flight
Simulator\Official" />

<parameter name="afolder1" value="D:\Games\MicroSoft Flight
Simulator\Community" />

<parameter name="afolder0" value="D:\Program Files\Lockheed
Martin\Prepar3D v5" />

<parameter name="afolder2" value="D:\Addons" />
```

A folder search includes all sub-folders!

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

5.2 Airport file 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="35" />
</parameters>
```

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

The **supplier** is deduced (if possible) by PSXT from the pathname of the BGL file, while most of the time the suppliers name is part of the path. Stock airports are indicated by supplier=<MSFS|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 Parkpositions

There can be zero or more parking positions (parkpos) in the file. Each **parkpos** has several attributes.

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

The heading is in degrees and the radius is in meters.
For a helipad the cargo and ramp attributes are replaced by attribute **heli="true"** . A helipad has no name in the Sim, so they will be given the name helipad followed by a number.

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

5.4 Airlines for parking positions

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

For example, 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.

5.5 Permanently parked aircraft

You may manually add a section to the airport file that specifies aircraft that will be parked permanently because they are in maintenance or not needed.
For instance, for FlyTampa EHAM P3Dv5:

```
<permanent>
  <parkpos id="P16" airline="KLM" type="A332" />
  <parkpos id="P2" airline="KLM" type="B738" />
  <parkpos id="P3" airline="KLM" type="B78X" />
  <parkpos id="U1" airline="KLM" type="B738" />
  <parkpos id="U2" airline="TRA" type="B738" />
</permanent>
```

For each parking position (indicated by the **id** as being used in the airport file) you should specify airline + type, and optional opf.

5.6 User specific airport files parameters

PSXT will add and update the following parameters to the airport file because they are user specific, as soon as PSXT adds information by Learning.

- *actual_hour_only and real_only*
  These values initialise and store checkbox values 7.3 and 7.4 from the UI.

- *home_base*
  This parameter lists the ICAO airline code(-s) separated by commas of the carriers that have this airport as home base or hub and therefore may appear unlimited at the same time at different positions. Default value is empty, you must add this by hand.

- *max_two*
  This parameter lists the ICAO airline code(-s) separated by commas of the carriers that may have max two aircraft parked at the same time. Default value is empty, you must add this by hand.

  If home_base or max_two is not empty, any airline other than in home_base or max_two may only have max one aircraft parked at the airport at the same time.
  if both home_base and max_two are empty, there will be no special arrangements and airlines will be chosen according to the hour and randomly.

5.7 How and when are airport files used?

When a user aircraft is at an airport, PSXT will search for a suitable airport file. First it searches the **parked_updates** folder. If no file is available, it searches the folder **thirdparty_airports**, if no file is found it searches in the **addon_airports** folder and if again no file is found it finally searches the **stock_airports** folder.

During the session at the airport PSXT will learn and add new info to the airport file. The file is therefore stored and further updated in the **parked_updates** folder, so, it will not be overwritten at the next start of PSXT.

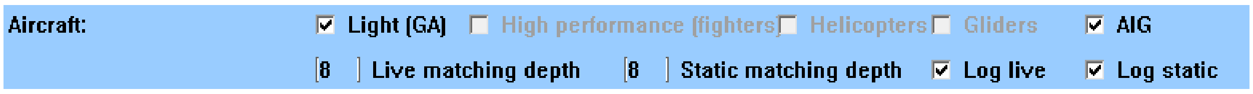
If an airport file is already available in the parked_updates folder, PSXT will still always check if there is not a better file available in **thirdparty_airports** or **addon_airports.** If so, it will delete the file in **parked_updates** and load the better file.

For instance, we had a stock airport in use but bought an add-on airport from a third party. Then the switch to the third-party file will go automatically.

Same in MSFS with third party airports that "overrule" the same add-on airports from Microsoft and Asobo that come with MSFS!
Example, if you bought LSZH from FS Dreamteam it will automatically remove a possibly available add-on file from Microsoft Zurich (OneStore) and load the third-party file from FS Dreamteam automatically. Thus, preventing you from making errors.

6 Aircraft parameters

The following parameters all have effect on the aircraft you will see in your Sim.


Aircraft: ☑ Light (GA) ☐ High performance (fighters) ☐ Helicopters ☐ Gliders ☑ AIG
8 Live matching depth 8 Static matching depth ☑ Log live ☑ Log static

6.1 Aircraft

The aircraft categories are according to the "ADS-B Emitter Category SET" Code Definitions.
Light (GA) (< 15500 lbs.), Small (15500 to 75000 lbs), Large (75000 to 300000 lbs), heavy (> 300000 lbs) and High Performance (>5 g acceleration and > 400 knots).

By default, PSXT shows small, large, and heavy types of live aircraft. Ground Vehicles, Airships and Balloons are not shown. For static and permanent aircraft, you can use any type you want.

You may add light aircraft, high-performance aircraft (F16, F35 and so on), helicopters and gliders. Note that your helicopter liveries should have Category=airplane in their aircraft.cfg file, and not Helicopter.

6.2 AIG

Default: **on**

If checked PSXT will process the info in **`data\dontuseAIG.txt`** (4.7) and in the folder **`data\regcodesAIG`** (4.3.2).
If no AIG liveries are found, the checkbox will be switched off and disabled.

6.3 Live matching depth

Default **8**, range 1 … 8

This parameter determines how many matching steps PSXT will take at max in a search for a livery for a real live aircraft. Set it to 3 if you want perfect matches only, to 5 if you want rather good matches or to 8 if you want as many aircraft as possible no matter how close they are to reality.

From the information coming from RealTraffic the following items are important for live aircraft matching:

- From the call sign the three letter ICAO **airline** code will be derived.
- The type determines the ICAO aircraft **type** (or model).
- The **registration code** uniquely identifies an aircraft (airline, type, and livery).

Next to that PSXT has extra info about what airline and type belong to a registration code (file **`data\dbase.xml`**), thus if there is no airline and/or type found it tries to fix that by registration lookup. Also in that file is info about airlines that operate for other airlines (**opf**).

With the information available a livery for a live aircraft is searched in a best effort matching process in up to eight steps.

Step	Match	Indicated in the Log with
1	Match of an aircraft by **registration code**	=1=
2	Match of an aircraft by **airline** and **type** and **opf**	=2=
3	Match of an aircraft by **airline** and **type**	=3=
4	Match an aircraft by **airline** and similar type of **type**	=4=
5	Match of a generic (**ZZZ**) livery by **airline** and **type**. *This will result in a so called white or house paint.*	=5=
6	Match an aircraft by **type**	=6=
7	Match an aircraft by a similar type of **type**	=7=
8	Match an aircraft in the same aircraft category (Heavy, Middle, Light, Helicopter) as **type**	=8=

There are two special cases:
=c= means "matched" from cache (an aircraft that re-appeared, after a hick-up in RT)
=u= means "updated" same livery but changes in origin and or destination and/or callsign

6.4 Static matching depth

Default **8**, range 1 … 8

This parameter determines how many matching steps PSXT will take at max in a search for a livery for a static parked aircraft. For the steps see the similar steps defines in the previous section.

There is one special cases:

=s= means "matched" from a **snapshot** (7.6)

6.5 Log live

Default **on**.

Logs each live aircraft match.

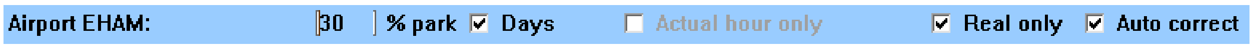
6.6 Log static

Default **on**.

Logs each static aircraft match.

7 Airport parameters

The following parameters all have effect on what happens at the airport, how many and what aircraft will park, do live and static aircraft collide or not, etc.


Airport EHAM: 30 % park ✓ Days ☐ Actual hour only ✓ Real only ✓ Auto correct

7.1 % park

Value is taken from and stored in the airport file, default **35**.

This field shows the percentage parameter of the current airport file. This value determines the minimum number of parking positions at the park airport that will get a parked aircraft, live or static.

Live aircraft are not affected by this parameter, so it could be that you see already more live aircraft at gates than requested as minimum. However, if there are not enough live aircraft PSXT will fill up parking positions with static aircraft to reach the wanted percentage.

At every half past the hour necessary changes in parking positions will be determined and they will be made effective in the coming 50 minutes.

If you change this percentage, the necessary changes will be carried out completely in the next 1 or 2 minutes. The changed percentage value will be stored in the airport file.

At each parking position options for aircraft that park at that position may be added. There are several ways to accomplish this.

7.2 Days

Value is taken from and stored in the airport file, default **off**.

If on, the current day (utc) is added to new parking position options. If time is shifted it will take the day according to the shifted time. When options for static parked aircraft are generated, only options for the current day are considered.

Do note that this checkbox has no effect on the learning function of PSXT and Learner, the day of the option is always recorded in the airport file.

7.2.1 [HIDDEN] REMOVE_DAYLESS_THRESHOLD

Default: **50** %, range 0 … 100

The hidden parameter REMOVE_DAYLESS_THRESHOLD is a percentage. If Days is on and the percentage of parking positions with at least day attribute for 4 different days is greater than this threshold value, all options without a day attribute will be removed from the airport file. This saves processing time; the dayless options are not being used anyway if Days is on.

7.3 Actual hour only

Value is taken from and stored in the airport file, default **off**.

When Actual hour only is off the parking algorithm goes as follows:

> Suppose the actual hour is 14:00 UTC. PSXT first searches in the airport file for positions that have an aircraft at 14:00 hours. If there are not enough positions to park the wanted number of aircraft (*based on the parking percentage in 7.1*), it will widen its time window to 13:00-15:00, and it will look for positions with an aircraft in that broader time window. If there are still not enough found it will again widen its time window to 12:00 - 16:00, and so on, till the wanted number is found or PSXT has searched all 24 hours.

PSXT will go through the above process for as well real=true as real=false options, unless Real only is on (7.4)

However, if Actual hour only is on, it will not widen its time window, so you will only see aircraft parked that have been there at that hour at that gate.

If you change this parameter, the next two minutes the necessary changes will be carried out completely.

7.4 Real only

Value is taken from and stored in the airport file, default **off**.

If on, only options with attribute real=”true” are considered for parking positions (7.3). If you change this parameter, the next two minutes the necessary changes will be carried out completely. The changed value will be stored in the airport file. See also the effect of this parameter on soft learning.

If Real only and Actual hour only (7.3) are both off, and there are not sufficient parking options in the airport file to fill the parking percentage that you have set, PSXT will generate options based on the airlines information found in the <afcad> section of the airport file. Of course, this only happens if there is an afcad section. When you start new with an empty airport file, this gives you a populated airport right-away. The airlines match, only the type and hour may/will be wrong, but PSXT will correct that automatically over time when it starts learning.

7.4.1 [HIDDEN] REMOVE_FALSE_THRESHOLD

Default **50** %, range 0 … 100

The hidden parameter REMOVE_FALSE_THRESHOLD is a percentage. If Real only is on and the percentage of parking positions with a real=”true” attribute is greater than this threshold value, all options with real=”false” will be removed from the airport file. This saves processing time; the false options are not being used anyway if real only is on.

7.5 Auto correct

Value is taken from and stored in the airport file, default **off**.

If on, PSXT will made the following corrections to the airport file *(assuming there are errors in it):*

- If a live cargo aircraft parks at a parking position with cargo=”false”, the cargo attribute in the airport file will be set to “true”.
- If a live aircraft parks at a parking position but its wingspan is greater than the radius, the radius attribute in the airport file will be set to the half wingspan of the aircraft.

If you are confident your airport file is correct, you should switch this function off to protect the airport file. It will also save a little bit of processing (cpu) time.

The value of the checkbox is saved in the airport file because it is airport dependent.

7.6 [HIDDEN] SNAPSHOT_MINUTES

Default **30** minutes, range 0 ... 60 minutes.

If you exit PSXT while you were at an airport, or if you fly away from an airport, PSXT will generate a “snapshot file” named “ICAOAirportCode.xml” in the subfolder **snapshots** of **parked_updates**, with the occupied parking positions (static and live).

If you restart PSXT or if you fly back to the airport within SNAPSHOT_MINUTES, PSXT will place the same liveries back at the parking positions saved in the snapshot file, instead of drawing a new random selection, and hence it does not disturb the immersion.

Note that live aircraft parked will first re-appear as static parked aircraft but as soon as real live data about these aircraft appear, they will become live parked aircraft again with the very same livery, so you’ll not notice any differences.
If a static livery is taken from a snapshot instead of being matched it is indicated by **=s=** if Log matching static is on.

If you set SNAPSHOT_MINUTES=0 this functionality will be disabled, although I see no reason why you would like to do that.

7.7 [HIDDEN] COLLISION_LIVE_PARKED_GND

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 live and static aircraft will be prevented by removing the static aircraft.

If you do not want that set the hidden parameter

COLLISION_LIVE_PARKED_GND=false.

7.8 [HIDDEN] COLLISION_USER_GND

Note that collision detection between the user aircraft and live or static aircraft on the ground always takes place (not in the air).

If you do not want that set the hidden parameter

COLLISION_USER_GND=false.

8 Traffic parameters

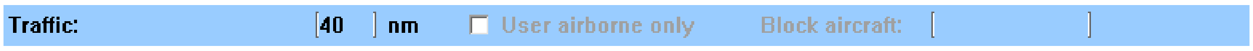
PSXT always adds a simple flight plan with departure and destination to the live aircraft in your Simulator.

In P3D you can then tick Flight Plan in the Information options for Vehicle labels. You will ten see **from -> to** displayed above each live aircraft.

External programs such as **Little Navmap** and **EFB** will benefit and show this information too.

This simple flight plan is just added information, it has no meaning for automated flights. PSXT still controls each aircraft

The following parameters all have effect on the traffic that will show up in your Simulator.



8.1 Set lateral range

Default **40 nm**, range 2 … 100 nm.

All live aircraft within the lateral range of the user aircraft will be injected.

If you are using FSUIPC make sure that the settings for "Limit TCAS range" on the misc tab are equal to the value in PSXT. And don't use the FSUIPC Traffic limiter!

8.2 User airborne only

Default **off**.

If checked, live traffic will be injected in the sim when the user aircraft is airborne **only**!

8.3 Block aircraft

Default **empty**.

If a callsign is entered, PSXT will block a live aircraft with that callsign from appearing in your Simulator or if it is already in your sim, it will be removed immediately. This makes it possible to fly the real live flight instead of the real aircraft (while listening to live ATC) or in landings to take the place of a real aircraft in front of you.

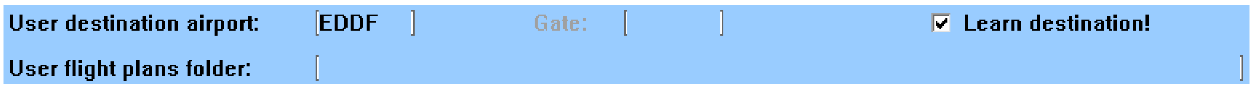
8.4 [HIDDEN] Randomness

Default **on**.

If set to off there will be no randomness in livery matching and no randomness in the gates chosen for static aircraft. You better leave this value on; it is only needed for users running two PSXT's at two different computers at the same time.

9 User parameters

The following parameters all have to do with the aicraft that you are flying (the so called user aircraft).



User destination airport: EDDF Gate: ☑ Learn destination!

User flight plans folder:

9.1 User destination airport

Default **empty**.

Enter here the ICAO code of the airport you are (going to fly)/ flying to.
PSXT will park static aircraft and put live aircraft at the airport when the airport is in sight, otherwise it will start when you have landed.

9.2 Gate

Default **empty**.

Enter the name of the gate, as defined in the airport file if you want the gate to be excluded from a static or live aircraft. This will ensure that your gate is empty when you arrive.
The gate you entered will be checked for validity as soon as the airport file is loaded.

9.3 **Learn destination!**

Default **off**, available if you have a RealTraffic professional license only.

PSXT will make the separate Learner tool (11.4) to learn the destination airport for you. You only must start Learner after you have started PSXT and select Learn destination, PSXT will do the rest.

PSXT will control the Learner tool once you have filled in a Destination airport. This can start while you are still at your origin airport preparing your flight or when you are already airborne.
Learner will connect to RealTraffic Professional and receive a second traffic stream and a second QNH stream for the destination airport.
When the destination airport is in sight, PSXT will force Learner to disconnect from the streams, save the airport file and generate a snapshot file.
These two files are then almost immediately used by PSXT to set up the airport with static aircraft.

This way you'll get the most actual situation!

> You can fly to a destination where you never have been before and still arriving at a populated airport. There is no longer a need to run the Learner program beforehand, although that of course remains possible.

If you want to use Learner in the normal way after you have used it to Learn a destination, you should first deselect Learn destination in PSXT and close PSXT. Then start Learner.

9.4 User flight plans folder

Default empty.

Here you can enter (the path to) to the folder with **.pln** files for the user aircraft. Enter either a subfolder of PSXT or a full path. It may be a full path at the same PC or at a networked PC. In case of networked, use two slashes followed by the PC name followed by a slash and then the name of the share.

PSXT may automatically detect your Destination airport from a **.pln** file exported by a Flight Planner (such as PFPX and ProATC) in your flight plans folder. It will take the latest created **.pln** file with a <DepartureID> equal to your actual origin airport. The destination airport will be taken from the <DestinationID> entry in the Flight Plan.

The airport codes should be according to the ICAO format.

10 Session information

During a run of PSXT a minimum of essential information is provided by several means.

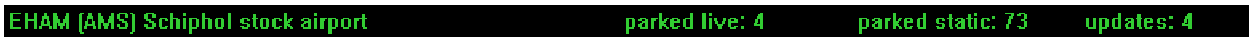
10.1 Log file

During each session a file `Log.txt` will be written in the PSXT folder. The previous version will be renamed to `Log0.txt`, and the previous `Log0.txt` will be renamed `Log00.txt`.

The liveries that you missed are as well logged as kept between runs in a file `MissedLiveries.txt` in the PSXT folder.

10.2 Parking airport line

The black box below the parameters section provides the following information about parking of aircraft at your departure and/or destination based on your airport file.


```
EHAM (AMS) Schiphol stock airport          parked live: 4      parked static: 73     updates: 4
```

<ICAO airport code> (<IATA airport code>) <Airport name> <supplier>

The ICAO and IATA codes of your origin or destination airport and the airport name and supplier from the airport file with parking positions.

parked live: <number>
Shown is the total number of live aircraft parked

parked static: <number>
Shown is the total number of static aircraft parked.

updates: <number>
Shown is the total number of learned updates made to the airport file during the session.

10.3 Messages

In the green box below the status line several kinds of messages may 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. You must exit PSXT with the Windows X button.

#	Error Message *(and what to do)*
1	No valid LIVERIES_FOLDER, STOCK_AIRPORTS_FOLDER, ADDON_AIRPORTS_FOLDER or THIRDPARTY_AIRPORTS_FOLDER parameter(s) found *See 12.1*
5	Learner is already running and connected to RealTraffic *Stop both programs. If you want learning a destination (9.3), start PSXT first, then Learner. If not, choose one program to start (they cannot use RT at the same time).*
41	PSXT should be run from the installation folder! *PSXT's working directory must be the installation folder. Don't remove the* `docs` *folder.*
42	afolderx parameter <> points to stock airports instead of addon airports! *Remove that parameter (0)*
43	Automatic search has not found any liveries, switch off automatic and set lfolderx parameter(-s) *Switch off automatic and set parameters (chapter 3.4.2)*
45	In airport file `<XXXX>.xml` airline or type info for <gate> is missing

	Add the missing part manually or remove the option from the airport file completely
48	Set parameters to your liveries first (or switch automatic scan liveries on) *See Chapter 3.4.2*
49	Set parameters to your add-on airports first *See 5.1*
50	No liveries found *Check your lfolderx parameters (in case of a non automatic scan).*
51	**`AI_liveries.xml`** not found *Scan for liveries first, or in case of networked setup:* *Run PSXT as Server first and/or check the Clients LIVERIES_FOLDER parameter.*
60	**`data\dbase.xml`** not found. *Re-install PSXT.*
61	Some parts of the **`data\dbase.xml`** file is missing. *Re-install PSXT*
71	**`(FSX/P3D only)`** **`atc_model=`** in [General] section of your aircraft.cfg is empty or missing *Add that line to the aircraft.cfg file of your user selected user aircraft. For instance, for the Quality Wings 787 it should read* **`atc_model=B788`** *or* **`atc_model=B789`***.* *The type should be a valid ICAO aircraft type.*
73	It looks like you have not selected an aircraft as user vehicle … *Select an aircraft to fly with or stand somewhere.*
81	There are too many liveries that could not be created, see Log.txt *Activate your AI-aircraft in your Simulator.*
87	xml error in **`airports.xml`** *Re-install PSXT.*
88	Error in converting airport data. *Re-install PSXT.*
89	**`data\airports.xml`** not found. *Re-install PSXT.*
94	Parking position <id> in <file> appears more than once? *Remove redundant parking positions from the file.*
95	Simconnect Version Mismatch exception *Make sure that you are not mixing up P3D and MSFS ….*
97	Airport file name <file> attribute ICAO != <icao> *Correct either the ICAO attribute in the airport file or the airport file name…*
98	This version has been reverted, download version x.y *This "error" is to stop users from keep on using a reverted version.*
99	This version has expired *This "error" has been introduced to make an upgrade to a new full version mandatory (not for decimal increments).*
171	No model info from your user aircraft received in SimConnect *Same problem as #71 for FSX/P3D, the (commercial) aircraft.cfg file might be encrypted*.

11 Learning

The airport files with the parking positions are initially empty. There are several ways to get traffic data, what airline parks where, with what type of aircraft at what hours of the day and optionally at what day of the week, in the airport file.

11.1 Real learning

PSXT will update the airport file automatically in real time (or shifted time). If a real live aircraft positions at a gate or an aircraft park at a gate, the **airline** and **type** combination is added as an option to the airport file with the attribute **real="true"**, plus if Days is on, d**ays="n"**
(n = the code of the day (utc), 1 … 7, 1 = Monday), plus:

- the current hour (**utc time**) is recorded with the attribute **hours=" "**
  the hour is rounded, i.e., 9:30 becomes 10, 23:30 becomes 0, so the range is 0 … 23.
- for arriving aircraft, the next hour is recorded too and for departing aircraft the previous hour.

The entry in the airport file will look like:

```
<aircraft>
  <option airline="KLM" type="E75L" real="true" day="3" hours="17,18" />
</aircraft>
```

The airline and type codes must be according to ICAO rules.

The updated airport file will be stored in the subfolder **parked_updates**.

11.2 Soft learning

If the standard operating procedures (SOP) at an airport forces pilots to switch off their transponder after landing, or to switch it on when they take-off, there will be no traffic visible taxiing from/to gates and PSXT will not learn and thus not update the airport file.

Therefore, PSXT will also use the aircraft that land and/or take off at the airport to update the airport file by adding **airline** and **type** and **hours** to a randomly chosen gate out of the gates that fit best. These entries will get attribute **real="false".** The coding of the hours is the same as in real learning.
Although the gate assignments will not be according to reality, you will at least see parked aircraft of airlines and types that fly to/from the airport at the right hour.

Soft learning is disabled if real only (7.4) is on.

Airport files generated by PSXT have 90% of the aircraft at gate positions and 10% at ramp positions instead of random over all parking positions. *Except for airports starting with a 'K', then 100% at gate positions.*

Most commercial add-on airports have information in their AFCAD file about what airlines park at what gates. PSXT generates an extra **<afcad>** section in the airport file with that info. PSXT uses that info for better soft matches, thus not only that the gate matches the wingspan but also -if possible- that the airline is known at that gate.

PSXT also uses the info from live aircraft parked at gates in the airport file (thus options with attribute **real="true"** -if available- to improve the soft matching.

The updated airport file will be stored in the subfolder **parked_updates**.

PRIORITIES:
If a real=true option is found for a parking position for a certain hour, any real=false option for that same hour will lose that hour in their hours attribute. If no hours are left, that option will be removed from the airport file.

11.2.1 No soft airlines

If you want to prevent that some airlines get soft parked, for instance because you do not want to see business jets parked at gates, you can specify this by adding the file **no_soft_airlines.xml** to the PSXT **data** folder.

The contents of that file should be:

```
<?xml version="1.0"?>
<no_soft_airlines>
    <NoSoft airls="" />
</no_soft_airlines>
```

You may specify ICAO codes for airlines separated by commas. You may specify more than one **<NoSoft airls=** line.

11.3 Manually

You may also add parking positions by hand. If you are sure that they match with real live you should add **real=”true”**, otherwise set **real=”false”**. You also must add an **hour’s** attribute with at least one hour.

If the park position is still empty, like this:

```
<parkpos id="S507" latitude="50.031612" longitude="8.563018" heading="69.9"
radius="50.0" />
```

you should change it into:

```
<parkpos id="S507" latitude="50.031612" longitude="8.563018" heading="69.9"
radius="50.0" >
     <aircraft>
      <option airline="KLM" type="B772" real="true" hours="11,17" />
     </aircraft>
</parkpos>
```

Do not forget to change **/>** into **>** in the first parkpos line!

Second or more options for the same parking position can be added with just one line, like this:

```
<parkpos id="S507" latitude="50.031612" longitude="8.563018" heading="69.9"
radius="50.0" >
     <aircraft>
      <option airline="KLM" type="B772" real="true" hours="11.17" />
      <option airline="DLH" type="B78X" real="true" hours="13" />
     </aircraft>
</parkpos>
```

11.4 Learner

This slimmed version of PSXT 'learns' an airport without the need of a Simulator connection. You only need RealTraffic. Learner can run in the background without further attention.

It is available in each version; run it from the installation folder.

Learner copies the airports and liveries information from PSXT. Therefore, PSXT must have run at least once.

The check boxes control the same functions as in PSXT.

Type the ICAO code(-s) of the airport(-s) wanted.
If you enter one airport code, it will run until you exit Learner. If you type more than one airport code, separated by comma's, it will start with a randomly chosen one and after several minutes (to be set in the textbox below) it will switch to the next one in the row automatically. If it has done the last of the series, it will start with the first one again.

The duration of learning a single airport may be set in minutes between 5 and 10080 (= 1 week in minutes).

If you type ***** and press enter it will take all airport files it finds in your parked_updates folder.
This multiple airport option is handy to let Learner learn a few of your favourite airports in one go, without having to pay attention.

The lateral range is fixed and set to 5 nm around the airport at hand. This is sufficient for learning aircraft that are on the ground at the airport and aircraft that take-off or land.

The updated airport files will be stored in the subfolder parked_updates, just like PSXT does.

All the behaviour described so far is called the **normal mode** (see in the top right section of the UI).

In normal mode you cannot run Learner and PSXT at the same time, while they both need RealTraffic. However, if you buy a Real Traffic Professional License you can run Learner together with PSXT (9.3), this is called **Learning destination mode**.

In a networked PSXT configuration (Chapter 12) you should run Learner at the computer of the Client PSXT. PSXT will issue an error if you forgot that.
(Otherwise, the updated airports would be stored at the server PC and will never be used by PSXT at the Client).

The Log file of Learner is named LearnerLog.txt.

12 Networked use of PSXT and/or RT

The most efficient and recommended way to run PSXT and RealTraffic is at the same PC as your Simulator.

For networked use you must configure SimConnect:

Open the file **SimConnect.cfg** with Notepad and change the IP address in the line **`Address=192.168.1.105`** to show the IP address of the computer running your Flight Simulator and put the file in the installation folder!

Open the file **SimConnect.xml**, that comes with PSXT, with *NotePad* and change the IP address in the line **`<Address>192.168.1.105</Address>`** to show the IP address of the computer running your Flight Simulator.

If you have no other applications using SimConnect with P3D/FSX you will need to install SimConnect.xml as follows:

> P3D: Copy **`SimConnect.xml`** to the computer running P3D and place it in the folder **`C:\Users\<user_name<\AppData\Roaming\Lockheed Martin\Prepar3D v5`**
>
> FSX: similar

Otherwise, if you do have other applications using SimConnect with P3D such as AS16, you will need to **append** the lines starting with **`<SimConnect.Comm>`** and ending with **`</SimConnect.Comm>`**, to your existing Simconnect.xml (its location is given above) above the line **`</SimBase.Document>.`** So, add this block of code, with of course the IP address of your Flight Simulator PC:

```
<SimConnect.Comm>
    <Disabled>False</Disabled>
    <Protocol>IPv4</Protocol>
    <Scope>global</Scope>
    <MaxClients>64</MaxClients>
    <Address>192.168.1.105</Address>
    <Port>29764</Port>
    <MaxRecvSize>8192</MaxRecvSize>
    <DisableNagle>False</DisableNagle>
</SimConnect.Comm>
```

If you want to run PSXT networked, you must install PSXT as well at the computer running Flight Simulator (Server) at the networked computer (Client). Install RealTraffic at the Client. Check the Server or Client checkboxes in both PSXT versions.

Use PSXT networked as: ☐ Client ☐ Server

First let the Server PSXT scan for your liveries and airports in your Sim, and close PSXT when finished. Then connect the client PSXT to the output folders of the Server PSXT by setting three parameters (12.1).

12.1 Use PSXT networked as Client

Default **off**.

If checked, PSXT acts as a client in a networked configuration:

> **`<parameter name="LIVERIES_FOLDER" value="?" />`**
> This parameter specifies the full path to the sub folder **liveries** of PSXT at the Server.

```
<parameter name="STOCK_AIRPORTS_FOLDER" value="?" />
```
This parameter specifies the full path to the sub folder **stock_airports** of PSXT at the Server.

```
<parameter name="ADDON_AIRPORTS_FOLDER" value="?" />
```
This parameter specifies the full path to the sub folder **addon_airports** of PSXT at the Server that contains the airport files.

```
<parameter name="THIRDPARTY_AIRPORTS_FOLDER" value="?" />
```
This parameter specifies the full path to the sub folder **thirdparty_airports** of PSXT at the Server that contains the airport files.

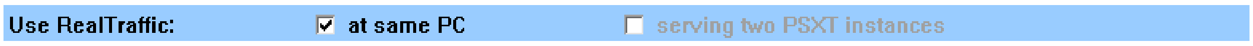
NOTE: for all these 4 parameters holds that you may also copy the stock_airports, addon_airports and liveries folder to the client PC and set the parameters accordingly.

See FAQ #23 with examples how to connect to PSXT when running networked.

12.2 Use PSXT networked as Server

Default **off**.

If checked PSXT acts as a Server in a networked configuration. PSXT will scan for liveries and airports only and then stop. You may exit PSXT afterwards.



Use RealTraffic: [✓ at same PC] serving two PSXT instances

12.3 Use RealTraffic at same PC

Default: **on**

By default, PSXT expects RT at the same PC, if not uncheck this box.

12.4 Use RealTraffic serving two PSXT instances at different computers

Default **off**.

If you have one PSXT (1) + RT combination at one computer and another PSXT (2) at another computer, then check this box for PSXT (2).

This will force PSXT (2) not to set up a TCP IP connection with RT, but it will only listen to the ports on which RT broadcasts its traffic and QNH messages over your LAN controlled by PSXT (1).