xPipe software package operating manual

XPIPE: RUN BEFORE FLIGHT

Contents

Introduction.....2
System requirements.....3
Installation.....4
  xPipeServer installation.....4
  xPipeViewer installation.....4
  xPipeConnector installation.....4
  xPipeDemo installation.....5
Setup and operation.....5
  xPipeServer: setup and operation.....6
  xPipeViewer: setup and operation.....8
  xPipeConnector: setup and operating.....16
Features of working with xPipeViewer and MFSClient/xPipeConnector.....23
Preparing to fly with xPipeConnector.....24
Document change history.....27
Appreciations.....27
Contacts.....28

Introduction

First of all, I'd like to apologize for text translation quality — the translation from the original language was made mainly by online translators (Google, etc.). I clearly understand that the translation of the text is not completely accurate and unpleasant to read. Unfortunately, I do not know English enough to write texts.

The software package «xPipe» is intended for reception from the flight simulator X-Plane of the data presented in «datarefs» (hereinafter — datarefs), and it's further display and use. The software package «xPipe» (hereinafter — xPipe) includes:

- the plugin for X-Plane, xPipeServer, that acts as a data server;
- the xPipeViewer program that connects to the xPipeServer and displays the datarefs data;
- the set of libraries xPipeConnector allowing the MFSClient program (the client of www.virtairlines.com) to get access to X-Plane flight simulator datarefs;

- the source code of the xPipeDemo demo program in C#, which acts as the xPipeServer client and displays the values of some datarefs.

The scenarios for possible use of xPipe are varied. You can simply use xPipeServer and xPipeViewer to get data from the datarefs you are interested during the flight. If you are using the www.virtairlines.com, you can connect its client MFSClient to X-Plane via xPipeServer using the xPipeConnector library to get a number of advantages over the XPUIPC commonly used in X-Plane. If you have experience in C # programming, you could write your own client for xPipeServer server to get datarefs data from X-Plane.

System requirements

- Microsoft Windows operating system (xPipe was developed and tested on Microsoft Windows 10);
- NET Framework version 3.5 or higher (most likely it is already installed when the operating system updates are enabled, but if it is not installed for some reason, you can download it from this link);
- xPipe was tested in work with the X-Plane version 11.26 (64-bit version), it is assumed that in future versions of the X-Plane xPipe will successfully perform its functions;
- xPipeConnector library set has been tested with MFSClient version 2.2.8.1. It is impossible to be completely sure that the xPipeConnector will work with future versions of MFSClient, we can only hope for it.

The programs included in xPipe are very undemanding of computer resources: according to test results, the average CPU time consumption is about 1%.

Installation

The xPipe comes in a zip archive. You need to unpack its contents in any place convenient for you. The archive consists of directories:

- xPipeServer
- xPipeViewer
- xPipeConnector
- xPipeDemo

Next, you will need to copy the contents of these directories depending on which of the xPipe components you intend to use, which will be described below.

xPipeServer installation

The xPipeServer directory must be copied to the location where the X-Plane plugins are located. This is usually «<X-Plane directory>\Resources\ plugins»

xPipeViewer installation

The xPipeViewer program is «portable», so it doesn't matter where it is located. You can copy the xPipeViewer directory to any place convenient for you. For ease of use, it is recommended to create a shortcut on your desktop for xPipeViewer.exe.

xPipeConnector installation

Before installing xPipeConnector, it is strongly recommended to back up the MFSClient program directory (usually located in «C:\Program Files\MFSClient») or at least back up the FSUIPCClient.dll file contained in the program directory. This file is responsible for the exchange of information between the program MFSClient and XPUIPC (this is a plugin for X-Plane, which emulates the FSUIPC operation protocol for Microsoft Flight Simulator). If for some reason xPipe does not suit you, you can always return the file FSUIPCClient.dll from the backup copy to its original location, and thus return to the standard MFSClient program operating mode.

After all the necessary backups have been made, the contents of the xPipeConnector directory (just the contents, but not the entire directory!) need to be copied to the MFSClient program directory (usually located in «C:\Program Files\MFSClient»). In the process of copying an existing FSUIPCClient.dll file will be replaced with a new file (you will need to confirm overwriting when copying) from the xPipeConnector directory.

xPipeDemo installation

xPipeDemo is a directory containing a project for Microsoft Visual Studio 2017. If you are going to write a program to interact with xPipeServer or want to get an idea of how it all works, then this project will be useful to you. For an example, you can learn the principles of organizing data exchange between xPipeServer and the client program. Otherwise, just ignore xPipeDemo.

Setup and operation

All xPipe programs store settings in settings.xml files, usually located in the same directories as the programs themselves. The settings are read once during the launch of the programs, so if you change any values of the settings, they will not take effect until the programs are restarted. Restarting most programs xPipe made obvious way, but xPipeServer is a plugin for the X-Plane, so it can be restarted as follows:

- restart X-Plane (long way);
- using standard X-Plane tools, stop the xPipeServer plugin, and then start it again (the preferred method).

xPipeServer: setup and operation

The xPipeServer settings are stored in the <X-Plane directory>\Resources\plugins\xPipeServer\settings.xml file. The following is the default content of such a file:

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
  <settings>
    <aquisitionfrequency value="10"/>
    <maxconnections value="2"/>
  </settings>
</root>
```

xPipeServer has two settings: aquisitionfrequency and maxconnections. Let's consider each of them in more detail:

- aquisitionfrequency — dataref reading frequency in Hz (the number of times per second). Possible values: integers in the range from 1 to 50. The default value is 10 Hz.Tests have shown that such a value is sufficient for most xPipe applications. It is important to note that this is the frequency of reading xPipeServer datarefs, but there are no guarantees that xPipeServer clients will receive data at exactly the same frequency, since many extraneous factors can directly affect the transfer of data to the client. *Usually* clients receive the data from the xPipeServer with the expected frequency of reading datarefs, but *sometimes* the frequency can be reduced by approximately 10%.
- maxconnections – maximum number of clients simultaneously connected to xPipeServer. Possible values: integers between 1 and 64. The default value is 2.

You can change the value in the allowable range, but it is not recommended to overestate it without explicit necessity, because for each connection (even unused) with the client xPipeServer consumes some, albeit negligible, the amount of computing power of the computer.

xPipeServer starts work immediately after the launch of X-Plane, even without a running flight. Until the flight has begun (the aircraft is not located at the place of departure), the values of the datarefs, which can be read by xPipeServer, are uncertain. xPipeServer does not assume the function of understanding the content of datarefs data, this task is entirely the responsibility of customers. From this it follows that any clients working with xPipeServer are best run after the flight has already begun.

At the time of launch (and also at the time of stopping) xPipeServer uses the regular X-Plane log file to display service messages about its state. If for some reason xPipeServer could not start, you should look for records that start with the line «xPipeServer: » to analyze the problems in the X-Plane log file.

If the launch was successful, then xPipeServer writes all the information about its work to the xPipeServer.log file located in the plug-in directory (<X-Plane directory>\Resources\plugins\xPipeServer). This file is overwritten every time you run xPipeServer.

To interact with clients, xPipeServer uses the technology of named pipes, opening the «xpipe» channel for data exchange. If for some improbable reason more than one X-Plane instance is running on the computer at the same time, only xPipeServer from the first X-Plane instance can work.

xPipeViewer: setup and operation

The xPipeViewer settings are stored in the settings.xml file located in the program directory. The following is the typical content of such a file:

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
  <settings>
    <serveraddress value="."/>
    <writeflightlog value="1"/>
  </settings>
  <datarefs>
    <acficao path="sim/aircraft/view/acf_ICAO" type="s"/>
    <groundspeed path="sim/flightmodel/position/groundspeed" type="f"/>
    <onground path="sim/flightmodel/failures/onground_any" type="i"/>
    <latitude path="sim/flightmodel/position/latitude" type="d"/>
    <gear path="sim/flightmodel2/gear/deploy_ratio" type="f[]"/>
    <burn path="sim/flightmodel2/engines/engine_is_burning_fuel" type="i[]"/>
  </datarefs>
</root>
```

The settings are grouped into two sections: settings and datarefs. The number of parameters in the settings is constant, but the datarefs section can include any number of «records» related to datarefs, the data of which we would like to see displayed in the xPipeViewer program. For this reason, it is stated above that this is the «typical» content of the settings file.xml: section datarefs can be arbitrarily large or empty (which makes no practical sense, but it is possible).

The xPipeViewer in the settings section has two settings: serveraddress and writeflightlog. Let's consider each of them in more detail:

- serveraddress — xPipeServer location address. If xPipeViewer and xPipeServer are located on the same computer, the value of this parameter must be «.» (dot). *Note that the named pipes technology used for communication between xPipeServer and clients (here the client is xPipeViewer) allows data transmission over the network, but the network mode of operation of xPipe has not been tested, and the task of configuring the network environment for the operability of such data transfer mode is beyond* *the scope of this guide. It is assumed that if xPipeViewer and xPipeServer are located on different computers, the parameter value must be set to the network name of the computer (not the IP address!) with xPipeServer, but again – this mode of operation of xPipe was not tested.*

- writeflightlog — flag of flight logging («black box»). If the parameter value is 1, then after launching the program in the «FlightLogs» subdirectory a file will be created with the name corresponding to the date and time of the program launch, and the extension «.log». In this file xPipeViewer will record all data received from xPipeServer datarefs until the completion of their work. The first line of the file contains a list of pseudonyms of datarefs from the datarefs settings section (this will be described later), separated by a tabulation character. Starting from the second line, the date and time of receiving the data, as well as all actually received data, will be recorded in the order corresponding to the list of pseudonyms of the datarefs, separated by a tabulation character. At the end of the program, the resulting log file can be opened with any suitable spreadsheet editor (Microsoft Excel, OpenOffice Calc, etc.), specifying the «tabulation» character as a column separator for flight analysis. The total size of the log file depends on the number of datarefs, the frequency of data acquisition and the duration of the program. Be careful with this setting turned on — the size of the log file can be significant!

Let's go back to the settings section datarefs. Here all datarefs are listed, the data of which we would like to see displayed in the xPipeViewer program. Using the example of one of them, we will study the description of the format of the dataref, which xPipeViewer will ask xPipeServer for:

```
...
    <acficao path="sim/aircraft/view/acf_ICAO" type="s"/>
...
```

The dataref description consists of three parts (according to colors):

- dataref's pseudonym – dataref identifier inside xPipeViewer, also, if writeflightlog = «1», is used to output to the flight log file («black box») as the name of the data column. Name can be any (subject to restrictions xml-format), but it does need to be unique within the section datarefs settings, it is very important!
- path — points to, in fact, the address of the dataref in X-Plane. The same address of a dataref can be used in several descriptions of datarefs in xPipeViewer.
- type — one of several xPipe dataref formats supported. In most cases, the X-Plane dataref has a uniquely known type specified in the X-Plane SDK, but xPipeViewer does not have this information, so your task is to specify the correct data type.

xPipe supports the following data types:

- i – 32-bit signed integer;
- f – single-precision floating-point number;
- d – double precision floating point number;
- s – character string;
- i[] – one-dimensional array of arbitrary length, consisting of 32-bit signed integers;
- f[] – one-dimensional array of arbitrary length, consisting of single-precision floating-point numbers.

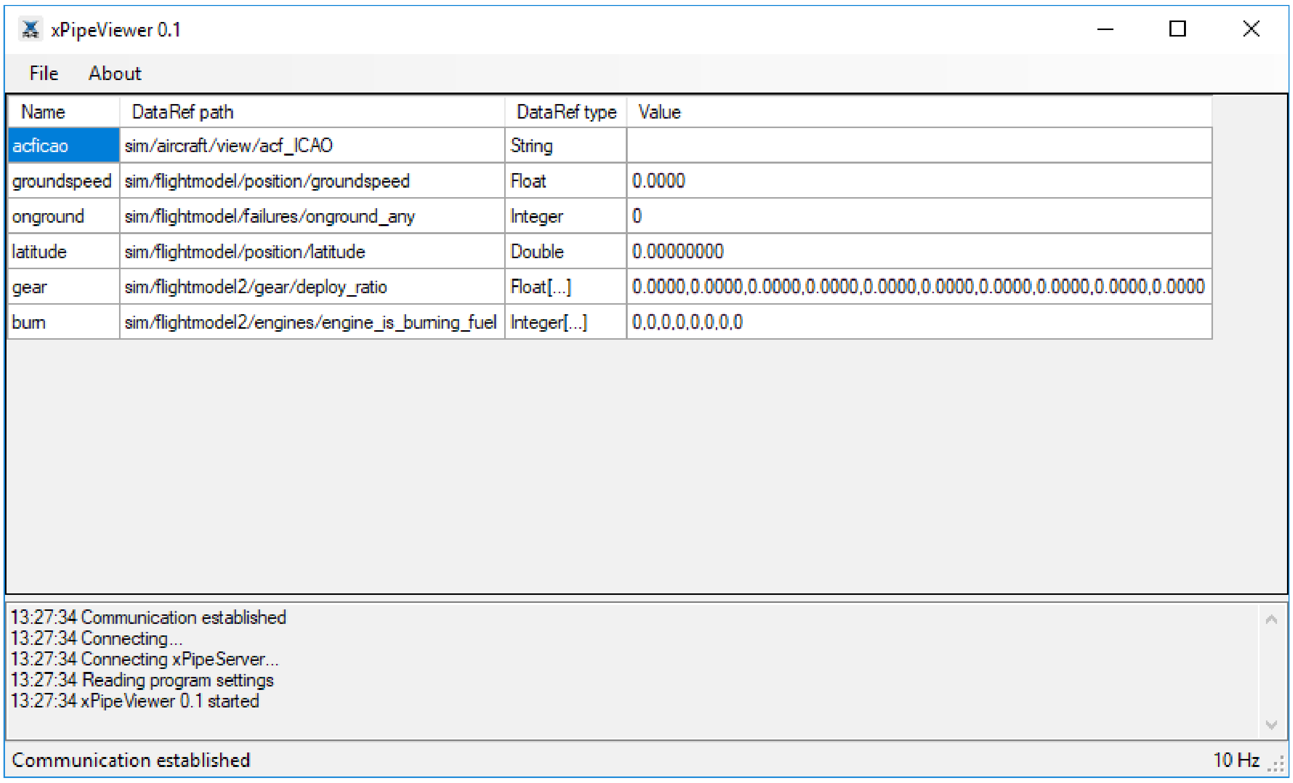
In the X-Plane SDK there are datarefs represented by multidimensional arrays, but xPipe does not support working with such data types.

Let's go back to the datarefs settings section of the source settings.xml file and find out what xPipeViewer will ask xPipeServer here:

```
<datarefs>
  <acficao path="sim/aircraft/view/acf_ICAO" type="s"/>
  <groundspeed path="sim/flightmodel/position/groundspeed" type="f"/>
  <onground path="sim/flightmodel/failures/onground_any" type="i"/>
  <latitude path="sim/flightmodel/position/latitude" type="d"/>
  <gear path="sim/flightmodel2/gear/deploy_ratio" type="f[]"/>
  <burn path="sim/flightmodel2/engines/engine_is_burning_fuel" type="i[]"/>
</datarefs>
```

1. Pseudonym `acficao`, dataref `"sim/aircraft/view/acf_ICAO"`, data type — character string. This is the ICAO code of the model loaded on the X-Plane.
2. Pseudonym `groundspeed`, dataref `"sim/flightmodel/position/groundspeed"`, data type — floating point number of single precision. This is the current speed of the aircraft relative to the ground.
3. Pseudonym `onground`, dataref `"sim/flightmodel/failures/onground_any"`, data type — 32-bit signed integer. This position of the aircraft relative to the ground: 0 – the plane in the air, 1 — the plane of at least one of the landing gear touches the ground.
4. Pseudonym `latitude`, dataref `"sim/flightmodel/position/latitude"`, data type is a double-precision floating point number. This is one of the current geographic coordinates of the aircraft — latitude.
5. Pseudonym `gear`, dataref `"sim/flightmodel2/gear/deploy_ratio"`, data type — one-dimensional array of single-precision floating-point numbers. This is the current position of each of the aircraft landing gear: 0 — completely removed, 1 — fully released.
6. Pseudonym `burn`, dataref `"sim/flightmodel2/engines/engine_is_burning_fuel"`, data type — one-dimensional array of 32-bit signed integers. This is the current state of the aircraft engines: 0 — the engine does not burn fuel, 1 — the engine burns fuel.

So, we figured out what the settings written in xPipeViewer settings.xml mean. It's time to get acquainted with the program interface. Let's launch X-Plane, but for now we will not start the flight, remaining on the start screen, and after that we will launch xPipeViewer (if X-Plane is running as administrator, then xPipeViewer should also be launched as administrator!). When you start xPipeViewer, it tries to establish connection with xPipeServer by itself, you do not need to take any action to do this. In the event of a connection failure, xPipeViewer will also attempt to reconnect on its own. After starting and establishing a connection, we will see the following image:



Name	DataRef path	DataRef type	Value
acficao	sim/aircraft/view/acf_ICAO	String	
groundspeed	sim/flightmodel/position/groundspeed	Float	0.0000
onground	sim/flightmodel/failures/onground_any	Integer	0
latitude	sim/flightmodel/position/latitude	Double	0.00000000
gear	sim/flightmodel2/gear/deploy_ratio	Float[...]	0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000
burn	sim/flightmodel2/engines/engine_is_burning_fuel	Integer[...]	0,0,0,0,0,0,0,0

```
13:27:34 Communication established
13:27:34 Connecting...
13:27:34 Connecting xPipeServer...
13:27:34 Reading program settings
13:27:34 xPipeViewer 0.1 started
```

Communication established 10 Hz

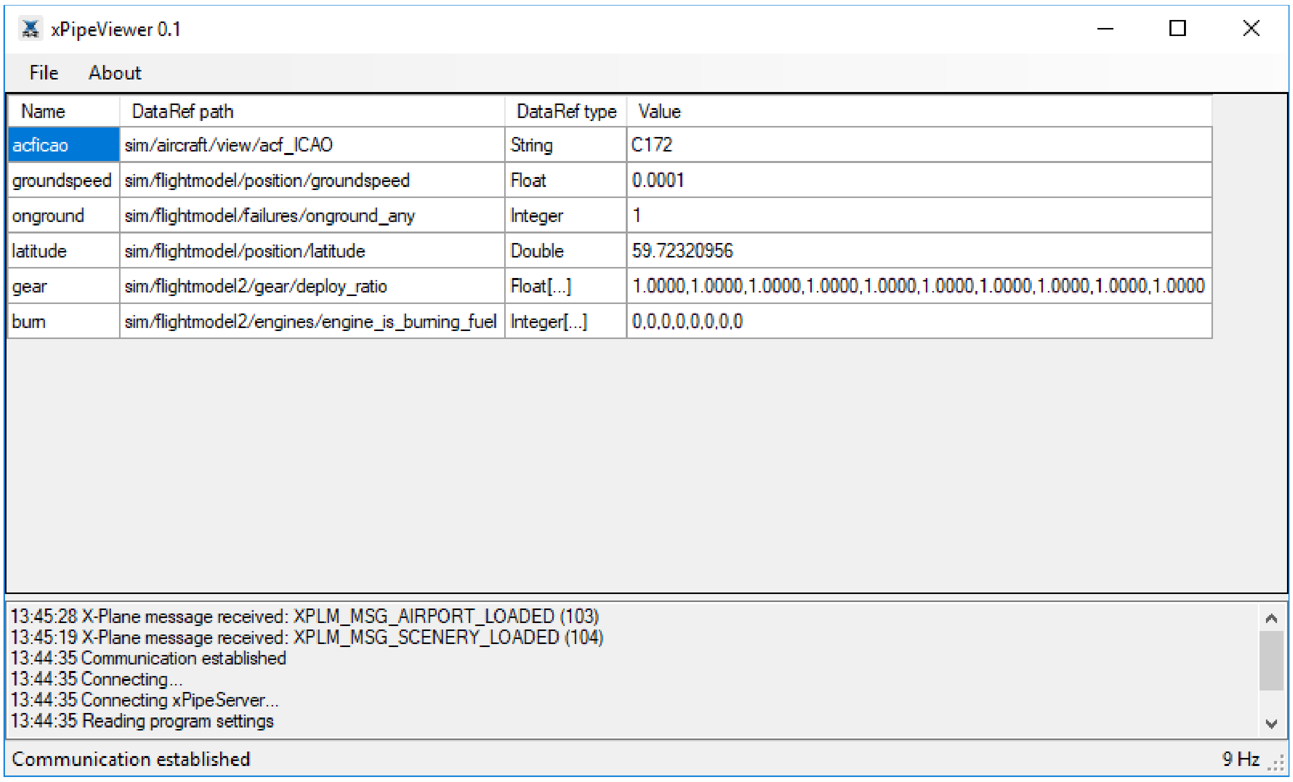
The xPipeViewer interface is standard (top-down):

- system menu;
- data table (columns: pseudonym/name, dataref path, data type, current value);
- a window with a log of several recent events;
- a status bar indicating the status of the connection with the xPipeServer and (if

the connection is established) the frequency of data reception.

If the connection with xPipeServer has not yet been established, the column of current values for all pseudonyms will contain the value «NO DATA» (after the connection is broken, the data obtained last will remain in the column of current values). In our example, the connection is established, so we see in the current values either «0» or a null value — the flight has not started yet, and xPipeServer transmits «undefined» values.

Let's start the flight in X-Plane, choosing the Cessna 172 and an arbitrary airport for this. After loading the flight data appeared in the column of current values:

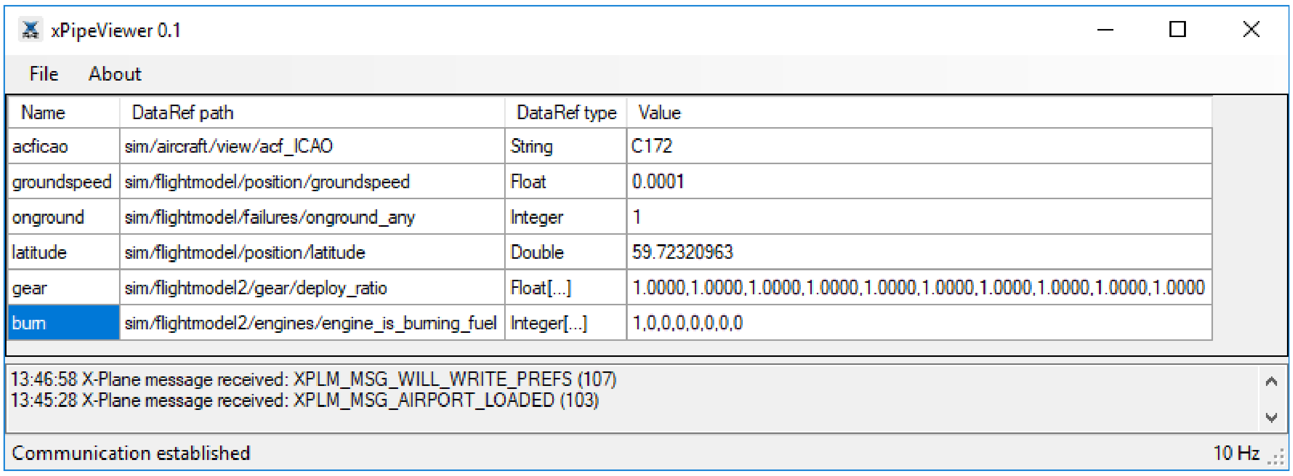


Name	DataRef path	DataRef type	Value
acficao	sim/aircraft/view/acf_ICAO	String	C172
groundspeed	sim/flightmodel/position/groundspeed	Float	0.0001
onground	sim/flightmodel/failures/onground_any	Integer	1
latitude	sim/flightmodel/position/latitude	Double	59.72320956
gear	sim/flightmodel2/gear/deploy_ratio	Float[...]	1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000
burn	sim/flightmodel2/engines/engine_is_burning_fuel	Integer[...]	0,0,0,0,0,0,0,0

```
13:45:28 X-Plane message received: XPLM_MSG_AIRPORT_LOADED (103)
13:45:19 X-Plane message received: XPLM_MSG_SCENERY_LOADED (104)
13:44:35 Communication established
13:44:35 Connecting...
13:44:35 Connecting xPipeServer...
13:44:35 Reading program settings
```

Communication established 9 Hz

Slightly resize the xPipeViewer form so that no empty space is displayed in the data table. Let's start the engine of our Cessna. Please note that the data of the pseudonym «burn» has changed - it was «0, 0, 0, 0, 0, 0, 0, 0», it became «1, 0, 0, 0, 0, 0, 0, 0». There is one engine installed on Cessna, and the dataref showed a change in its state:

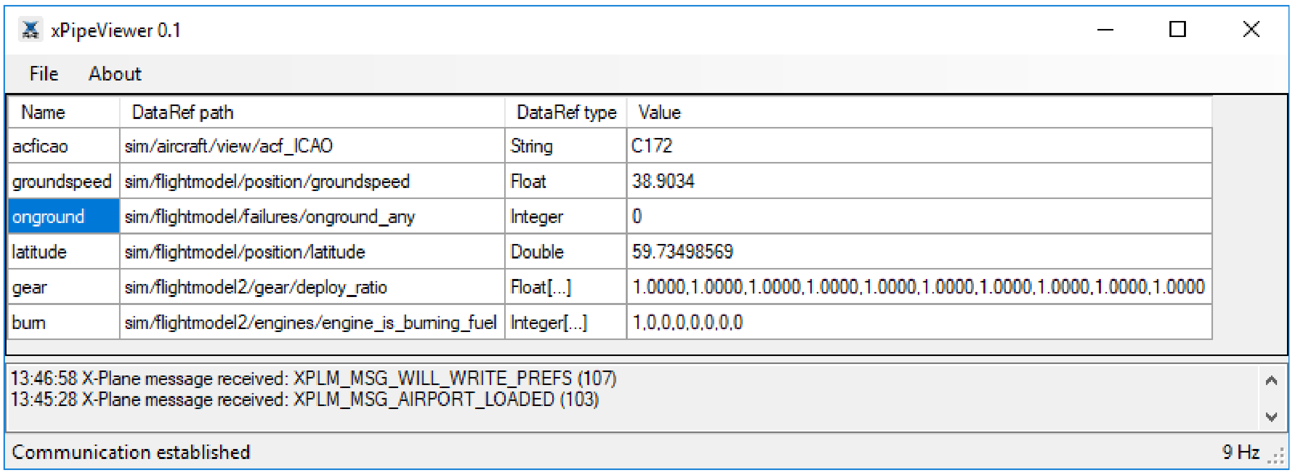


Name	DataRef path	DataRef type	Value
acficao	sim/aircraft/view/acf_ICAO	String	C172
groundspeed	sim/flightmodel/position/groundspeed	Float	0.0001
onground	sim/flightmodel/failures/onground_any	Integer	1
latitude	sim/flightmodel/position/latitude	Double	59.72320963
gear	sim/flightmodel2/gear/deploy_ratio	Float[...]	1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000
burn	sim/flightmodel2/engines/engine_is_burning_fuel	Integer[...]	1,0,0,0,0,0,0,0

```
13:46:58 X-Plane message received: XPLM_MSG_WILL_WRITE_PREFS (107)
13:45:28 X-Plane message received: XPLM_MSG_AIRPORT_LOADED (103)
```

Communication established 10 Hz

Now take off. Current data of pseudonyms «groundspeed», «onground», «latitude» has been changed, which is not surprising — we picked up speed, off the ground and flying some distance, change the geographical latitude of our aircraft. The data of the «gear» pseudonym remained unchanged — the Cessna 172 gears are not removable.

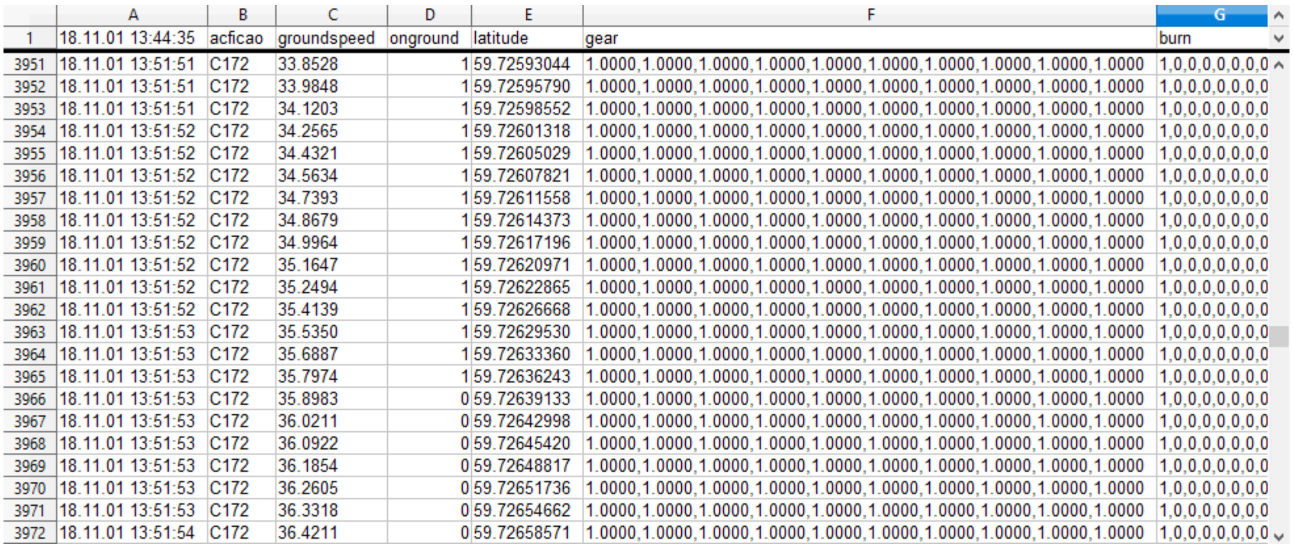


Name	DataRef path	DataRef type	Value
acficao	sim/aircraft/view/acf_ICAO	String	C172
groundspeed	sim/flightmodel/position/groundspeed	Float	38.9034
onground	sim/flightmodel/failures/onground_any	Integer	0
latitude	sim/flightmodel/position/latitude	Double	59.73498569
gear	sim/flightmodel2/gear/deploy_ratio	Float[...]	1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000
burn	sim/flightmodel2/engines/engine_is_burning_fuel	Integer[...]	1,0,0,0,0,0,0,0

```
13:46:58 X-Plane message received: XPLM_MSG_WILL_WRITE_PREFS (107)
13:45:28 X-Plane message received: XPLM_MSG_AIRPORT_LOADED (103)
```

Communication established 9 Hz

Let's complete the flight: close X-Plane and xPipeViewer. Since the writeflightlog parameter was set to «1» in the settings, a flight log file («black box») was created in the FlightLogs subdirectory with the name 181101134435.log. The name of this file makes us understand when it began to fill with data (from left to right): 18th year, 11th month (November), 01st day, time — 13:44:35 (this is the date and time of writing this chapter of the manual). You can open this file in any spreadsheet editor as a text file with the «tabulation» separator column to analyze flight data:



	A	B	C	D	E	F	G
1	18.11.01 13:44:35	acficao	groundspeed	onground	latitude	gear	burn
3951	18.11.01 13:51:51	C172	33.8528	1	59.72593044	1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000	1,0,0,0,0,0,0,0
3952	18.11.01 13:51:51	C172	33.9848	1	59.72595790	1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000	1,0,0,0,0,0,0,0
3953	18.11.01 13:51:51	C172	34.1203	1	59.72598552	1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000	1,0,0,0,0,0,0,0
3954	18.11.01 13:51:52	C172	34.2565	1	59.72601318	1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000	1,0,0,0,0,0,0,0
3955	18.11.01 13:51:52	C172	34.4321	1	59.72605029	1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000	1,0,0,0,0,0,0,0
3956	18.11.01 13:51:52	C172	34.5634	1	59.72607821	1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000	1,0,0,0,0,0,0,0
3957	18.11.01 13:51:52	C172	34.7393	1	59.72611558	1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000	1,0,0,0,0,0,0,0
3958	18.11.01 13:51:52	C172	34.8679	1	59.72614373	1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000	1,0,0,0,0,0,0,0
3959	18.11.01 13:51:52	C172	34.9964	1	59.72617196	1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000	1,0,0,0,0,0,0,0
3960	18.11.01 13:51:52	C172	35.1647	1	59.72620971	1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000	1,0,0,0,0,0,0,0
3961	18.11.01 13:51:52	C172	35.2494	1	59.72622865	1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000	1,0,0,0,0,0,0,0
3962	18.11.01 13:51:52	C172	35.4139	1	59.72626668	1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000	1,0,0,0,0,0,0,0
3963	18.11.01 13:51:53	C172	35.5350	1	59.72629530	1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000	1,0,0,0,0,0,0,0
3964	18.11.01 13:51:53	C172	35.6887	1	59.72633360	1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000	1,0,0,0,0,0,0,0
3965	18.11.01 13:51:53	C172	35.7974	1	59.72636243	1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000	1,0,0,0,0,0,0,0
3966	18.11.01 13:51:53	C172	35.8983	0	59.72639133	1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000	1,0,0,0,0,0,0,0
3967	18.11.01 13:51:53	C172	36.0211	0	59.72642998	1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000	1,0,0,0,0,0,0,0
3968	18.11.01 13:51:53	C172	36.0922	0	59.72645420	1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000	1,0,0,0,0,0,0,0
3969	18.11.01 13:51:53	C172	36.1854	0	59.72648817	1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000	1,0,0,0,0,0,0,0
3970	18.11.01 13:51:53	C172	36.2605	0	59.72651736	1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000	1,0,0,0,0,0,0,0
3971	18.11.01 13:51:53	C172	36.3318	0	59.72654662	1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000	1,0,0,0,0,0,0,0
3972	18.11.01 13:51:54	C172	36.4211	0	59.72658571	1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000,1.0000	1,0,0,0,0,0,0,0

It is clearly seen here, when the value of the pseudonym «onground» changed from «1» to «0» — at this moment the take-off was made.

The attentive reader of this manual may have noticed that in the right lower corner of the xPipeViewer form the frequency of data acquisition changed: in some cases – 9 Hz, in some — 10 Hz. Periodic short-term reduction in the frequency of receiving data by about 10% is considered normal (see the manual section «xPipeServer setup and operation»).

In addition to the datarefs, xPipeViewer receives from xPipeServer X-Plane's system messages relating to the state of the simulator itself: loading the model of the aircraft, livery, scene, airport scene, the aircraft crashing, and others. The acceptance of such messages is registered in the log window of the latest events and in the program log file — xPipeViewer.log.

Thus, knowing the datarefs you need, you can always get data from them using xPipeViewer. A list and a brief description of all the datarefs supported in the current version of X-Plane can be found in <X-Plane

directory>\Resources\plugins\DataRefs.txt

xPipeConnector: setup and operating

Before reading this chapter of the manual, make sure that you have carefully read the previous chapter «xPipeViewer: setup and operation». The fact is that xPipeConnector stores its settings in the same format as xPipeViewer, and in general it is not too different from xPipeViewer in its internal structure. The difference between them is as follows:

- xPipeConnector settings compared to xPipeViewer have additional sections.
- the list and descriptions of datarefs in the settings section datarefs are not recommended to be changed (especially, to delete the elements of the list!), as this may result in the program's inoperability. But if necessary, you can add this list of new items.
- xPipeConnector does not have its own form to display datarefs data, because it transfers them to the MFSClient program.

From the above it follows that you can use the xPipeConnector settings file when working with xPipeViewer, if necessary — xPipeViewer will ignore unfamiliar sections when reading settings. But the opposite is not true — do not use the xPipeViewer settings file in the xPipeConnector, since such a file will not contain the sections necessary for xPipeConnector to work!

The xPipeConnector settings are stored in the settings.xml file located in the MFSClient program directory (usually located in «C:\Program Files\MFSClient»).

Consider the typical content of this file:

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
  <settings>
```

```
    <serveraddress value="."/>
    <writeflightlog value="1"/>
  </settings>
  <flight>
    <stopflightonenginesoff value="1"/>
    <freezevfpmontouchdown value="2000"/>
  </flight>
  <aliases>
    <alias0 acficao="AN24" virtairlines="An-24RV"/>
    <alias1 acficao="YK40" virtairlines="Yak-40"/>
  </aliases>
  <redefinitions>
    <redefinition0 acficao="AN24" file="redefinitions\An24.xml"/>
    <redefinition1 acficao="YK40" file="redefinitions\Yak-40.xml"/>
  </redefinitions>
  <datarefs>
    <acficao path="sim/aircraft/view/acf_ICAO" type="s"/>
    <airspeed path="sim/cockpit2/gauges/indicators/airspeed_kts_pilot"
type="f"/>
    <groundspeed path="sim/flightmodel/position/groundspeed" type="f"/>
    <localsec path="sim/time/local_time_sec" type="f"/>
    <zulusec path="sim/time/zulu_time_sec" type="f"/>
    <day path="sim/time/local_date_days" type="i"/>
    <msl path="sim/flightmodel/position/elevation" type="d"/>
    <latitude path="sim/flightmodel/position/latitude" type="d"/>
    <longitude path="sim/flightmodel/position/longitude" type="d"/>
    <vfpm path="sim/flightmodel/position/vh_ind_fpm" type="f"/>
    <onground path="sim/flightmodel/failures/onground_any" type="i"/>
    <simrate path="sim/time/sim_speed" type="i"/>
    <lighttaxi path="sim/cockpit/electrical/taxi_light_on" type="i"/>
    <lightland path="sim/cockpit/electrical/landing_lights_on" type="i"/>
    <lightbeacon path="sim/cockpit/electrical/beacon_lights_on" type="i"/>
    <lightnav path="sim/cockpit/electrical/nav_lights_on" type="i"/>
    <lightstrobe path="sim/cockpit/electrical/strobe_lights_on" type="i"/>
    <gear path="sim/flightmodel2/gear/deploy_ratio" type="f[]"/>
    <brake path="sim/cockpit2/controls/parking_brake_ratio" type="f" />
    <flaps path="sim/flightmodel/controls/flaprat" type="f"/>
    <windspeed path="sim/weather/wind_speed_kt" type="f"/>
    <winddirection path="sim/weather/wind_direction_degt" type="f"/>
    <pressure path="sim/weather/barometer_sealevel_inhg" type="f"/>
    <agl path="sim/flightmodel/position/y_agl" type="f"/>
    <fuelkgmax path="sim/aircraft/weight/acf_m_fuel_tot" type="f"/>
    <fuelkg path="sim/flightmodel/weight/m_fuel_total" type="f"/>
    <fuelburn path="sim/flightmodel2/engines/engine_is_burning_fuel"
type="i[]"/>
  </datarefs>
</root>
```

The settings and datarefs sections are completely similar to those for the xPipeViewer program (except for the writeflightlog parameter, which is a sign of a flight log ("black box"). If the value of the parameter is equal to 1, then after the start of the program in the subdirectory "FlightLogs" will be created two files with the name corresponding to the date and time of start of the program, and the extension

".log". The file with the ending "-in" contains the data received from the server, and the file with the ending "-out" contains the data sent to the program MFSClient.). Three new sections have been added: flight, aliases, redefinitions — then we look at them in more detail.

The flight settings section and its parameters adjust the MFSClient program's perception of the flight situation.

```
...
  <flight>
    <stopflightonenginesoff value="1"/>
    <freezevfpmontouchdown value="2000"/>
  </flight>
...
```

Consider each of the parameters section:

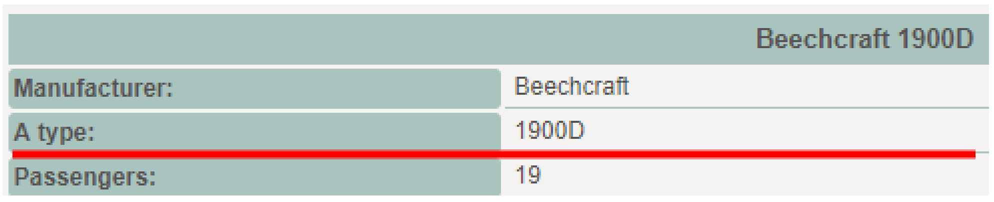
- stopflightonenginesoff — affects the condition of completion of the flight. If the value is «1», then in addition to installing the aircraft on the parking brake, in order to complete the flight, it is necessary that all engines be turned off. You can set this according to your choice («0» or «1»).
- freezevfpmontouchdown — sets the duration of the «freezing» of the vertical speed of the aircraft at the time of touching the ground, the parameter value is set in milliseconds. The fact is that the MFSClient program reads data from the flight simulator approximately once per second, but xPipeServer usually transfers data to clients much more often (with default settings 10 times per second), therefore it is possible to read outdated data. As a result, the vertical speed from MFSClient program point of view may be different from what it was in fact at the moment it touches the ground. «Freezing» the value of the vertical speed of the aircraft at the time of touching the ground for 2 seconds (2000 milliseconds) ensures that the MFSClient program will receive the correct data.

The aliases settings section is intended for a list of «accordances» of ICAO codes of X-Plane aircraft models and aircraft types used in the MFSClient program. For many models of X-Plane aircraft, these data do not match, then in order to correctly recognize the model of the aircraft by the MFSClient program, it is enough to add the «accordance» element in the aliases section.

```
...
  <aliases>
    <alias0 acficao="AN24" virtairlines="An-24RV"/>
    <alias1 acficao="YK40" virtairlines="Yak-40"/>
  </aliases>
...
```

The description of the element of «accordance» consists of three parts (according to colors):

- aliasxxx – «accordance» identifier inside the xPipeConnector. Name can be any (subject to restrictions xml-format), but it does need to be unique within the section setting aliases, it is very important!
- acficao — ICAO code of the aircraft model in X-Plane. You can find out this code by opening the acf-file of the airplane model in a text editor and performing the text search «acf/_ICAO» (without quotes). The next text fragment after «acf/_ICAO» will be the required code. Also, this code can be recognized using xPipeViewer, by adding the dataref "sim/aircraft/view/acf_ICAO" to the list of datarefs and loading the flight with the plane you are interested in.
- virtairlines — aircraft type used by the MFSClient program. The type of aircraft you are interested in can be found at www.virtairlines.com. For example, Beechcraft 1900D, the aircraft type will be «1900D»



	Beechcraft 1900D
Manufacturer:	Beechcraft
A type:	1900D
Passengers:	19

The redefinitions settings section is intended for a list of «redefinitions» of the descriptions of xPipeConnector datarefs. The meaning of such «redefinitions» is as follows: it is possible that for certain X-Plane aircraft models, the datarefs described in the datarefs settings section will not contain the necessary information. In this case, you can override the data sources for such aircraft models to those that will contain the necessary information. At the same time, for other aircraft models, the sources of the datarefs will remain the same.

```
...
  <redefinitions>
    <redefinition0 acficao="AN24" file="redefinitions\An24.xml"/>
    <redefinition1 acficao="YK40" file="redefinitions\Yak-40.xml"/>
  </redefinitions>
...
```

The description of the «redefinition» element consists of three parts (according to the colors):

- redefinitionxxx – «redefinition» identifier inside the xPipeConnector. Name can be any (subject to restrictions xml-format), but it does need to be unique within the section setting redefinitions, it is very important!
- acficao — ICAO code of the aircraft model in X-Plane. You can find out this code by opening the acf-file of the airplane model in a text editor and performing the text search «acf/_ICAO» (without quotes). The next text fragment after «acf/_ICAO» will be the required code. Also, this code can be recognized using xPipeViewer, by adding the dataref "sim/aircraft/view/acf_ICAO" to the list of datarefs and loading the flight with the plane you are interested in.
- file — the relative path to the file that contains the dataref source «redefinitions». Usually the file with the «redefinitions» are placed in a subdirectory «redefinitions».

The file with redefinition of sources of datarefs has a structure identical to the structure of the settings file settings.xml, but contains only one section of settings – datarefs. All the rules for filling the datarefs settings section are fully consistent with those that apply to this section in the settings files for xPipeViewer and xPipeConnector («xPipeViewer: setup and operation»).

We give an example of how this works. Suppose that for some reason you wanted to be able to «turn on» all the light equipment in Cessna 172 with just one switch «taxi lights». At the same time, you should understand that they will be «turned on» only from the point of view of the MFSClient program, since such data from the xPipeConnector will be transferred to it. To do this, create a file C172.xml (the file name can be any, «C172» is taken for clarity), place it in the subdirectory «redefinitions». Write the following text to the file:

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
  <datarefs>
    <lightland path="sim/cockpit/electrical/taxi_light_on" type="i"/>
    <lightbeacon path="sim/cockpit/electrical/taxi_light_on" type="i"/>
    <lightnav path="sim/cockpit/electrical/taxi_light_on" type="i"/>
    <lightstrobe path="sim/cockpit/electrical/taxi_light_on" type="i"/>
  </datarefs>
</root>
```

According to the content of the text, it is clear that the same address of the dataref is now set for all pseudonyms of datarefs — «`sim/cockpit/electrical/taxi_light_on`». The data of this dataref will be transferred into pseudonyms of datarefs `lightland`, `lightbeacon`, `lightnav`, `lightstrobe`.

After that, you need to modify the redefinitions settings section of the settings.xml file of the xPipeConnector, adding the «redefinition» for Cessna 172. Name «redefinition», for example, `redefinition2`, for `acficao` set the value «C172» (this is the ICAO code of this aircraft model in X-Plane), and in `file` set the path to the redefinition file «redefinitions\C172.xml». Now the «redefinitions» settings section will look like this:

```
...
  <redefinitions>
    <redefinition0 acficao="AN24" file="redefinitions\An24.xml"/>
    <redefinition1 acficao="YK40" file="redefinitions\Yak-40.xml"/>
    <redefinition2 acficao="C172" file="redefinitions\C172.xml"/>
  </redefinitions>
...
```

Now you can start the flight in X-Plane and in the MFSClient program (after completing all the necessary procedures for booking a flight, etc. on the website www.virtairlines.com). Turn on the «taxi lights» in the airplane and you will see that the MFSClient program shows that all the lighting systems are on: landing and navigation lights, a beacon, and strobes.

This example shows the simplest use of «redefinitions». But for some models of airplanes with complex internal logic such redirection of data from one dataref to another may not be enough. In this case, you probably can't do it without programming (in one form or another): for example, you can use the Lua script or a plugin to extract the necessary data from the aircraft model and transfer it to a self-created dataref, the link to which is already used in xPipeConnector. But a detailed description of such work practices is beyond the scope of this guide.

We devoted a lot of space to describing the xPipeConnector settings, but the description of working with it will be quite brief — the xPipeConnector does not have its user interface, it works «transparently» for the user, the MFSClient program simply loads it and uses it to receive data from X-Plane. The xPipeConnector receives from the MFSClient program a list of the data of interest to it, sends a request to receive it regularly (considering «redefinitions», if described in the settings) in xPipeServer and is waiting for a response. Having received a response with data from xPipeServer, xPipeConnector transfers data to the MFSClient program, having previously converted (accordingly to it's own settings) to a format understandable for the MFSClient program.

Information about its state xPipeConnector saves the log file xPipeConnector.log, located in the program directory, this file is overwritten each time the program starts. If the «black box» function is enabled in the settings, then all data received from xPipeServer during the session is recorded in the flight log file (see «xPipeViewer: setup and operation») in the «FlightLogs» subdirectory.

In the introduction (see «Introduction») it was mentioned that the xPipeConnector offers several advantages over the XPUIPC package commonly used for communication between X-Plane and the MFSClient program. We list them:

- Known and open datarefs set used for transmission in MFSClient program, and this set can be changed (redefinitions, Lua, etc.), if necessary.
- Supporting «accordances» allows you to fly to X-Plane with the MFSClient program without renaming the model catalogs of aircraft models and/or other non-obvious «magic».
- «Freezing» the value of the vertical speed of the aircraft at the moment of touching the ground ensures that the MFSClient program will receive the correct data.
- In the MFSClient program the flap position and the state of the light equipment are displayed (for aircraft models using the «standard» datarefs).
- Built-in «black box» function, which allows you to find out all the interesting parameters of the flight at its completion.

Features of working with xPipeViewer and MFSClient/xPipeConnector

If an artificially created dataref is used to obtain data (for example, using a Lua script, third-party software), it is important that the xPipeViewer and the MFSClient program (running through xPipeConnector) be run strictly after all the scripts, etc. created an artificial dataref! The fact is that in the overwhelming majority of scenarios of its work, xPipeViewer and xPipeConnector send a request to the xPipeServer data server with a list of necessary datarefs only once, during their (we mean here xPipeViewer and xPipeConnector) launch. Therefore, if at their launch the dataref does not yet exist, then during the entire session, xPipeViewer and xPipeConnector will not be able to receive data from this dataref (most likely the data will look like «ERROR»).

Preparing to fly with xPipeConnector

The content of this section is advisory in nature – as in the opinion of the author, it is more practical to start flying in the system www.virtairlines.com using xPipe.

Since for X-Plane has been created and continues to create a huge number of different aircraft models, it is impossible to guarantee that each of them will be able, without additional settings, «out of the box», to work successfully with the MFSClient program (and therefore — with www.virtairlines.com) via xPipe. Under these conditions, it is reasonable to assume that before performing regular flights, it is necessary to make sure that all the necessary data on the state of the aircraft model are transmitted and correctly perceived by the MFSClient program.

The following algorithm is proposed for checking the model of the aircraft:

1. Create and book a training flight with the aircraft you are interested in.
2. Create a list of faults (for example, a text file). We will hope that it will remain empty, but we must be ready for anything.
3. Launch X-Plane with the aircraft at the airport of departure and the MFSClient program, make sure that the connection between the flight simulator and the MFSClient program is established. If the connection is not established, check the xPipeServer and xPipeConnector log files to eliminate the problem (you may need to restart the MFSClient and/or xPipeServer if it's connection

settings have changed).

4. Make sure that the MFSClient program correctly recognized the model of the aircraft. If this does not happen, add the necessary «accordance» to the aliases xPipeConnector settings section and restart the MFSClient program. Repeat this action point until a positive result is achieved.
5. Launch all systems (electric, fuel, engines and all others), bring the aircraft into a state of readiness for departure.
6. Check the lights equipment: the program MFSClient should correctly display «turning on» and «turning off» for all the necessary light equipment. If the status of some light equipment is not displayed in the MFSClient program, mark them in the list of faults.
7. Compare the weight of the fuel on board according to the flight simulator and according to the MFSClient program. If the difference is greater than 1...2%, mark it in the fault list.
8. Check the extension and retraction of the flaps — the program MFSClient should display the change in the angle of release flaps. The angle of release according to the flight simulator and according to the MFSClient program may differ, it is not so important. If the program MFSClient does not show the change of the angle of release flaps, mark it in the fault list.
9. Check the transfer of the status of the brakes. To do this, you need to use xPipeViewer, the datarefs xPipeViewer settings section must exactly match the datarefs xPipeConnector settings section. Switch on the parking brake (pressure 100%) and check the data in the dataref pseudonym «brake» — the value should be equal to «1». If the condition is not met, mark this in the list of faults.
10. The MFSClient program should display the current state of the landing gear: it must be released when aircraft is on the ground. If the plane has retractable

landing gear — take off and retract it. If the program MFSClient after retracting does not show that the landing gear is retracted — mark it in the list of faults. If you took off — make a landing.

11. In case you have set the xPipeConnector stopflightonenginesoff setting to «1», then to complete the flight, it is necessary (besides the included parking brake) to stop the engines. To check the status of engines use xPipeViewer. When the engines are turned off, the dataref pseudonym «fuelburn» (this is an array of numbers) must contain only the values «0». If the condition is not met, mark this in the list of faults.

If there are no entries in the list of faults — congratulations, the plane is ready to fly in the www.virtairlines.com system using xPipe! Create regular flights and fly in this plane!

What if there is something in the list of faults? Depends on the type of fault. Fatal failures in which you cannot fly in the www.virtairlines.com system using xPipe include problems with the parking brake and engine status (only if you set the xPipeConnector stopflightonenginesoff setting to «1») – after landing at the port of destination you will not be able to complete the flight. Malfunctions with lighting equipment, landing gears, flaps are likely to result in a penalty for their improper use. It is not known what could be the result of a malfunction due to an excessively large difference in the weight of the fuel, but this is not good.

How to fix these faults? Unfortunately, there can be no single answer: it depends on the model of the aircraft. If the model aircraft designers not created any ways of getting data about the internal state of its systems, it looks like to fly on it in www.virtairlines.com system using xPipe be either impossible or would lead to a permanent penalties. If this data can be obtained, then using redefinitions, Lua scripts, etc. you can get the necessary data for the flight in xPipe — you just need to find out it's location, the way of organizing (xPipeViewer and aircraft model documentation will help), but you will need a lot of work to achieve success. What exactly do you need to do? Unfortunately, the discussion of this issue is beyond the scope of this guide.

Document change history

- June 11, 2019 – the chapter «xPipeConnector: setup and operating» has been changed, the writeflightlog parameter has been added. Document version – 0.3.
- November 21, 2018 – added chapter «Features of working with xPipeViewer and MFSClient/xPipeConnector». Document version – 0.2.
- November 16, 2018 — writing the first version of the translation into English is completed. Document version — 0.1.

Appreciations

On behalf of the author xPipe, I would like to thank:

- Anton Shlain — for the main idea, a discussion of the concept of the project, initial beta testing, and the development of the project logo.
- Mikhail Dubovik — for informational support, allocation of space for a project on the virtavia.online community site, assistance with the design and development of the project site.
- Community of virtual pilots VirtAvia — for inspiration, support and communication.

Contacts

xPipe site — xpipe.virtavia.online

xPipe developer – Stanislav Chankov, you can contact me for xPipe issues via forum, VK, FaceBook.