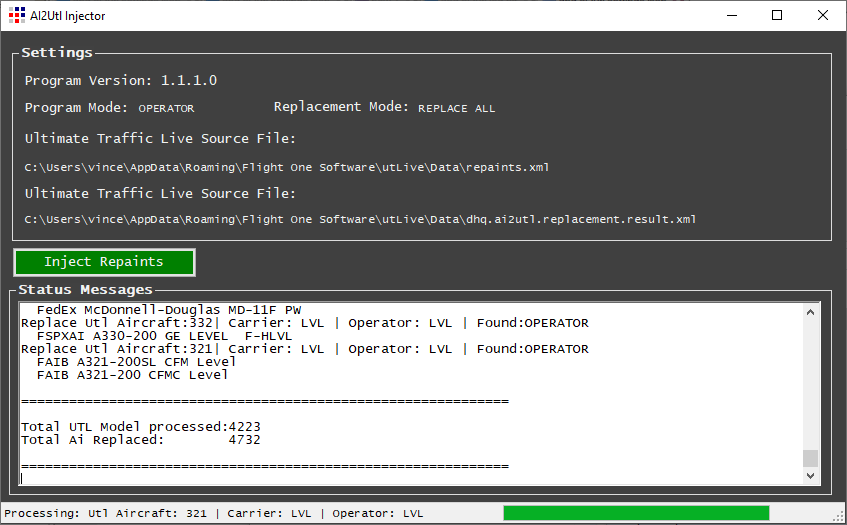
AI2UTL Injector



# Purpose

AI2UTL injector is a tool that automatically searches and replaces Ultimate Traffic Lite (UTL) repaints by corresponding repaints of several AI-packages, like FLAi, Alpha-India, WoA, etc.  
 **It does not change anything** in your P3D or UTL configuration and/or other files.  
All it does is creating a new replacement XML file for your current UTL repaints file. **You manually have to replace the original UTL file (repaints.xml) with this new XML file.**

Please read the whole manual as it contains all information for AI2UTL to run properly.

BE ADVISED

1. **This application is provided ‘AS IS’, no support will be given**.  
   Initially I developed it for myself but decided to share it with others.
2. **You use it at your own risk**.   
   As said before, nothing is changed to your P3D/UTL/whatever configuration.  
   Nevertheless, if you think AI2UTL has caused damage to your system, don’t call me.  
   The application has been written using C#, so if you like you can inspect it using Dot.Peek, ILSpy or likewise tool.
3. **All the usual software and usages disclaimers apply**. Go figure.  
   Most importantly, it may not be included in any pay-ware package or whatsoever

Contents

[Purpose 1](#_Toc1290154)

[Prerequisites 3](#_Toc1290155)

[Advise for new AI2UTL users 3](#_Toc1290156)

[Business Logic 4](#_Toc1290157)

[1. UTL repaint.xml 4](#_Toc1290158)

[2. Search Definition 4](#_Toc1290159)

[3. Included AI repaints 5](#_Toc1290160)

[4. Aircraft.cfg – ATC\_Parking\_codes 5](#_Toc1290161)

[5. Add-only or Replace-ALL UTL repaintS 5](#_Toc1290162)

[6. XML replacement file 5](#_Toc1290163)

[Program Settings 7](#_Toc1290164)

[utl section (3) 7](#_Toc1290165)

[flai section (7) 7](#_Toc1290166)

[miscai section (10) 7](#_Toc1290167)

[AI2UTL repaint inclusion file 8](#_Toc1290168)

[includeOper 8](#_Toc1290169)

[addonlyMissing 8](#_Toc1290170)

[UTL – AI2UTL aircraft code translations 9](#_Toc1290171)

[Log files 10](#_Toc1290172)

[dhq.ai2utl.log.txt 10](#_Toc1290173)

[dhq.ai2utl.error.txt 10](#_Toc1290174)

[Troubleshooting 11](#_Toc1290175)

[Invalid Aircraft Models 11](#_Toc1290176)

[No ai aircraft code(s) defined for Utl aircraft: XXX 11](#_Toc1290177)

[repaints.backup.xml 12](#_Toc1290178)

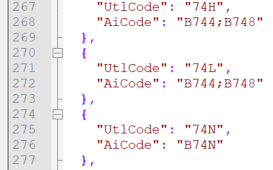
[Json file format 12](#_Toc1290179)

# Prerequisites

* .NET Framework, V4.5 or higher
* One or more installed AI packages (FLAi, Alpha-India, WoA, etc.)
* Flight1 Ultimate Traffic Live (pay-ware)

# Advise for new AI2UTL users

Please read this manual since AI2UTL doesn’t run properly out-of-the-box.  
You need to change some settings and also prepare non-FLAi repaints, like Alpha-India, WoA etc. for use in AI2UTL.  
  
Also, it is strongly advised to experiment with injecting only a very few repaints in UTL initially, so you can validate the application runs as expected.  
When you process all UTL aircraft at once, most likely 100’s of new repaints will be replaced and added. If there is something wrong with a repaint and P3D crashes, it is nearly impossible to troubleshoot which specific injected repaint is causing the issue.  
AI2UTL will automatically validate the type of aircraft model and only use models suitable for FSX and P3D (so called MDLX &PV20 models) but still a specific repaint can be faulty.  
  
You can limited the processed UTL repaints by removing (most) entries from   
the **dhq.ai2utl.search-codes.json** file**.**(Make sure to back up the file so you can use all aircraft after experimenting)For example:



This will only search for replacements for the UTL 738 and 747.

At all times keep a safety copy of your initial UTL repaints.xml so you always can revert back to your original situation.

# Business Logic

How does FLAi2UTL replace UTL repaints?

## UTL repaint.xml

The application loops through the UTL **repaint.xml** file and or each and every repaint it (potentially) searches for any corresponding repaint(s) in the available AI repaint packages.  
**Example UTL repaint entry:**  
 <repaint\_fleet>  
 <equip>330</equip>  
 <car>KLM</car>  
 <oper>AFR</oper>  
 <vis>  
 <repaint\_visual>  
 <title>UTLive\_332.KLM.KLM</title>  
 <val>100</val>  
 <useOnce>false</useOnce>  
 </repaint\_visual>  
 </vis>  
 </repaint\_fleet>

## Search Definition

Based on the entries in the **dhq.ai2utl.search-codes.json** file, the corresponding AI aircraft type code(s) are collected based on the UTL <equip> code (see above).

Example search-code

{

"UtlCode": "330",  
 "AiCode": "A332;A333

},

Since UTL uses IATA codes and AI2UTL ICAO codes, the UTL code needs to be translated. In above example, the ‘330’ code is translated to ‘A332’ and ‘A333’. The latter codes will be used when searching for replacement AI aircraft.

## Included AI repaints

Depending on program settings, **all** FLAI repaints are included in the search. For the other packages, repaints must be manually included (only once). This can be done by simply creating/saving an empty text file in the repaints folder you want to include.   
This file should be called: {AiCode}.ai2utl, i.e. B738.ai2utl  
See also the “UTL – AI2UTL aircraft code translations” section of this manual.

## Aircraft.cfg – ATC\_Parking\_codes

Next, based on the ‘atc\_parking\_codes’ entry in the aircraft.cfg, a match is being made with either the UTL <carrier> code or the <operator> code. The one that is used, is defined in the program settings. When one of these codes match the atc-parking-code, the AI aircraft model is selected to replace the UTL repaint

[fltsim.11]  
title=FLAi\_TFS\_A330-200\_GE\_KLM  
model=GE\NEW  
texture=KLM  
sim=TFS\_A332  
atc\_airline=KLM  
atc\_id=  
atc\_heavy=1  
ui\_manufacturer=The Fruit Stand  
ui\_type=Airbus A330-200  
ui\_variation=KLM  
description=Repaint by Juergen Baumbusch  
atc\_parking\_types=GATE  
atc\_parking\_codes=KLM

After a match is made, automatically is checked if the corresponding aircraft model file (\*.mdl) is compatible with P3D V4. If not, the found AI aircraft will be disregarded.  
Also is checked if the aircraft.cfg is compatible with UTL. It seems that UTL requires a **blank line** above each [fltsim.x] entry.  
All incompatible aircraft models and aircraft.cfg files are listed in the AI2UTL log files.

## Add-only or Replace-ALL UTL repaintS

Then based on program setting, **already existing** UTL-repaint sections can be replaced, or alternatively **only missing** UTL-repaint sections. The latter thus leaves existing UTL-repaint sections unaffected.  
When multiple AI repaints have been found for the same aircraft (i.e. multiple liveries), each repaint will be used equally used in UTL.

## XML replacement file

When all UTL repaints have been processed, a new xml file has been created. (**dhq.ai2utl.replacement.result.xml**)   
**This file fully replaces the UTL repaints.xml file**, you have to manually rename/copy this file to overwrite the original UTL repaint.xml file.

BEFORE:

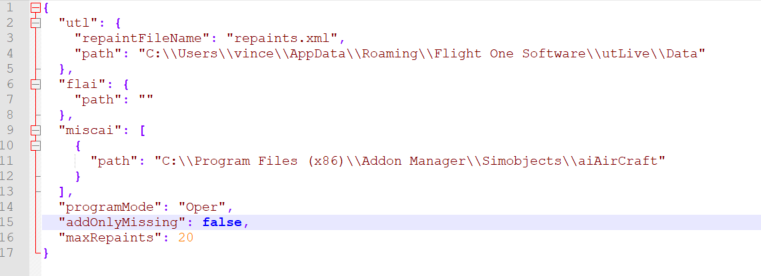


AFTER:



# Program Settings

The file **dhq.ai2utl.settings.json** contains all settings for AI2UTL to run properly.



## utl section (3)

The “utl” section contains the name of the UTL repaint file. This entry should be left unchanged unless Flight1 decides to rename the file.

The “path” setting points to the location where the repaints.xml file resides.   
Normally this should be :  
C:\\Users\\<UserName>\\AppData\\Roaming\\Flight One Software\\utLive\\Data

**IMPORTANT:**Make sure that you use **double backslashes** (\\) for the folder values, otherwise the settings file cannot be read correctly.

## flai section (7)

The “path” setting contains the **root folder** of the FLAi package.  
If you don’t have FLAi AI models installed, leave the setting empty: “path”: “”

## miscai section (10)

Here you can define the **root folder(s)** of one or more other (miscellaneous) AI packages.  
For each package define a path entry.  
   
If you don’t have/want other AI packages to be included, leave the section entry empty:   
"miscai": [], (2 rectangular brackets)  
  
If you want to include only 1 AI package, enter for example:  
"miscai": [

{

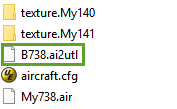
"path": "C:\\Data\\Flai2Utl\\Repaints\\WoA"

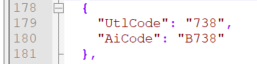
}

],

## AI2UTL repaint inclusion file

When including other AI packages, you also need to include a so called **‘ai2utl’ inclusion file** in each repaint folder you want to include.  
**These inclusion files are not required for FLAi package** because those repaints automatically are recognized by their folder name.

For example, if you have a Boeing 737-800 of Alpha-India, you should add an empty text file named ‘B738.ai2utl’ in the appropriate folder. This way AI2UTL will recognize what type of aircraft the folder contains and includes it consequently in the search.  


The aircraft code in the file name (B738) correspondents to the ‘AiCode’ defined in the **dhq.ai2utl.search-codes.json** file:  
  


If you have more AI folders containing the B738, add to each folder a B738.ai2utl file.  
Then they all will be included in the search.

## programMode

The “programMode” can have 4 values: “Carrier”, “CarrierOnly”, “Oper”, “OperOnly”. “Carrier” and “Oper” indicates that if a repaint is search in by one category first, then the other. “CarrierOnly” and “OperOnly” searches only in each category:

<repaint\_fleet>  
 <equip>330</equip>  
 <**car**>KLM</**car**>  
 <**oper**>AFR</**oper**>  
 <vis>  
 <repaint\_visual>  
 <title>UTLive\_332.KLM.KLM</title>  
 <val>100</val>  
 <useOnce>false</useOnce>  
 </repaint\_visual>  
 </vis>  
 </repaint\_fleet>

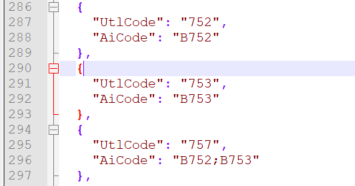
Based on above setting, AI2UTL uses one of these values to match corresponding AI repaints.

## addonlyMissing

If set to **true,** AI2UTL will only add AI repaints for those **missing in UTL**, already existing UTL repaints will be left unaffected. If set to **false, all UTL repaints** will be replaced (of course only for which alternative repaints have been found).

# UTL – AI2UTL aircraft code translations

The file **dhq.ai2utl.search-codes.json** contains (or should contain) all UTL aircraft types and the corresponding AI2UTL aircraft codes:



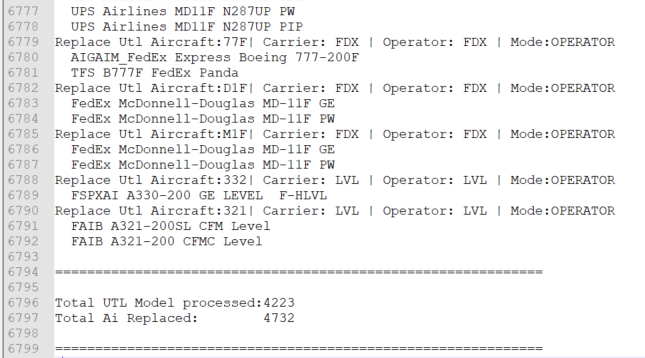
UTL uses IATA codes, AI2UTL ICAO codes, hence the translation.  
Also UTL uses sometimes generic codes, representing multiple aircraft versions. These can be translated to multiple AI codes. (i.e. 757 -> B752;B753)

Also, per aircraft type can be defined which AI packages needs to be included in the search.  
This can be **FLAi** and /or **MISC.** If you don’t want to replace a specific UTL repaint, just leave the “AiCode”  
In case an UTL model code is missing, you can add an entry in the file.  
**IMPORTANT:   
Make sure that each UTL code only appears once** in the file. If multiple entries exist, only the first occurrence is used.

# Log files

## dhq.ai2utl.log.txt

This log file contains extensive information about the replacement process. You can you this log to investigate any issues.



# Troubleshooting

## Invalid Aircraft Models

Please use AI-Aircrat Editor.Net to validate that all of your models are valid:

<http://www.owlsnest.eu/tools.php>

## No ai aircraft code(s) defined for Utl aircraft: XXX

You can encounter one or more warnings like:  
**No ai aircraft code(s) defined for UTL aircraft: xxx**

Although it is not an error, the warning advices you that there is no AiCode(s) defined for the UTL aircraft in the file **dhq.ai2utl.search-codes.json.** Consequently,the aircraft is excluded in the search**.**

For example, the entry below will generate the above-mentioned warning:  
 {  
 "UtlCode": "141",  
 "AiCode": "",  
 "SearchIn": "FLai;MISC"  
 },

## repaints.backup.xml

Please be sure that you have your own backup of the repaints.xml file so that you are able to roll back any changes that was done by the tool:   
C:\Users\<UserName>\AppData\Roaming\Flight One Software\utLive\Data

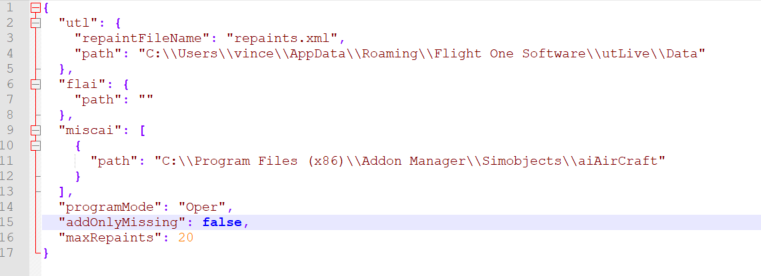
This way you always can revert back to your original UTL repaints in case of issues (P3D crashes etc)

# Json file format

A few remarks about the so called ‘json’ files that are being used for:

* dhq.ai2utl.search-codes.json
* dhq.ai2utl.settings.json

Json is basically a text file with specifically formatted content.



It makes extensively use of brackets like { } and [ ]. Also, fields/settings are separated by commas.

When editing json files, make sure that the structure remains correct. More then often after editing, a bracket is missing, or a comma is misplaced.  
If after AI2UTL startup you get an error message, check the affected json file on these special characters.

**Also, folder names should include double backslashes: \\**